Seminar paper

“mod_ooRexx”

A beginner’s guide for installing and using ooREXX on the Apache webserver.

WIRTSCHAFTS
UNIVERSITÄT
WIEN VIENNA
UNIVERSITY OF
ECONOMICS
AND BUSINESS

Signaturwert	vmium5Oe5kST9B4Aslf+EGzSTCIeg/3nWF8AiXq2BlHNClwzuFiCxM40VjXtOjI8	
	Unterzeichner	Dipl.-HTL-Ing. Robert Maschek
	Datum/Zeit-UTC	2010-01-29T21:02:47Z
	Aussteller-Zertifikat	CN=a-sign-Premium-Sig-02,OU=a-sign-Premium-Sig-02,O=A-Trust Ges. f. Sicherheitssysteme im elektr. Datenverkehr GmbH,C=AT
	Serien-Nr.	291878
	Methode	urn:pdfsigfilter:bka.gv.at:text:v1.1.0
	Parameter	etsi-bka-1.0@1264798967-9155984@30826-12441-0-10994-19345
Prüfhinweis	Informationen zur Prüfung der elektronischen Signatur und des Ausdrucks finden Sie unter: https://www.buergerkarte.at/signature-verification	

Institute for Management Information Systems

WS 2009/10

A seminar paper for the Vertiefungskurs VI – Wirtschaftsinformatik (LV 0902)

Student ID: 8952711

Table of contents

List of figures

List of abbreviations:

API ..... Application Programming Interface
APR ..... Apache Portable Runtime library
ASF ..... Apache Software Foundation
ASP ..... Active Server Pages
DSO ..... Dynamic Shared Object
JSP ..... Java Server Pages
MPM ..... Multi-Processing Module
REXXLA ..... Rexx Language Association
RSP ..... Rexx Server Pages
TCP ..... Transmission Control Protocol
URI ..... Uniform Resource Identifier
URL ..... Uniform Resource Locator
XML ..... Extensible Markup Language
XSL ..... Extensible Stylesheet Language
XSLT ..... Extensible Stylesheet Language Transformation

1 Introduction

The Apache webserver is still the most popular webserver and still the showcase for Open Source Software development with a market share of about 60% according to the most recent surveys from Netcraft [1]

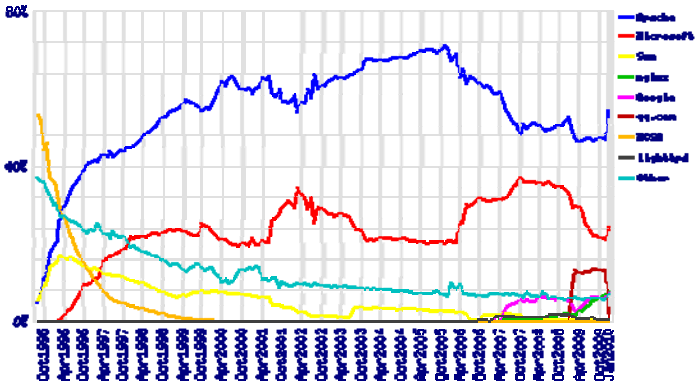

Figure 1 Netcraft Survey of webserver marketshare

There are a lot of different reasons why Apache is still so popular and keeps it market share even when commercial products (like Microsoft IIS) appeared on the market but one reason is its modularity.
Together with being Open Source it has become very easy for developers from all over the world to add modules for specific purposes and makes Apache even more powerful.

Especially there are a lot of modules for the direct integration of programming languages like PHP, Perl or even REXX and ooREXX.

The main focus of this paper is an introduction on beginner level. This means that after a theoretical introduction to Apache's software architecture we will focus on installing mod_ooREXX and getting in touch with it providing different examples.

[1] http://news.netcraft.com/archives/web_server_survey.html (2010-01-24)

2 The Apache Webserver

2.1 History

The roots of the Apache webserver date back to the early days of the Internet and the NCSA-httpd daemon developed by Rob McCool at the National Center for Supercomputing Applications (NCSA), University of Illinois.
When Rob was leaving the NCSA the further development of the web server got stuck even having a big installed base at this time.

So a group of administrators joined a team in developing patches for the server and in April 1995 the version 0.6.2 of the “a patchy server” was officially released even if the Apache software foundation claims that “The name 'Apache' was chosen from respect for the various Native American nations collectively referred to as Apache, well-known for their superior skills in warfare strategy and their inexhaustible endurance.” [2]
The codebase was completely redesigned and a lot of additional features were added until December 1995 when the version 1.0 was released.

In February 1997 after the release of the version 1.2 the webserver was undergoing another major redesign. The aim for 2.0 was to ensure that the same code is used for all operating systems with a set of platform-specific routines to handle anything that varies between operating systems.
All the plans for Apache 2.0 were summarized in February 1998, Apache Week issue 102[3]. The major changes being discussed were multithreading, filtering, new process models, better system configuration, API changes and changes to the configuration syntax.

The alpha version of Apache 2.0 was presented at the ApacheCon in 2000 followed by the Beta version in 2001. The first general available version 2.0.35 of the web server was released in April 2002.

[2] http://www.apache.org/foundation/faq.html#name (2010-01-24)
[3] http://www.apacheweek.com/issues/98-02-13#apache20 (2010-01-24)

2.2 Apache 2 Software Architecture

2.2.1 Overview

The Apache HTTP server in the version 2 consists of a relatively small core and some modules which may be compiled statically into the server or held in a specific directory (/modules or /libexec) and loaded dynamically at runtime[4].

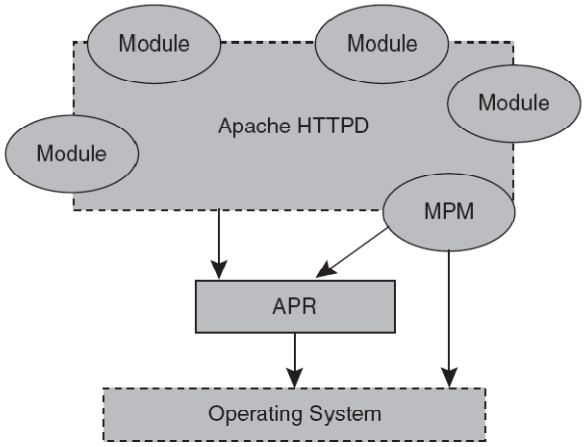

Figure 2 Apache software architecture overview (Kew 2007, p. 22)

The two most important parts of the architecture are the Apache Portable Runtime (APR) library and the Multi-Processing Module (MPM).

The APR provides a standard application programming interface (API) for the core of the Apache so that developers don't need to care about operating system calls like accessing files and memory management for example[5]. According to the Apache Software Foundation (ASF) "The mission of the Apache Portable Runtime (APR) project is to create and maintain software libraries that provide a predictable and consistent interface to underlying platform-specific implementations. The primary goal is to provide an API to which software developers may code and be assured of predictable if not identical behavior regardless of the platform on which their software is built, relieving them of the need to code special-case conditions to work around or take advantage of platform-specific deficiencies or features."[6].

[4] Kew (2007), p. 21
[5] Wolfgarten (2003), p. 19
[6] http://apr.apache.org/ (2010-01-24).

The APR maintained by the ASF is not only used for the webserver. It is also used for other Open Source projects like Tomcat[7], an open source software implementation of the Java Servlet, and JavaServer Pages technologies or Subversion[8], an open source version control system for software development.

The MPM is used to optimize Apache for the underlying operating system so that incoming requests are mapped onto an execution primitive which could be either a process or a thread depending on the operating system. For this reason it is the only module talking directly to the operating system. Any other module uses the APR for communication with the operating system.

To really understand the need for MPMs, it is important to look at how the former versions of Apache (up to version 1.3) are working. These versions are pre-forking servers, meaning that when Apache is started the original process forks a specified number of copies of itself, which actually handle the requests. As more requests come in, more copies are forked. The original process doesn't actually do anything other than monitor the new processes to make sure there are enough of them. This model works well on Unix variants and most mainframes but it doesn't work as well on Windows. The original support for Windows actually re-wrote the section of code that created the child processes. On Windows this section created just one child process, which then had multiple threads to serve the requests.

Apache 2 now supports three different kinds of MPM. Some of them are used as a default for the underlying operating systems (shown in brackets): [9]

- Process based
  MPM module: prefork (Unix) and beos (BeOS)
- Thread based
  MPM module: mpm_netware (Netware) and mpm_winnt (Microsoft)
- Hybrid mode which is a mixture of process - and thread based
  MPM module: worker, event and mpmt_OS2 (OS2)

[7] http://tomcat.apache.org/ (2010-01-24)
[8] http://subversion.tigris.org/ (2010-01-24)
[9] Wolfgarten 2003, p. 19; http://httpd.apache.org/docs/2.2/en/mpm.html (2010-01-24)

Each of these MPMs has strengths and weaknesses. For example, the process based MPM will be more robust than the hybrid MPMs on the same platform. The reason for this is quite simple: if a child process terminates unexpectedly, connections will be lost. How many connections are lost is up to which MPM is used. If the process based MPM is used, one connection will be lost. If a thread based MPM is used, no more than 1/n connections, where n is the number of child processes used, will be lost. If the hybrid based MPM is used, the number of lost connections will depend on the OS the server is being run on.
However, the trade-off in robustness comes at a price: scalability. The process based MPM is the least scalable MPM, followed by thread based, and then hybrid. Which MPM is used will depend on what the site requires. If a given site must use a lot of third-party nontrusted modules, then that site should use the process based MPM because if the module is unstable, it will affect the site the least. However if all a site is going to do is serve static web pages and doesn't require any modules but will need to serve thousands of hits per second, then the hybrid based MPM is probably the correct choice[10].

2.2.2 Operation

Apache operation consists of two phases:

- Startup phase
- Operational phase

Actually there is no shutdown phase for Apache 2. Everything that needs to be done is registered as a cleanup and run when the application stops[11].

During the startup phase Apache reads and verifies the configuration files. The main configuration file of the Apache web server is a plain text file called httpd.conf. Be careful: This is just a convention. There are binary distribution available (like for Debian GNU/Linux) which are using totally different conventions.
Further actions include loading modules, open network connection, and initialize system resources such as log-files, shared memory segments [12]

[10] http://www.serverwatch.com/news/article.php/10824_1129161_1/An-Introduction-to-Apache-20.htm (2010-01-24)
[11] Kew (2007), p. 26).
[12] Kew (2007) p. 22; Ford (2008), p. 2

During this time Apache runs as a single-process, single-thread program and has full system privileges.
Before entering the operational phase Apache relinquishes its system privileges. This basic security measure helps to prevent a simple bug in the software or a module from becoming a very dangerous system vulnerability like seen with “Code Red” or “Nimda” on the Microsoft IIS[13].

At the end of the startup phase the control is passed to the MPM which handles Apache’s operation at system level.

Once Apache has entered its operational state the child processes or threads will accept external connections.

2.3 Apache2 Basic Concepts and Structures

When it comes to system programming with Apache you need to know more about the core objects and some the basic concept of “Pools” which are used to represent the operation within the webserver.

2.3.1 Basic concept: Pools

APR pools are a main part of the Apache application design and are a grouped collection of resources (i.e., file handles, memory, child programs, sockets, pipes, and so on) that are released when the pool is destroyed. Almost all resources used within Apache reside in pools, and if you don’t want to use them you should really be very careful.

Pools can be hierarchically structured (Pool → Subpool → Subsubpool and so on).
Another interesting feature of pool resources is that many of them can be released only by destroying the pool. When a pool is destroyed, all its subpools are destroyed with it.

[13] Kew (2007) p. 22

During the startup phase Apache creates a pool from which all others are derived. Configuration information is held in this pool (so it is destroyed and created new when the server is restarted).
The next level of pools is created for each connection Apache receives and is destroyed when the connection ends. A connection can span several requests and a new pool is created (and destroyed) for each request.

In the process of handling a request, various modules create their own pools, and some also create subrequests, which are processed like real requests. Each of these pools can be accessed through the corresponding structures (i.e., the connect structure, the request structure, and so on).

Is there a reason for not using pools?
For sure there is. In Apache prior to version 2 you can't use pools when the lifetime of the resource in question is bigger than the lifetime of the top pool.
"Apache 2.0 gives us both a new example and a new excuse for not using pools. The excuse is where using a pool would cause either excessive memory consumption or excessive amounts of pool creation and destruction, and the example is bucket brigades (or, more accurately, buckets)."[14].
A bucket brigade contains a sequence of buckets which represent both data content and metadata and are especially used in connection with filtering[15].

2.3.2 Apache core objects

Basically Apache provides the following core objects[16]:

- process_rec
- server_rec
- conn_rec
- request_rec

[14] Laurie, Laurie (2003) p. 406
[15] http://httpd.apache.org/docs/trunk/developer/output-filters.html (2010-01-24)
[16] Kew (2007) p. 29

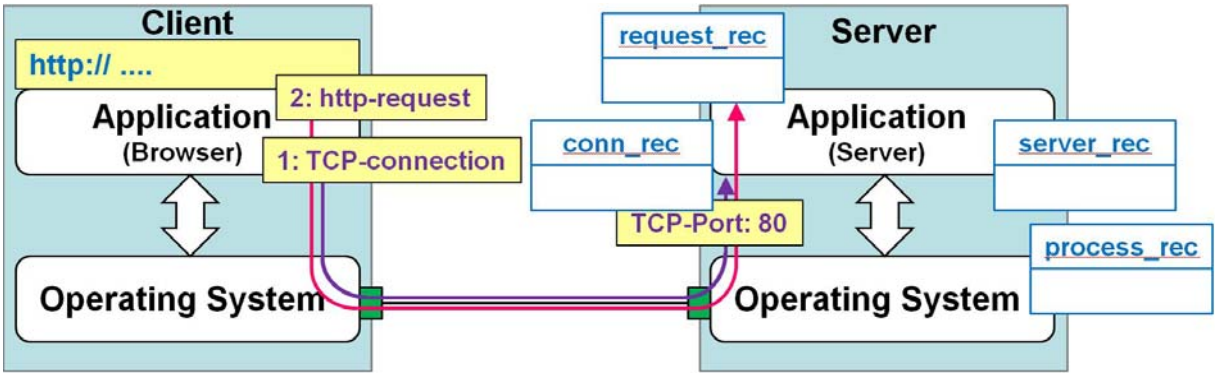

Figure 3 Apache core objects

2.3.2.1 process_rec

The process_rec object is considered to be more a part of the operating system than the belonging to the actual webserver.
The main goal for this object is the handling of pools (see chapter 2.3.1 for more details).

2.3.2.2 server_rec

This object is created during the startup phase and defines a logical webserver[17]. Since Apache is able to run “virtual hosts” (means running multiple instances of the webserver on the same box) this object is created separately for each virtual server.

The object exists as long as the httpd daemon is running.

The server_rec object also uses only process pools maintained by the process_rec object. It does not have any own pools[18].

2.3.2.3 conn_rec

The core objects described so far are created during the startup phase of the server and are available even there is no work for the server.
The conn_rec object is created when a client connects to the webserver.

There is a very important difference between (http) request and a connection.

[17] Kew (2007) p. 35
[18] Kew (2007) p. 35

One connection can handle an amount of (means more than one) requests or more technical – "the former is always a subcomponent of the later."[19]
For this reason there can be more than one request_rec object (see chapter 2.3.2.4) instantiated from one conn_rec object.

Mostly this object is ignored by programmers. You have to deal with it mainly when you are writing connection-level filters or any kind of protocol module.

2.3.2.4 request_rec

The request_rec object is often called the heart and the soul of the Apache webserver. This object deals with handling the http requests and is defined in the httpd.h file. It is created whenever a request is accepted, stores and processes all the relevant data for all stages of the entire request handling process (see chapter 2.4 on page 10) and is destroyed when the request is finished. This object is passed to every event handler implemented by any module[20].

It also includes a number of fields used internally to maintain state and client information by the webserver needed for processing the request:[21]

- A request pool, for management of objects having the lifetime of the request. It is used to manage resources allocated while processing the request.
- A vector of configuration records for static request configuration (per-directory)
- A vector of configuration records for transient data used in processing.
- Tables of http input, output, and error headers.
- A table of Apache environment variables as seen in scripting extensions and a similar "notes" table for request data that should not be seen by scripts.
- Pointers to all other relevant objects
- Pointers to the input and output filter chains.
- The Uniform Resource Identifier (URI) requested and the internal parsed representation of it, including the handler and filesystem mapping.

[19] Kew (2007) p. 37
[20] Kew (2007) p. 30
[21] Kew (2007) p. 30

2.4 HTTP Request handling

2.4.1 Introduction

Now we know about the core objects making the webserver up and running. So it is time to deal with the real “business” of such a server: Processing http requests from clients.

At this stage we assume that the needed Transmission Control Protocol (TCP) connection between the client and the server is already established.

Apache splits the request handling in different phases.

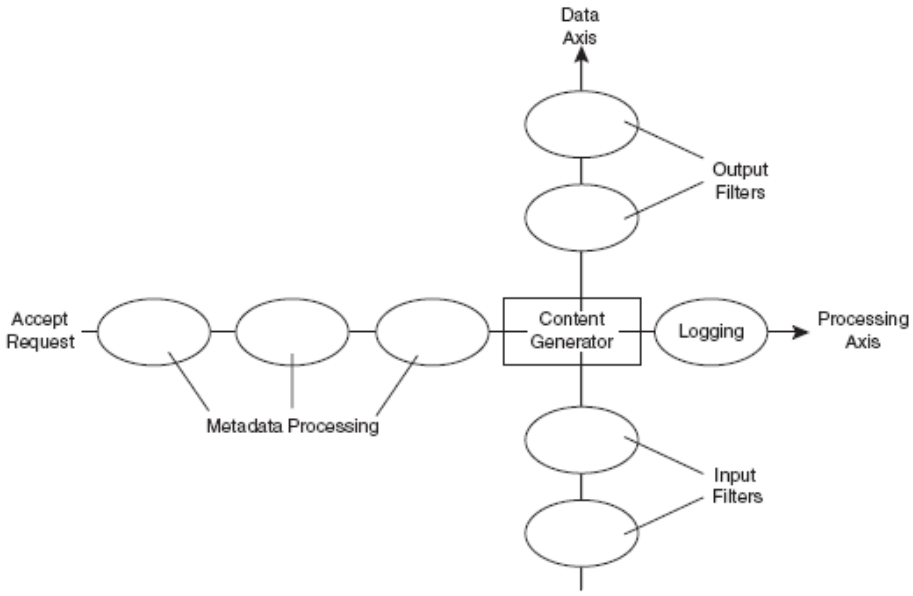

Figure 4 Apache 2 request handling [22]

The “Processing Axis” describes the basic operation of a webserver.
After accepting the request during the “Metadata Processing” phase processes like checking user authentication or processing of the Uniform Resource Identifier (URI) mapping take place.
As soon as this is finished it comes to the “Content Generation” phase. Here the server “creates” the webpage. By default it sends a file from the local disk.
Before returning to the client the request passes the “Logging” phase where the logfiles are written.

[22] Kew (2007) p. 47

Apache 2 implemented a new concept: The filter chain was introduced and is represented by the “Data Axis” in Figure 4. The reason for this was to enable a much cleaner and more efficient data processing after or prior to reach the content generator.
Filter might be used especially for modules that have both data inputs and outputs like transforming Extensible Markup Language (XML) pages using the Extensible Stylesheet Language Transformation (XSLT)[23]. In this case “Content handlers requiring XSLT can simply output the XML as is, and leave the transformation to Apache.”[24].

A detailed view of the request handling process is shown in Figure 5.

[23] http://en.wikipedia.org/wiki/XSL_Transformation (2010-01-24)
[24] Kew (2007) p. 48

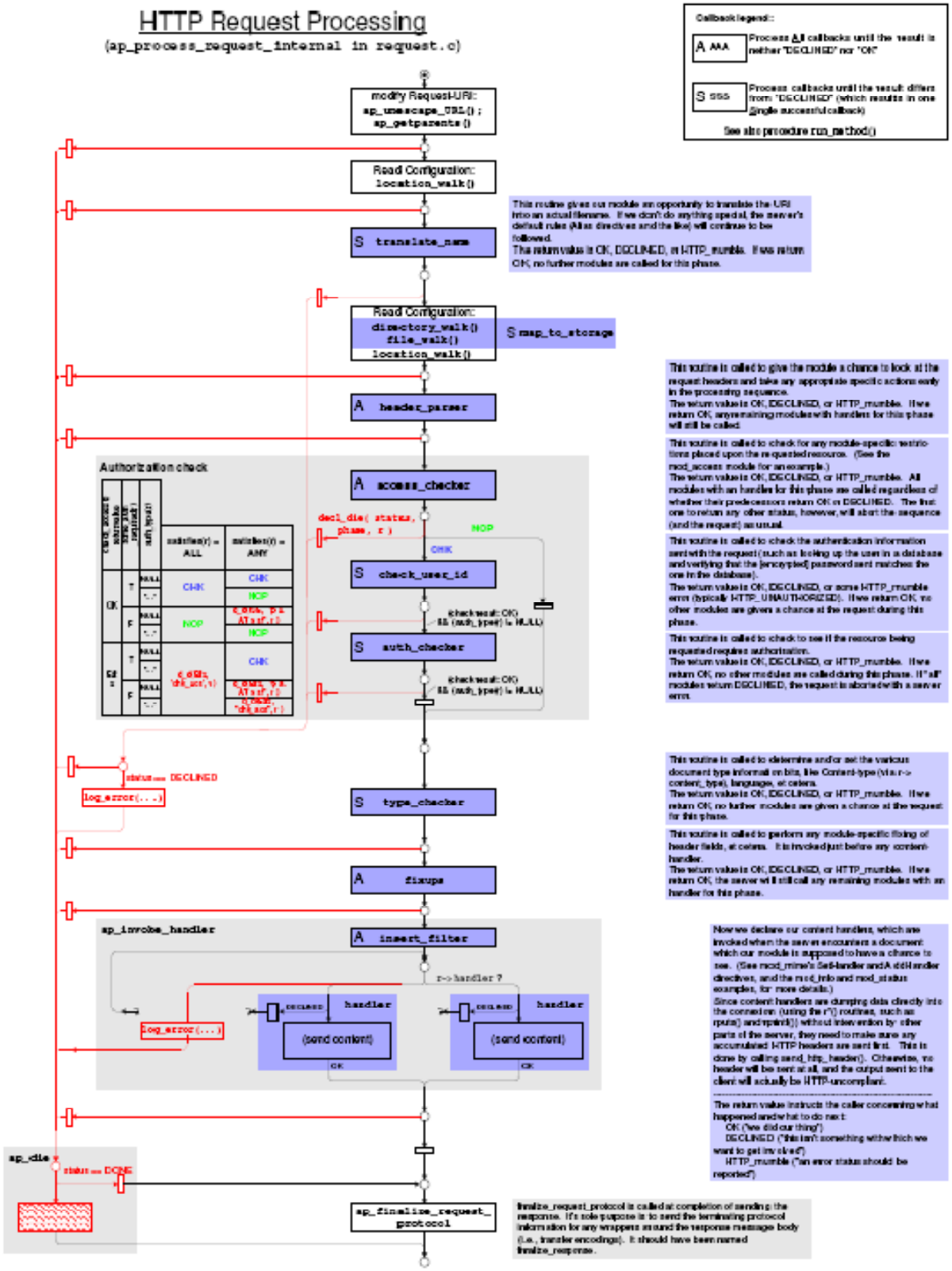

Figure 5 Apache request handling (detailed view) [25]

[25] Gröne et.al (2004), p. 83

2.4.2 Hooks for requests

The interaction between modules and the webserver is done using hooks.
A hook is a point at which a module can request to be called. Each hook specifies a function prototype, and each module can specify functions that get called at the appropriate moment. “When the moment arrives, the provider of the hook calls all the functions in order. It may terminate when particular values are returned — the hook functions can return either "declined" or "ok" or an error. In the first case all functions are called until an error is returned (if one occurs, of course); in the second case functions are called until either an error or "ok" is returned.”[26].

Enclosed a list of standard hooks for request processing [27]

- post_read_request: This is the first hook available to modules in normal request processing.
- translate_name: Apache maps the request Uniform Resource Locator (URL) to the filesystem. A module can insert a hook here to substitute its own logic.
- header_parser: This hook inspects the request headers. It is rarely used, as modules can perform that task at any point in the request processing, and they usually do so within the context of another hook.
- access_checker: Apache checks whether access to the requested resource is permitted according to the server configuration (httpd.conf).
- check_user_id: If any authentication method is in use Apache will apply the relevant authentication and set the username field. A module may implement an authentication method with this hook.
- auth_checker: This hook checks whether the requested operation is permitted to the authenticated user.
- type_checker: This hook applies rules related to the MIME type (where applicable) of the requested resource, and determines the content handler to use (if not already set).
- Fixups: This general-purpose hook enables modules to run any necessary processing after the preceding hooks but before the content generator. Like

[26] Laurie, Laurie (2003) p. 418
[27] Kew (2007) p. 44

the post_read_request it is something of a catch-all, and is one of the most commonly used hooks.

- Handler: This is the content generator hook. It is responsible for sending an appropriate response to the client. If there are input data, the handler is also responsible for reading them. Unlike the other hooks, where zero or many functions may be involved in processing a request, every request is processed by exactly one handler.
- log_transaction: This hook logs the transaction after the response has been returned to the client. A module may modify or replace Apache's standard logging.

3 Mod_ooRexx

3.1 Introduction

One of the great things of the Apache webserver is its modularity. If it does not do what you want or is not capable to do so you can change it by adding a module.

This also happened to include the Open source scripting language REXX into the Apache webserver by using the module mod_rexx.

A brief history: In the mid 1990s IBM offered a product called Object REXX.
In 2004 Object REXX was transferred from IBM to the Rexx Language Association (RexxLA) and became an open-source project named Open Object REXX or ooREXX[28].

REXX and ooREXX are compatible. This means that REXX scripts work with ooREXX (normally) without changes. Just to make sure: The other way round is only possible as long as you don't use object orientation.
This helps protecting investment in old traditional REXX code whereas new developments can be done using the new object orient style and features.

The source code of mod_REXX has been rewritten to improve the performance and to support the latest version (4.0) of ooREXX. This module will only support ooREXX version 4.0 and later. To show this clearly to all of us the module has been renamed to mod_ooRexx.

Mod_ooRexx is a Dynamic Shared Object (DSO) for the Apache webserver. "These files usually stay inside a program-specific directory and there is no automatically established link to the executable program where they are used. Instead the executable program manually loads the DSO at run-time."[29].

[28] Fosdick (2005) p. 496
[29] http://httpd.apache.org/docs/2.0/dso.html (2010-01-24)

3.2 Installation of mod_ooRexx on Windows Server 2008

ATTENTION: There is a bug in the Windows thread part of ooRexx 4.0 which causes mod_ooRexx to crash.

So at the moment it is not possible to run mod_ooRexx on Windows.

3.2.1 Prerequisites

Before installing Mod_ooRexx you have to complete the following steps

- Windows Server 2008 installation (see Appendix B: Setup Virtual machines: Window 2008 Server on page 47 for more details)
- Apache Webserver installation and customization (see Appendix D: Installing Apache on Windows 2008 Server on page 73 for more details
- ooRexx installation (see Appendix F: Installing ooRexx on Windows Server 2008 on page 91 for more details)

If you are not using a binary distribution you will also need

- The “wget for Windows” Utility installed on your system[30]
- Microsoft Visual C++ 2008 installed on the system for compilation of mod_ooRexx

3.2.2 Installation from source code

We will use the latest version of the software for the seminarpaper which can be downloaded from the incubator directory of the sourceforge project site.

Open a DOSbox and type the following command:

```
E:\>"C:\Program Files\GnuWin32\bin\wget.exe" --no-check-certificate -r
-erobots=off -w 2 --limit-rate=200
https://oorexx.svn.sourceforge.net/svnroot/oorexx/incubator/mod_oorexx
```

This downloads the content of the incubator website to the directory oorexx.svn.sourceforge/svnroot/oorexx/incubator/mod_oorexx.

[30] http://gnuwin32.sourceforge.net/packages/wget.htm (2010-01-24)

For a better handling move the created directory to a directory called mod_ooRexx using the explorer. Open a DOSbox and switch to the directory

```
E:\# cd mod_oorexx
```

Download the Apache Source distribution for Windows (httpd-2.2.14-win32-src.zip) and unpack it to the mod_oorexx directory. You will get a subdirectory httpd-2.2.14.

For compiling Apache you will also need a utility called AWK[31] copied to the binary directory of Microsoft Visual Studio. Usually this is C:\Program Files\Microsoft Visual Studio 9.0\VC\bin.

Open the “Visual Studio 2008 Command Prompt” from Start / Programs / Microsoft Visual C++ 2008 Express Edition / Visual Studio Tools.

In the command prompt issue the following commands:

```
C:\Programme\Microsoft Visual Studio 9.0\VC>E:
E:\>
E:\>cd mod_oorexx\http-2.2.14
E:\mod_oorexx>nmake -f Makefile.win _apacher
```

To prepare the final compilation process of mod_ooRexx we have to copy some files to the expected destinations. To do this issue the commands shown below:

```
E:\mod_oorexx>mkdir httpd-2.2.14\Release\include
E:\mod_oorexx>copy httpd-2.2.14\include\*.* httpd-
2.2.14\Release\include\*.*
E:\mod_oorexx>copy httpd-2.2.14\srclib\apr\include\*.* httpd-
2.2.14\Release\include
E:\mod_oorexx>copy httpd-2.2.14\srclib\apr-util\include\*.* httpd-
2.2.14\Release\include
E:\mod_oorexx>mkdir httpd-2.2.14\Release\lib
E:\mod_oorexx>copy httpd-2.2.14\srclib\apr\Release\libapr-1.lib httpd-
2.2.14\Release\lib\
E:\mod_oorexx>copy httpd-2.2.14\Release\*.lib httpd-2.2.14\Release\lib\
```

[31] http://cm.bell-labs.com/cm/cs/who/bwk/awk95.exe (2010-01-24)

To start the compilation of mod_ooRexx we need to adjust the makefile to our needs. To do so open the makefile.nt located in the mod_oorexx directory and change the following values:

```
AP_PATH = "c:\Program Files\Apache Software Foundation\Apache2.2"
```

to

```
AP_PATH = "E:\mod_oorexx\httpd-2.2.14"
```

Depending on your ooREXX installation path you might also need to change the RX_PATH according to your needs.

After all this steps it is time for the final compilation:

```
E:\mod_oorexx>rexx make_mod_oorexx.rex
```

Next we need to copy the binary file to the needed locations.

```
E:\mod_oorexx> copy bin\mod_oorexx.dll “C:\Program Files\Apache Software Foundation\Apache2.2\modules”
```

3.2.3 Customization

3.2.3.1 Copy sample pages

The distribution comes with some samples which can be used for testing the system. For this reason we will copy them to the appropriate locations.

```
E:\mod_oorexx> copy rspcomp\rspcomp.rex “C:\Program Files\Apache Software Foundation\Apache2.2\bin”
E:\mod_oorexx> copy rexscripts\Apache.cls “C:\Program Files\Apache Software Foundation\Apache2.2”
E:\mod_oorexx> copy rexscripts\*.rex E:\htdocs\mydomain.com\pages
E:\mod_oorexx> copy rspscripts\*.rex E:\htdocs\mydomain.com\pages
```

3.2.3.2 Modify httpd.conf file

Last step is to edit the main configuration file httpd.conf located at C:\Program Files\Apache Software Foundation\Apache2.2\conf

Add the following line to the end of the appropriate httpd.conf LoadModule list:

```
LoadModule oorexx_module modules/mod_oorexx.dll
```

The following lines should be added at the end of the http.conf file

```
AddType application/x-httpd-rexx-script .rex .rexx
AddType application/x-httpd-rexx-rsp .rsp
#Add these for ooRexx Server Page support
RexxTempFileNameTemplate "c:/temp/execrsp?????.rex"
RexxRspCompiler "c:/Program Files/Apache Group/Apache2/bin/rspcomp.rex"
```

Restart the webserver to reread the configuration file.

3.2.4 Check the installation

When accessing the test.rex script which comes with the distribution from a remote webbrowser you should see the following page but as described at the beginning of this chapter mod_ooRexx crashes due to a bug.

3.2.5 Installation from a binary distribution

At the moment there is no binary distribution available.

3.3 Installation of Mod_ooRexx on Fedora 12

3.3.1 Prerequisites

Before installing Mod_ooRexx complete the following steps

- Fedora 12 installation (see Appendix C: Setup Virtual machines: Fedora 12 on page 58 for more details)
- Apache Webserver installation and customization (see Appendix E: Installing Apache on Fedora 12 on page 82 for more details
- ooRexx installed (see Appendix G: Installing ooRexx on Fedora 12 on page 98 for more details)

3.3.2 Installation

Like the windows version we will use the latest version of the software and download it from the incubator directory of the sourceforge project site.

```
[mod_rexx@www ~]$ su
Password:  *****
[root@www mod_rexx]# wget -r -erobots=off –w 2 --limit-rate=20
https://oorexx.svn.sourceforge.net/svnroot/oorexx/incubator/mod_oorexx
```

This downloads the content of the incubator website to the directory oorexx.svn.sourceforge/svnroot/oorexx/incubator/mod_oorexx.

For a better handling move the created directory to a directory called mod_oorexx and switch to this directory

```
[root@www mod_rexx]# mv
      oorexx.svn.sourceforge/svnroot/oorexx/incubator/mod_oorexx mod_oorexx
[root@www mod_rexx]# cd mod_oorexx
```

Prior to compilation and installation we need to adjust the makefile.linux according to our needs. Open the file in your favorite editor and change

```
INCLUDES = -I/usr/include/httpd –I/usr/include/apr-l
```

to

```
INCLUDES = -I/usr/local/apache2/include
```

Save the file.

Create the binaries by issuing the following command.

```
root@www mod_oorexx]# rexx make_mod_oorexx.rex
```

Next we need to copy the files to the needed locations.

```
[root@www mod_oorexx]# cp bin/mod_oorexx.so /usr/local/apache2/modules
[root@www mod_oorexx]# cp rexx.conf /usr/local/apache2/conf/
[root@www mod_oorexx]# cp rspcomp/rspcomp.rex /usr/local/apache2/bin
[root@www mod_oorexx]# cp rexscripts/Apache.cls /usr/bin
```

3.3.3 Customization

3.3.3.1 Copy sample pages

The distribution comes with some samples which can be used for testing the system. For this reason we will copy them to the appropriate locations.

```
[root@www mod_oorexx]# mkdir /var/htdocs/mydomain.com/scripts
[root@www mod_oorexx]# cp rspscripts/*.rsp
      /var/htdocs/mydomain.com/scripts/
[root@www mod_oorexx]# cp rexscripts/test.rex
      /var/htdocs/mydomain.com/pages/
[root@www mod_oorexx]# cp rexscripts/otest1.rex
      /var/htdocs/mydomain.com/pages/
[root@www mod_oorexx]# cp rexscripts/footer.rex
      /var/htdocs/mydomain.com/pages/
[root@www mod_oorexx]# cp rexscripts/access.rex
      /var/htdocs/mydomain.com/pages/
```

3.3.3.2 Modify rexx.conf file

For including RSP (Rexx Server Pages) we need to adjust the compiler parameter in the configuration file rexx.conf.

To do so move to the Apache server configuration directory by issuing the following command.

```
[root@www mod_oorexx]# cd /usr/local/apache2/conf
```

Open the file rexx.conf in your favorite editor and change the following lines in the section Rexx Server Page Support:

```
RexxRspCompiler "/usr/bin/rspcomp.rex"
```

to

```
RexxRspCompiler "/usr/local/apache2/bin/rspcomp.rex"
```

3.3.3.3 Include RSP support in the Apache main configuration file

Open the file httpd.conf in the configuration directory of the Apache webserver (/usr/local/apache2/conf/) and append the following lines:

```
# Mod_ooRexx support
Include conf/rexx.conf
```

3.3.3.4 Modifying selinux configuration

Fedora 12 comes with the Security-Enhanced Linux enabled by default.

“Security-Enhanced Linux (SELinux) is a Linux feature that provides a mechanism for supporting access control security policies, including U.S. Department of Defense style mandatory access controls, through the use of Linux Security Modules (LSM) in the Linux kernel.”[32]

We need to change the SELinux security context of the mod_oorexx.so file. Otherwise Apache won’t be able to start with error message similar to the one shown below:

```
[root@www bin]# ./apachectl start
httpd: Syntax error on line 414 of /usr/local/apache2/conf/httpd.conf:
Syntax error on line 16 of /usr/local/apache2/conf/rexx.conf: Cannot load
/usr/local/apache2/modules/mod_oorexx.so into server:
/usr/local/apache2/modules/mod_oorexx.so: cannot restore segment prot after
reloc: Permission denied
```

To do so issue the following command:

```
[root@www bin]# chcon -t textrel_shlib_t
      '/usr/local/apache2/modules/mod_oorexx.so'
```

3.3.4 Check the installation

When accessing the test.rex script included in the distribution from remote ebbrowser you should see the following page.

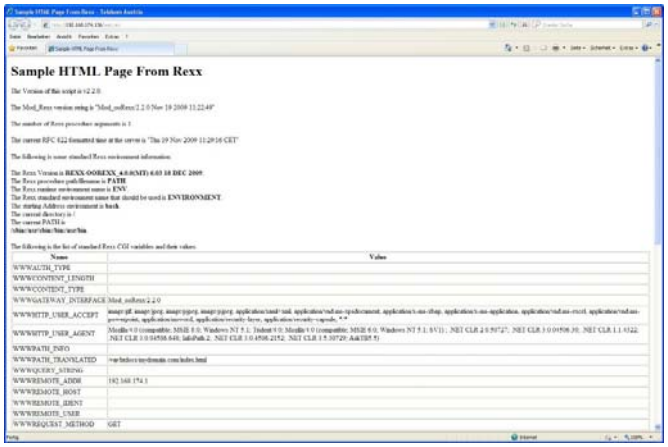

Figure 6 Mod_ooRexx Installation – Finished installation

[32] http://en.wikipedia.org/wiki/Selinux (2010-01-24)

4 Examples

4.1 Introduction

The following section should show you some example of scripts using ooREXX and mod_ooREXX which have been developed by “playing” around during the creation of this seminar paper.

Some further description is included as well.

4.2 Hello World – A different way

As usual when starting to learn a programming language the first program developed is the “Hello World” example. In our case I have used a little bit more sophisticated version - drawing an equilateral triangle with sizes based on user input.

4.2.1 Sample 01: The “quick and dirty” version

For the first example we just add the code to the script.

```
1:  /*******************************/
2:  /* Draw a equilateral triangle */
3:  /* (c)2010 Robert Maschek      */
4:  /*******************************/
5:
6:  /* Apache return codes used */
7:  OK = 0     /* Module has handled this stage. */
8:
9:  /* get the Apache request record pointer */
10: r = arg(1)
11:
12: /* set content-type and send the HTTP header */
13: CALL WWWSendHTTPHeader r, "text/html"
14:
15: /* start sending the html page */
16: SAY '<HTML>'
17: SAY '<HEAD>'
18: SAY '<TITLE>My first HTML Page From Rexx</TITLE>'
19: SAY '</HEAD>'
20: SAY '<BODY>'
21:     SAY '<H1><FONT SIZE="5" FACE="Arial" COLOR="RED"><U>My first HTML Page From
        Rexx</U></FONT><BR>'
22:     SAY '<FONT SIZE="3" FACE="Arial">Let us draw a equilateral triangle</FONT></H1>'
23:
24: SAY '<FORM method="post" action="./Sample01.rex">'
25: SAY '<P>Enter number of characters: '
26:     SAY '<INPUT type="text" name="visibletext" size="5">'
27: SAY '<BR>'
28: SAY '<INPUT type="submit" value="Start to draw">'
29: SAY '</FORM>'
30:
31: CALL WWWGetArgs r
32: IF wwwargs.0 > 0 THEN DO
33:     numberChar=wwwargs.1.!value
34:     SAY '<P>Number of characters: 'numberChar'</P>'
35:
```

```
36:     /* We need an uneven number to draw a triangle */
37:     IF (numberChar//2) == 0 THEN numberChar = numberChar + 1
38:
39:     SAY '<FONT SIZE=3 FACE=courier COLOR=blue>'
40:     SAY '<B>'
41:
42:     /* Draw hypotenuse */
43:     DO numberChar
44:             SAY '*'
45:     END
46:     SAY '<BR>'      /* adds the carriage return at EOL*/
47:
48:     /* Draws the triangle */
49:     DO a = 1 TO numberChar/2
50:     /* left side */
51:             DO b = 1 TO a
52:                     SAY ' '
53:             END
54:             SAY '*'
55:             /* right side */
56:             DO c = 1 TO numberChar - (2 * b)
57:                     SAY ' '
58:             END
59:             IF c <> 1 THEN        /* This is needed to draw the tip */
60:             DO
61:                     SAY '*'
62:                     SAY '<BR>'
63:             END
64:     END
65:     SAY '</FONT> </B>'
66: END
67: SAY '</BODY>'
68: SAY '<BR><BR>'
69: SAY '<FONT SIZE="1">Running on: 'wwwserver_software'</FONT>'
70: SAY '</HTML>'
71:
72: RETURN OK
```

Figure 7 Sample 01: equilateral triangle – the simplest way – source code (Sample01.rex)

Detailed description:

At the beginning of the script (line 6 and 7) there is the definition of the return codes when the script ends. In our case we just need the one even there are a few more available. The webserver uses these returncodes to continue with its operation. For example it is used to distinguish between different error messages being displayed on the browser (like 404 → not found).

The next two steps are taking the Apache request record pointer (line 9) and sending back the http header (line 12) to the browser. Please keep in mind that the last call has to be in the script.

The other Call statement that has to be in your script is shown in line 32. Using this function we return the GET/POST request arguments.

The script is actually processed twice:

- During the first run we just show the first part of the homepage. Since there is no number (argument) entered wwwargs.0 is also 0. This changes once we entered a number and the [Start to Draw] button is pressed.
- After pressing the button the script is executed for the second time. Now there is a value in the wwwargs.0 (not 0) so that we can really start to draw.

The other parts of the source code are pretty simple.
The start is to check whether an uneven number was entered (line 38). Since we use characters for the drawing we can't split one of them. That is the reason for the need of an uneven number. If an even number was entered we just add one to it.

Line 40 and 41 just set the font and the color for the triangle.
Between line 49 and 65 we are actually drawing the triangle using some loops. The only piece to mention is that we need to use a non breaking space ( ) to get to the needed points. Otherwise the leading spaces will be removed automatically.

Finally we add a footer line (line 70) which displays some information about the server.

The result is shown below.

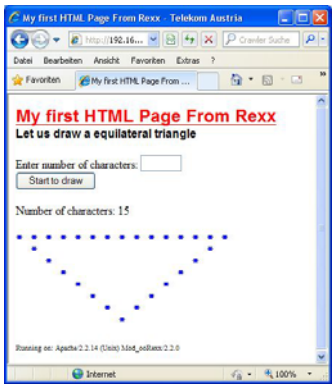

Figure 8 Sample 01: equilateral triangle – the simplest way – the result

4.2.2 Sample 02: Using routines and procedures

As a first improvement we structure the sourcecode using procedures, functions and routines.

The main difference between procedures and functions in ooREXX is that “functions can return a value ("functionvalue") to the caller via the RETURN-statement”[33] whereas a procedure is just a “grouping of statements which repeatedly get executed by different parts in a program”[34].

Routines are handled like external procedures/functions. This means a routine is more or less like running a program in the program.

They represent procedures and functions and after a successful syntax check they are made available in the scope of the program itself, and in addition in all superordinate (calling) programs, if the keyword PUBLIC is given[35].

```
1:  /*******************************/
2:  /* Draw a equilateral triangle  */
3:  /* with Procedures and Routines */
4:  /* (c)2010 Robert Maschek       */
5:  /*******************************/
6:
7:  /* Apache return codes used */
8:  OK = 0     /* Module has handled this stage. */
9:
10: /* get the Apache request record ptr */
11: r = arg(1)
12:
13: /* set content-type and send the HTTP header */
14: CALL WWWSendHTTPHeader r, "text/html"
15:
16: /* start sending the html page */
17: '<HTML>'
18: SAY '<HEAD>'
19: SAY '<TITLE>My second HTML Page From Rexx</TITLE>'
20: SAY '</HEAD>'
21: SAY '<BODY>'
22:     SAY '<H1><FONT SIZE="5" FACE="Arial" COLOR="RED"><U>My second HTML Page From
        Rexx</U></FONT><BR>'
23:     SAY '<FONT SIZE="3" FACE="Arial">Let us draw a equilateral triangle<BR>'
24: SAY '(Now we add routines and procedures)</FONT></H1>'
25:
26: SAY '<FORM method="post" action="./Sample02.rex">'
27: SAY '<P>Enter number of characters: '
28: SAY '<INPUT type="text" name="visibletext" size="5">'
29: SAY '<BR>'
30: SAY '<INPUT type="submit" value="Start to draw">'
31: SAY '</FORM>'
32:
33: CALL WWWGetArgs r
34: IF wwwargs.0 > 0 THEN DO
35:
36:     numberChar=wwwargs.1.!value
37:     SAY '<P>Number of characters: 'NumberChar'</P>'
38:
39:     /* We need an uneven number to draw a triangle */
```

[33] Flatscher (2009a) p. 5
[34] Flatscher (2009a) p. 3
[35] Flatscher (2009b) p. 16

```
40:     IF (numberChar//2) == 0 THEN numberChar = numberChar + 1
41:
42:     SAY '<FONT SIZE=3 FACE=courier COLOR=blue>'
43:     SAY '<B>'
44:
45:     /* Draw hypotenuse */
46:     CALL DRAW_HYPOTENUSE
47:
48:     /* Draws the triangle */
49:     DO a = 1 TO numberChar/2
50:             CALL DRAW_LEFT_SIDE
51:             /* right side */
52:             CALL DRAW_RIGHT_SIDE
53:             IF c <> 1 THEN CALL DRAW_TIP  /* needed to draw the tip */
54:     END
55:     SAY '</FONT> </B>'
56: END
57: SAY '</BODY>'
58: SAY '<BR><BR>'
59: SAY '<FONT SIZE="1">Running on: 'wwwserver_software'</FONT>'
60: SAY '</HTML>'
61: RETURN OK
62: /************************/
63: /* Procedures/Functions */
64: /************************/
65: DRAW_HYPOTENUSE:
66: DO numberChar
67:     SAY '*'
68: END
69: SAY '<BR>' /* adds the carriage return at the end of the line */
70: RETURN
71:
72: DRAW_LEFT_SIDE:
73: DO b = 1 TO a
74:     SAY ' '
75: END
76: SAY '*'
77: RETURN
78:
79: DRAW_RIGHT_SIDE:
80: DO c = 1 TO numberChar - (2 * b)
81:     SAY ' '
82: END
83: RETURN
84:
85: /************/
86: /* Routines */
87: /************/
88: :: ROUTINE DRAW_TIP PUBLIC
89: SAY '*'
90: SAY '<BR>'
91:
92: EXIT 0
```

Figure 9 Sample 02: equilateral triangle – using routines and procedures – source code (Sample02.rex)

Detailed description:

The behavior of the program is the same as described in section 4.2.1 on page 23.

We have taken out often used parts from the sourcecode and placed them into the procedures and routines after the final Return statement in line 62.

They are used with the Call statement (line 47, 51 and 53).

In our case there is also no return value from the procedures.

The result is shown below.

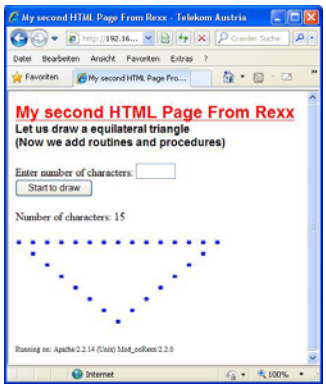

Figure 10 Sample 02: equilateral triangle – using routines and procedures – the result

4.2.3 Sample 03: Using Rexx Server Pages (RSP), Cascading Style Sheets (CSS) and Frames

Rexx Server Pages are dynamic webpages using server side scripting similar to Java Server Pages (JSP), Active Server Pages (ASP) or PHP for example. For this we place normal REXX statements in the HTML code. These statements are executed on the webserver at run time and the result is delivered to the client.

```
1:  <HTML>
2:  <HEAD>
3:      <TITLE>My third HTML Page from REXX </TITLE>
4:  </HEAD>
5:  <FRAMESET ROWS="20%,*">
6:      <FRAME Name="Top" SRC="Sample03_Top.html">
7:      <FRAME Name="Bottom" SRC="Sample03_Bottom.rsp">
8:  </FRAMESET>
9:  </HTML>
```

Figure 11 Sample 03: equilateral triangle – using RSP and CSS – source code (Sample03.html)

This is the HTML code for the main page. In our case the main page just provides the “Frame” for the frames.

```
1:  h1 {        color: indigo;
2:      text-align:left;
3:      font-family: Arial,sans-serif;
4:      font-size: 12pt;
5:  }
6:  body {      background-color: lightblue;
7:      font-family: Arial,sans-serif;
8:      font-size: 12pt;
9:  }
10:  p.blue { color: blue;
11:     font-family: Courier;
12:     font-weight: bold;
13:  }
```

```
14:  p.footer { color: black;
15:     text-align:left;
16:     font-family: Arial,sans-serif;
17:     font-size: 8pt;
18:  }
```

Figure 12 Sample 03: equilateral triangle – using RSP, CSS and frames – source code (Samples.css)

A CSS is a style sheet used to describe the presentation semantics (that is, the look and formatting) of a document written in a markup language[36]. The advantage of CSS is that webpages stay consistent over the entire website and the design can be changed very easy.

```
1:   <HTML>
2:   <HEAD>
3:      <TITLE>My third HTML Page from REXX </TITLE>
4:      <LINK REL="stylesheet" TYPE="text/css" HREF="Samples.css">
5:   </HEAD>
1:
2:   <BODY>
3:      <H1><U>My third HTML Page From Rexx</U></H1>
4:      Let us draw a equilateral triangle.<BR>
5:      (Now we add Cascading Style sheets and Frames. So we have to use RSP.)<BR>
6:      <FORM method="post" action="Sample03_Bottom.rsp" target="Bottom">
7:             Enter number of characters:
8:             <INPUT type="text" name="visibletext" size="5">
9:             <BR>
10:     <INPUT type="submit" value="Start to draw">
11:     </FORM>
12:  </BODY>
13:  </HTML>
```

Figure 13 Sample 03: equilateral triangle – using RSP, CSS and frames – source code (Sample03_Top.html)

Detailed description:

The Top frame states only plain HTML covering the input of the amount of characters. After pressing the [Start to draw] button (line 10) the Sample03_Bottom.rsp page is called from the webserver and displayed in the Bottom frame according to the FORM instruction in line 6.

```
1:  <HTML>
2:  <HEAD>
3:     <LINK REL="stylesheet" TYPE="text/css" HREF="Samples.css">
4:  </HEAD>
5:
6:  <BODY>
7:  <SCRIPT type="rexx">
8:     /* Apache return codes used*/
9:     OK = 0     /* Module has handled this stage. */
10:
11:    IF wwwargs.0 > 0 THEN DO
12:           numberChar=wwwargs.2.!value
13:           SAY 'Number of characters: 'NumberChar'<BR><BR>'
14:
15:           /* We need an uneven number to draw a triangle */
16:           IF (numberChar//2) == 0 THEN numberChar = numberChar + 1
17:
18:           SAY '<P CLASS="blue">'
```

[36] http://en.wikipedia.org/wiki/Cascading_Style_Sheets (2010-01-24)

```
19:             /* Draw hypotenuse */
20:             CALL DRAW_HYPOTENUSE
21:             /* Draws the triangle */
22:             DO a = 1 TO numberChar/2
23:                     CALL DRAW_LEFT_SIDE
24:                     /* right side */
25:                     CALL DRAW_RIGHT_SIDE
26:                     IF c <> 1 THEN CALL DRAW_TIP  /*Needed to draw tip*/
27:                     END
28:             SAY '</P>'
29:     END
30:
31:     SAY '<P CLASS="footer">Running on: 'wwwserver_software'</P>'
32:     RETURN OK
33:
34:     /************************/
35:     /* Procedures/Functions */
36:     /************************/
37:     DRAW_HYPOTENUSE:
38:     DO numberChar
39:             SAY '*'
40:     END
41:     SAY '<BR>'      /* adds the carriage return at the eol */
42:     RETURN
43:
44:     DRAW_LEFT_SIDE:
45:     DO b = 1 TO a
46:             SAY ' '
47:     END
48:     SAY '*'
49:     RETURN
50:
51:     DRAW_RIGHT_SIDE:
52:     DO c = 1 TO numberChar - (2 * b)
53:             SAY ' '
54:     END
55:     RETURN
56:
57:     /************/
58:     /* Routines */
59:     /************/
60:     :: ROUTINE DRAW_TIP PUBLIC
61:     SAY '*'
62:     SAY '<BR>'
63:
64:     EXIT 0
65: </SCRIPT>
66: </HTML>
```

Figure 14 Sample 03: equilateral triangle – using RSP, CSS and frames – source code (Sample03_Bottom.rsp)

Detailed description:

Now we are calling a page from the webserver ending with .rsp. From the directive “AddType application/x-httpd-rexx-rsp .rsp” placed in the rexx.conf file the server knows that this page includes server side scripting and needs to be executed in a proper way.

When such a request is made it is processed in four stages by Mod_ooRexx and a ooRexx program/script (the RSP compiler. See the directive RexxRspCompiler in the rexx.conf file).

1. Mod_ooRexx creates a temporary file for the compiled version of the RSP file.
2. RSPCOMP.REX (the RSP compiler) is called to compile the RSP file into a real ooRexx program and place it in the temporary file.
3. Mod_ooRexx calls the newly created ooRexx program.
4. Mod_ooRexx removes the temporary file.

The execution of the sourcecode is similar to Sample 02: Using routines and procedures on page 26.

The result is shown below.

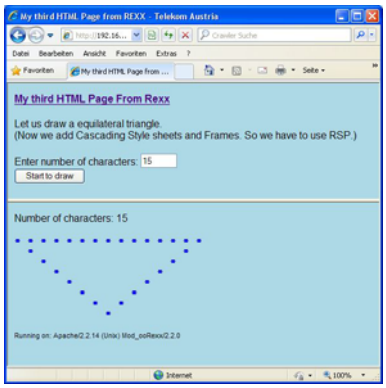

Figure 15 Sample 03: equilateral triangle – using RSP and CSS – the result

4.2.4 Sample 04: Using Object Oriented Programming

Since “Open Object Rexx is a fully object-oriented superset of standard Rexx”[37] why not to use it?

“Open Object Rexx is a powerful, object-oriented language that retains the ease of use of classic Rexx. The product lies at the intersection of three of today’s key software trends:

- High-level scripting
- Object-oriented programming
- Open-source software”[38]

For this reason we have updated the program to use OO technology.

[37] Fosdick (2005) p. 459
[38] Fosdick (2005) p. 473

```
1:  /*********************************/
2:  /* Draw a equilateral triangle  */
3:  /* with Procedures and Routines */
4:  /* (c)2010 Robert Maschek       */
5:  /*********************************/
6:
7:  /* Apache return codes used */
8:  OK = 0     /* Module has handled this stage. */
9:
10: /* get the Apache request record ptr */
11: r = arg(1)
12:
13: /* set content-type and send the HTTP header */
14: CALL WWWSendHTTPHeader r, "text/html"
15:
16: /* start sending the html page */
17: SAY '<HTML>'
18: SAY '<HEAD>'
19: SAY '<TITLE>My Fourth HTML Page From Rexx</TITLE>'
20: SAY '</HEAD>'
21: SAY '<BODY>'
22:
23:     SAY '<H1><FONT SIZE="5" FACE="Arial" COLOR="RED"><U>My fourth HTML Page From
        Rexx</U></FONT><BR>'
24:     SAY '<FONT SIZE="3" FACE="Arial">Let us draw a equilateral triangle<BR>'
25: SAY '(Now we add Object Oriented Programming)</FONT></H1>'
26:
27: SAY '<FORM method="post" action="./Sample04.rex">'
28: SAY '<P>Enter number of characters: '
29: SAY '<INPUT type="text" name="visibletext" size="5">'
30: SAY '<BR>'
31: SAY '<INPUT type="submit" value="Start to draw">'
32: SAY '</FORM>'
33:
34: CALL WWWGetArgs r
35: IF wwwargs.0 > 0 THEN DO
36:
37:     numberChar=wwwargs.1.!value
38:     SAY '<P>Number of characters: 'NumberChar'</P>'
39:
40:     /* We need an uneven number to draw a triangle */
41:     IF (numberChar//2) == 0 THEN numberChar = numberChar + 1
42:
43:     SAY '<FONT SIZE=3 FACE=courier COLOR=blue>'
44:     SAY '<B>'
45:
46:     /* Instantiate new object */
47:     triangle1 = .triangle~New(numberChar)
48:
49:     /* Draw hypotenuse */
50:     triangle1~~draw_hypotenuse()
51:
52:     /* Draws the triangle */
53:     triangle1~~draw_triangle()
54:     SAY '</font> </b>'
55: END
56: SAY '</BODY>'
57: SAY '<BR><BR>'
58: SAY '<FONT SIZE="1">Running on: 'wwwserver_software'</FONT>'
59: SAY '</HTML>'
60:
61: RETURN OK
62:
63: /***********/
64: /* Classes */
65: /***********/
66: ::CLASS triangle
67:
68: ::METHOD init
69:     EXPOSE numberChar
70:     USE ARG numberChar
71:
72: ::METHOD numberChar ATTRIBUTE
73:
```

```
74: /*******************/
75: /* Start of methods */
76: /*******************/
77: ::METHOD draw_hypotenuse
78: /* Draws the hypotenuse of the triangle */
79:     DO self~numberChar
80:             SAY '*'
81:     END
82:     SAY '<BR>'      /* adds the carriage return at eol */
83:
84: ::METHOD draw_triangle
85: /* Draws the other two sides of the triangle */
86:     DO a = 1 TO self~numberChar/2
87:             /* left side */
88:             DO b = 1 TO a
89:                     SAY ' '
90:             END
91:             SAY '*'
92:             /* right side */
93:             DO c = 1 TO self~numberChar - (2 * b)
94:                     SAY ' '
95:             END
96:             IF c <> 1 THEN                /* Needed to draw the tip */
97:             DO
98:                     SAY '*'
99:                     SAY '<BR>'
100:                    END
101:            END
```

Figure 16 Sample 04: equilateral triangle – using Objects – source code (Sample04.rex)

Detailed description:

Up to line 46 the sourcecode is similar to the examples shown before.

Line 47 instantiates a new object of the class “triangle” and in line 50 and 53 we call the methods of the class.

The class consists of two methods:

- draw_hypotenuse: Draws the hypotenuse of the triangle.
- draw_triangle: Draws the other two sides of the triangle.

The result is shown below.

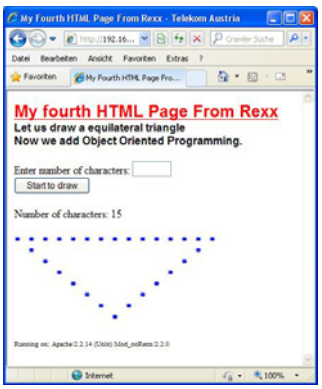

Figure 17 Sample 04: equilateral triangle – using Objects – the result

4.3 Using ooREXX to control access to the server

The usage ooREXX for Apache is not limited to create webpages. It is also possible to use it for controlling or changing the behavior of the Apache webserver.

4.3.1 Sample 05: Our standard for the Browser is Internet Explorer 8

One of the common problems within a corporate infrastructure is the enforcement of standards. In our case we want to deny the access to the webserver for all other browsers than our standard.

The sourcecode shown in Figure 18 is able to do this.

```
1:      /*****************************************/
2:      /* Check Browser for MSIE and Version 8.0 */
3:      /* (c)2010 Robert Maschek                 */
4:      /*****************************************/
5:      /* Apache return codes used */
6:      OK        = 0     /* Module has handled this stage. */
7:      FORBIDDEN = 403         /* Display an Error 403 webpage */
8:
9:      /* get the Apache request record ptr */
10: r = arg(1)
11:
12: /* Extract the needed information */
13:     PARSE VAR wwwhttp_user_agent part_1 "MSIE " part_2
14:     PARSE VAR part_2 before ";" after
15:
16:     IF before = 8.0 then DO
17:     /* MSIE 8.0 used */
18:             CALL WWWLogInfo r, "Check Browser: IE and version checked o.k.
     (requested:"wwwscript_name")"
19:     END
20:     ELSE DO
21:     /* other browser used */
22:             CALL WWWLogWarning r, "Check Browser: Wrong browser or version
     (requested:"wwwscript_name")"
23:     RETURN FORBIDDEN
24:     END
25:
26:     RETURN OK
```

Figure 18 Sample 05: Check Browser type and version (check_browser_and_version.rex)

Detailed description:

The program takes the Apache request record pointer (line 10) and extracts the needed information from the variable wwwhttp_user_agent. If we are using the corporate standard the content of wwwhttp_user_agent must be “Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0; Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1) ; .NET CLR 2.0.50727; .NET CLR 3.0.04506.30; .NET CLR 1.1.4322; .NET CLR 3.0.04506.648; InfoPath.2; .NET CLR 3.0.4506.2152; .NET CLR 3.5.30729; AskTB5.5)”. Line 13 extracts the part on the right side from the first

occurrence of the string MSIE and the result is used in line 14 to extract the version number.

The next step is to check the values (line 16) and place either a success message (line 18) or an error message (line 22) in the Apache error logfile.

If there is an error (means using the wrong browser) we will also RETURN the value 403 (line 23) which results in displaying the 403 Error page (You don't have permission to access…) from the Apache webserver.

To arm this program we need to place a directive in the main configuration file (httpd.conf) as shown below.

```
# Directives for Sample 05
<Directory "/var/htdocs/mydomain.com/pages/sample05">
      RexxAccessHandler
      '/var/htdocs/mydomain.com/scripts/check_browser_and_version.rex'
</Directory>
```

Figure 19 Apache configuration directive for checking webbrowser and version

Once everything is up and running it will lead to the following result when requesting a webpage when using the wrong browser …

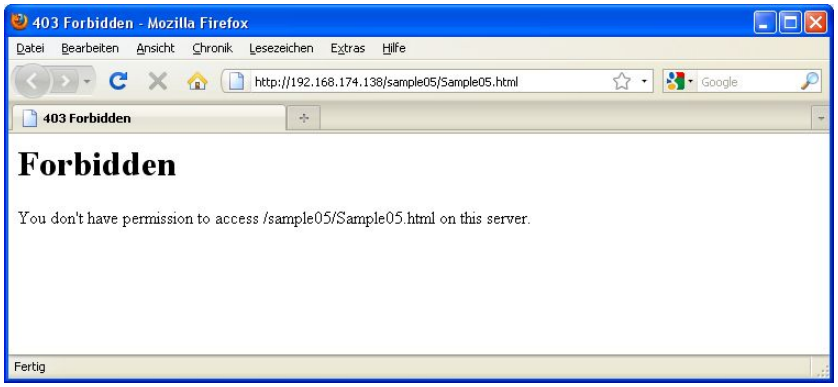

Figure 20 Check Browser and version: Wrong browser used

… and the following entry to the Apache error logfile

```
[Sat Jan 23 01:55:58 2010] [warn] [client 192.168.174.1] Check Browser:
Wrong browser or version (requested:/sample05/Sample05.html
```

When using the standard browser the following webpage will be displayed …

Figure 21 Check Browser and version: Successful tested

… and the following entries placed in the Apache error logfile

```
[Mon Jan 04 00:23:54 2010] [info] [client 192.168.174.1] Check Browser: IE8
and  version checked o.k. (requested:/sample05/Sample05.html)
[Mon Jan 04 00:23:54 2010] [info] [client 192.168.174.1] Check Browser: IE8
and  version checked o.k. (requested:/sample05/Samples.css), referer:
http://192.168.174.137/sample05/Sample05.htm
```

4.3.2 Sample 06: Access only for members of our IP-subnet

Today companies live in globalized “internal world”. So there is a good reason to protect your data. We want to do this by limiting access to the webserver to people residing in the same IP-subnet as the server is.

The following code takes the ip-address from the server, calculates subnet- and broadcast address and checks that the ip-address from the requestor is in that range.

```
1:  /*************************************************/
2:  /* Provide ACL based on IP-addresses and Subnetmask */
3:  /* (c)2010 Robert Maschek                        */
4:  /*************************************************/
5:  /* Apache return codes used */
6:      OK        = 0         /* Module has handled this stage. */
7:      FORBIDDEN = 403       /* Display an Error 403 page */
8:      INT_ERROR = 500       /* Display an Error 500 page */
9:
10: /* get the Apache request record ptr */
11: r = arg(1)
12:
13: /* Create entity */
14: ipcalculation1 = .ipcalculation~NEW()
15:
16: /* Setup net ip-address and the subnetmask which are allowed */
17: serverip="192.168.174.137"
18: subnetmask="255.255.0.0"
19:
20:     PARSE VAR serverip ip1"."ip2"."ip3"."ip4
21:     PARSE VAR subnetmask sn1"."sn2"."sn3"."sn4
22:
23: /* Calculate SubnetID */
24: ipcalculation1~~binarycalculation(ip1, sn1, 'subnetid')
25: subnetid1 = ipcalculation1~netaddrd
26: ipcalculation1~~binarycalculation(ip2, sn2, 'subnetid')
```

```
27: subnetid1 = subnetid1 || ipcalculation1~netaddrd
28: ipcalculation1~~binarycalculation(ip3, sn3, 'subnetid')
29: subnetid2 = ipcalculation1~netaddrd
30: ipcalculation1~~binarycalculation(ip4, sn4, 'subnetid')
31: subnetid2 = subnetid2 || ipcalculation1~netaddrd
32:
33: /* Calculate Broadcast Address */
34: ipcalculation1~~binarycalculation(SUBSTR(subnetid1,1,3), sn1, 'broadcastid')
35: broadcastid1=ipcalculation1~netaddrd
36: ipcalculation1~~binarycalculation(SUBSTR(subnetid1,4,3), sn2, 'broadcastid')
37: broadcastid1 = broadcastid1 || ipcalculation1~netaddrd
38: ipcalculation1~~binarycalculation(SUBSTR(subnetid2,1,3), sn3, 'broadcastid')
39: broadcastid2 = ipcalculation1~netaddrd
40: ipcalculation1~~binarycalculation(SUBSTR(subnetid2,4,3), sn4, 'broadcastid')
41: broadcastid2 = broadcastid2 || ipcalculation1~netaddrd
42:
43: /* Get requestors IP-address */
44: PARSE VAR wwwremote_addr ip1"."ip2"."ip3"."ip4
45:
46: /* Build string from ip-address */
47: ipcalculation1~~buildthreedigits(ip1)
48: requestorip1 = ipcalculation1~iprebuild
49: ipcalculation1~~buildthreedigits(ip2)
50: requestorip1 = requestorip1 || ipcalculation1~iprebuild
51: ipcalculation1~~buildthreedigits(ip3)
52: requestorip2 = ipcalculation1~iprebuild
53: ipcalculation1~~buildthreedigits(ip4)
54: requestorip2 = requestorip2 || ipcalculation1~iprebuild
55:
56: /* Check that ip is in the range */
57: IF (requestorip1 >= subnetid1) & (requestorip1 <= broadcastid) & (requestorip2 >
    subnetid2) & (requestorip2 < broadcastid2) THEN DO
58:     /* IP-Address in range */
59:     CALL WWWLogInfo r, "Check IP: Request checked and o.k. (requested:"wwwscript_name")"
60:     END
61: ELSE DO
62:     /* not in range --> unauthorized access */
63:     CALL WWWLogWarning r, "Check IP: Unauthorzied access attempt
    (requested:"wwwscript_name")"
64:     RETURN FORBIDDEN
65: END
66:
67: RETURN OK
68:
69: /***********/
70: /* Classes */
71: /***********/
72: ::CLASS ipcalculation
73:
74: ::METHOD init
75:
76: ::METHOD ip ATTRIBUTE            /* Ip-Address */
77: ::METHOD subnet ATTRIBUTE /* Subnetmask */
78: ::METHOD netaddrd ATTRIBUTE      /* Return value */
79: ::METHOD snbc ATTRIBUTE          /* Subnet(ID) or Broadcast */
80: ::METHOD iprebuild ATTRIBUTE     /* 3digit ip-address */
81:
82: /********************/
83: /* Start of methods */
84: /********************/
85: ::METHOD binarycalculation
86: /* Does all the needed binary calculations */
87:     EXPOSE ip subnet netaddrd
88:     USE ARG ip, subnet, snbc
89:
90:     /* Calculate binary values */
91:     ipb=X2B(D2X(self~ip))
92:     SELECT
93:             WHEN snbc='subnetid' THEN
94:                     subnetb=X2B(D2X(self~subnet));
95:             WHEN snbc='broadcastid' THEN
96:                     subnetb=X2B(D2X(255-self~subnet));
97:             OTHERWISE RETURN INT_ERROR
98:     END
99:
100:            netaddr=""
```

```
101:            /* We need to have 8Bit numbers */
102:            temp=""
103:            DO a = 1 TO 8 - LENGTH(ipb)
104:                    temp=temp||'0'
105:            END
106:            ipb=temp||ipb
107:
108:            temp=""
109:            DO a = 1 TO 8 - LENGTH(subnetb)
110:                    temp=temp||'0'
111:            END
112:            subnetb=temp||subnetb
113:
114:            /* Binary AND/ OR calculation of IP and SN */
115:            DO a=1 TO 8
116:                    a1 = SUBSTR(ipb,a,1)
117:                    a2 = SUBSTR(subnetb,a,1)
118:                    SELECT
119:                            WHEN snbc='subnetid' THEN
120:                            a3 = a1 & a2;
121:                            WHEN snbc='broadcastid' THEN
122:                            a3 = a1 | a2;
123:                    END
124:                    netaddr = netaddr||a3
125:            END
126:
127:            /* Calculate decimal value */
128:            netaddrd = X2D(B2X(netaddr))
129:
130:            /* We need 3digit numbers */
131:            temp=""
132:            DO a = 1 TO 3 - LENGTH(netaddrd)
133:                    temp=temp||'0'
134:            END
135:            netaddrd=temp||netaddrd
136:    EXIT 0
137:
138:    ::METHOD buildthreedigits
139:    /* Creates strings which are 3 digits long */
140:            EXPOSE iprebuild
141:            USE ARG ip
142:
143:            temp=""
144:            DO a = 1 TO 3 - LENGTH(ip)
145:                    temp=temp||'0'
146:            END
147:            iprebuild=temp||ip
148:
149:    EXIT 0
```

Figure 22 Sample 06: Check IP - sourcecode

Detailed description:

As usual it starts with passing the Apache request record pointer (line 11). Afterwards we instantiate an object from the class “ipcalculation” (line 14). This class covers all the needed calculations.

Line 17 and 18 provide the ip-address and the subnetmask of the webserver. These values will be split up in four parts (line 20 and 21).

The next step is to calculate the subnet and the broadcast address. This is done by calling the method “binarycalculation” of the instantiated object (line 24 to 40). The

return value is put together in two blocks (192168 and 174000 in our example). The reason for doing this is that ooREXX is having problems when we want to do calculations with numbers like 19216817400.

The method “binarycalculation” takes the part of the ip-address and converts it into the binary value (line 91). For the subnetmask we need to distinguish between “subnetid” (convert the value → line 94) and the “broadcastid” (convert 255-value → line 96).

Lines 100 to 112 are used to get 8Bit values for both values every time. We just put the proper amount of “0” in front to get a string like 00000111 for the decimal value 7.

This is followed by the binary calculation (line 115 to 128). We cut the strings down to one character (line 116,117) do the binary “and” operation for calculating the subnet part (line 120) or the ”or” operation for calculating the broadcast address. (line 122). Line 124 “collects” the results and rebuilds the string before it is transformed back to a decimal value (line 128).

The last step prior to returning the value is to make sure that we always will return three digits numbers by entering the needed leading zeros (line 131 to 135)

Building the same two blocks from the requestor ip-address (wwwremote_addr) is done in the next step. This is done similar than stated above. We take the value and split it into parts (line 44). We will call the method “buildthreedigits” to make sure that we have three digit numbers and put the string together (line 47 to 54).

After having done the hard bits we need to make sure to check that we are in the range (line 57) and place an entry in the Apache logfile (line 59). If this is not the case we will add an entry in the Apache logfile and return a 403 error page (line 62 to 64)

Like in Sample 05 to arm this program we need to place a directive in the main configuration file (httpd.conf) as shown below.

```
# Directives for Sample 06
<Directory "/var/htdocs/mydomain.com/pages/sample06">
      RexxAccessHandler
      '/var/htdocs/mydomain.com/scripts/check_ip.rex'
</Directory>
```

Figure 23 Apache configuration directive for checking webbrowser and version

Once everything is up and running it will lead to the following result when requesting a webpage when coming from the wrong ip-address …

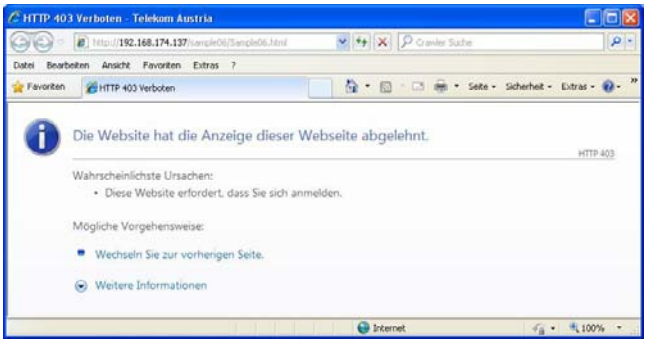

Figure 24 Sample 06: Check IP - coming from a wrong one

… or an equivalent according to your language settings and an entry in the Apache error logfile:

```
[Mon Jan 23 01:35:21 2010] [warn] [client 192.168.174.1] Check IP:
Unauthorzied access attempt (requested:/sample06/Sample06.html)
```

In case of a successful attempt the following webpage will be shown …

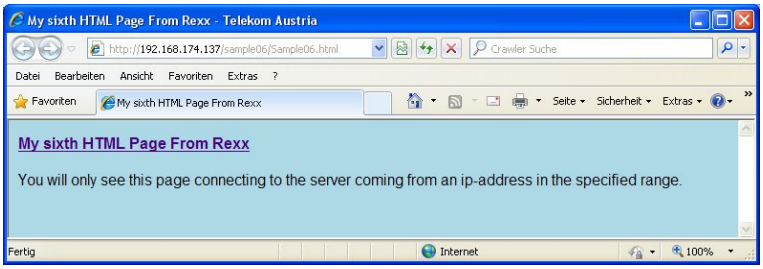

Figure 25 Sample 06: Check IP - coming from a right one

… and the following entries will be placed in the Apache error logfile:

```
[Mon Jan 23 01:40:54 2010] [info] [client 192.168.174.1] Check IP: Request
checked and o.k. (requested:/sample06/Sample06.html)
[Mon Jan 23 01:40:54 2010] [info] [client 192.168.174.1] Check IP: Request
checked and o.k. (requested:/sample06/Samples.css), referer:
http://192.168.174.137/sample06/Sample06.htm
```

5 Conclusion

Apache and mod_ooREXX are a very powerful package even we have only discovered very few parts of the possibilities so far.

Limitations in the future development might come from the lack of resources.
There are tons of books about Apache focusing on setting up, configuring and administration but as soon as it comes to system programming resources are getting rare. In combination with REXX or ooREXX it changes for the worse.

This makes it very difficult and very timeconsuming task to discover the possibilities.

Another issue is the existing bug in the Windows version. Today there are a lot of systems relying on this operating system. So can’t serving this marketshare is not an option.

On the other hand could further investigations lead to new additional perspectives in using mod_ooREXX. An example would be the combination of BSF4REXX and mod_ooREXX in a web environment.

6 Bibliography

Ford, Andrew (2008): Apache 2 Pocket reference, O'Reilly, ISBN:978-0-596-51888-2

Flatscher, Rony (2009a): An Introduction to Procedural and Object-oriented Programming (ooRexx) 2; http://wi.wu-wien.ac.at/rgf/wu/lehre/autowin/material/foils/ooRexx_2.pdf (2010-01-24)

Flatscher, Rony (2009b): An Introduction to Procedural and Object-oriented Programming (ooRexx) 3; http://wi.wu-wien.ac.at/rgf/wu/lehre/autowin/material/foils/ooRexx_3.pdf (2010-01-24)

Fosdick, Howard (2005): Rexx Programmer's Reference, Wiley Publishing, Inc., Indiana, ISBN: 0-7645-7996-7

Gröne, Bernhard; Knöpfel, Andreas; Kugel, Rudolf; Schmidt, Oliver (2004): The Apache Modeling Project; http://www.fmc-modeling.org/download/projects/apache/the_apache_modelling_project.pdf (2010-01-24)

Kew, Nick (2007): The Apache Modules Book, Prentice Hall, ISBN 0-13-240967-4

Laurie, Ben; Laurie, Peter (2003): Apache: The Definitive Guide, 3rd Edition, O'Reilly, ISBN: 978-0-596-00203-3

Wolfgarten, Sebastian (2003): Apache Webserver 2.0, Addison-Wesley, Munich

7 Appendix A: Installation of VMware Workstation

7.1 Prerequisites

You can download the latest version of VMware Workstation directly from the manufacturer by following this link http://www.vmware.com/de/products/ws/ .
VMware offers a 30 day trial version for free.

Download the file to a local disk drive of your PC

7.2 Installation

Doubleclick on the downloaded file to start the installation process.
The setup starts with loading some libraries. Afterwards the following screen will be displayed

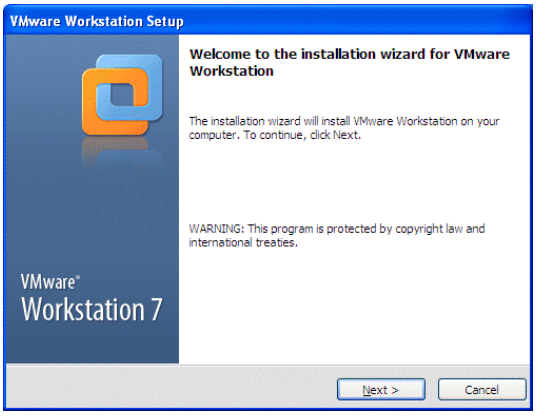

Figure 26 VMware Workstation Setup – Startup Screen

Click on the [Next] button or press ALT-N to continue.

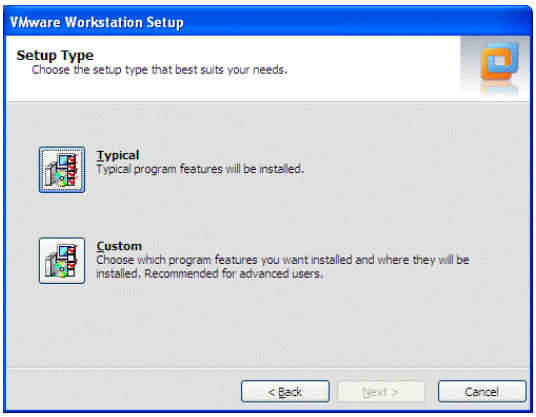

Figure 27 VMware Workstation Setup – Setup Type

Choose typical installation by clicking on the [Typical] Button or press ALT-T an continue by clicking on the [Next] button or press ALT-N to continue.

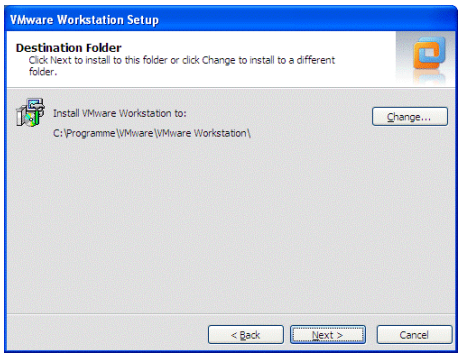

Figure 28 VMware Workstation Setup – Destination Folder

Choose an appropriate destination folder for the program files by clicking on the [Change] button or by pressing ALT-C.

We will the default values here.

Click on the [Next] button or press ALT-N to continue.

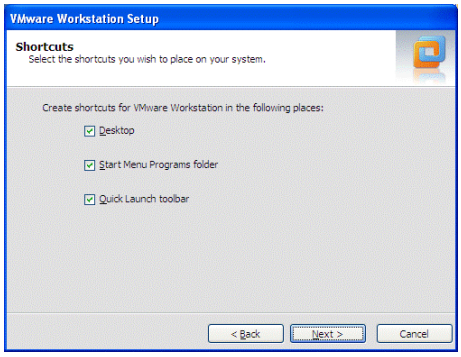

Figure 29 VMware Workstation Setup – Shortcuts

Choose the shortcuts you want to create by marking the checkboxes or leave the default values like we do.

Click on the [Next] button or press ALT-N to continue.

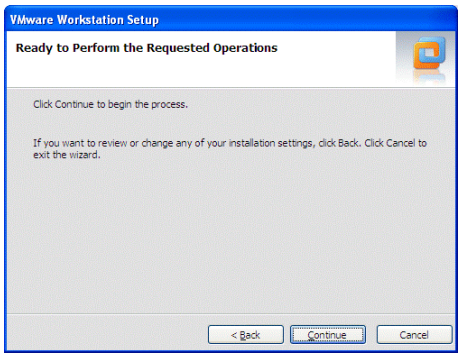

Figure 30 VMware Workstation Setup – Ready to Start installation

The installation is now ready for starting

Click on the [Continue] button or press ALT-C to start the installation.

VMWare Workstation will now be installed on your system.

After the completion of the installation the following screen appears.

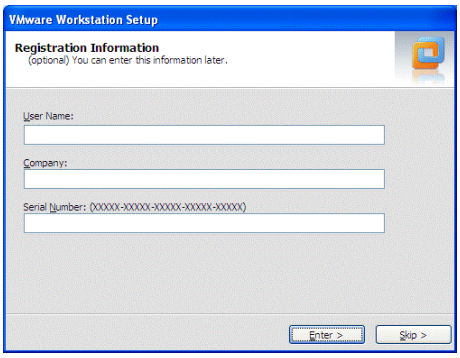

Figure 31 VMware Workstation Setup – Registration information

Type in the serial number for the trail version which can be found on the download page of the software or by clicking on the hyperlink in the mail you received.

Clicking on the [Enter] button or press ALT-E to continue.

If you want to do this later click on the [Skip] button or press ALT-S to continue.

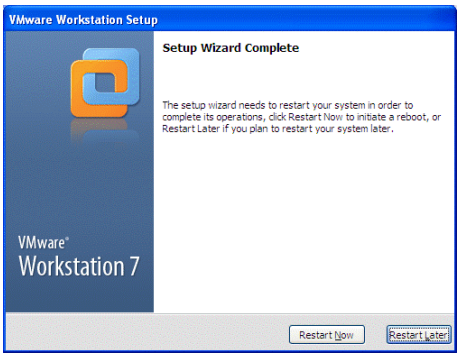

Figure 32 VMware Workstation Setup – Setup wizard completed

The Installation of the software is finished now but the PC needs to be restarted.

Restart PC by clicking on the [Restart now] button or press ALT-N.

After the reboot of the system start the application by doubleclicking on the icon

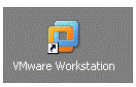

Figure 33 VMware Workstation Setup – Icon

or under All programs / VMware / VMware Workstation.

If you start the application the first time you have to accept the license agreement.

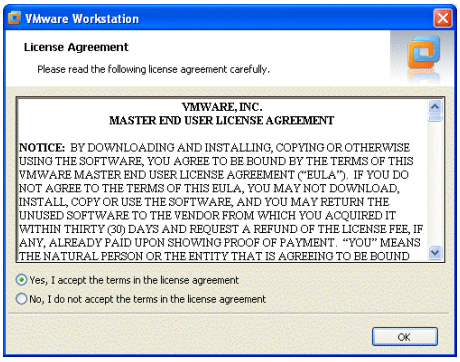

Figure 34 VMware Workstation Setup – License agreement

Mark "Yes, I accept the terms in the license agreement" and click on the [Ok] button to continue.

VMware Workstation is now ready for usage.

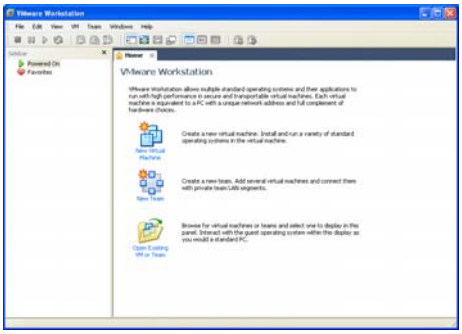

Figure 35 VMware Workstation Setup – Main screen

8 Appendix B: Setup Virtual machines: Window 2008 Server

8.1 Prerequisites

You need to have VMware Workstation installed and configured on the system (for more information see Appendix 7 on page 43).

The install media for Windows Server 2008 (including a product key) are required as well.

8.2 Installation

Start the application by doubleclicking on the icon

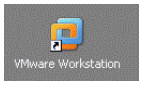

Figure 36 VMware Workstation – Icon

or under All programs / VMware / VMware Workstation.

The main window of the application (see Figure 35 on page 46) will appear.

To start the creation of a new virtual machine select File / New Virtual Machine from the menu or pres Ctrl+N

Figure 37 Windows Server 2008 Installation – Creation of virtual machine

Click on the [Next] button or press ALT-N to continue.

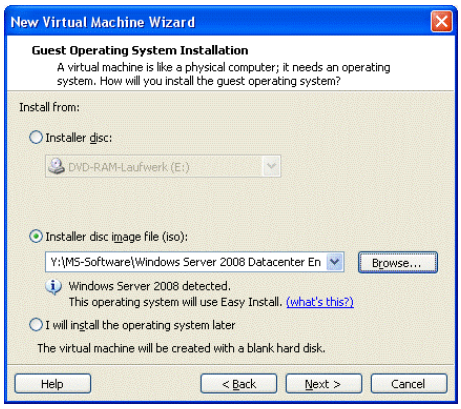

Figure 38 Windows Server 2008 Installation – Guest Operating System Sources

Choose the location of the installation directory by either browsing to the appropriate directory using the [Browse] button or typing the pathname.

Click on the [Next] button or press ALT-N to continue.

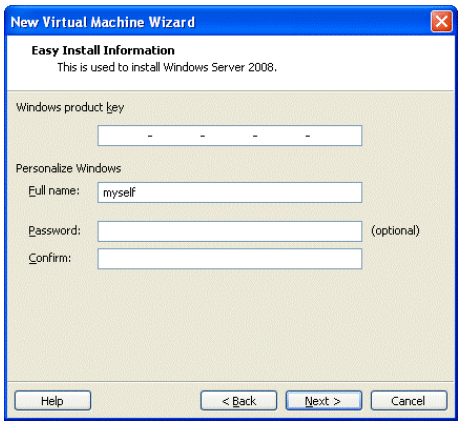

Figure 39 Windows Server 2008 Installation – Easy Install Information

Type the Product Key for Window Server 2008 in the right fields.

Click on the [Next] button or press ALT-N to continue.

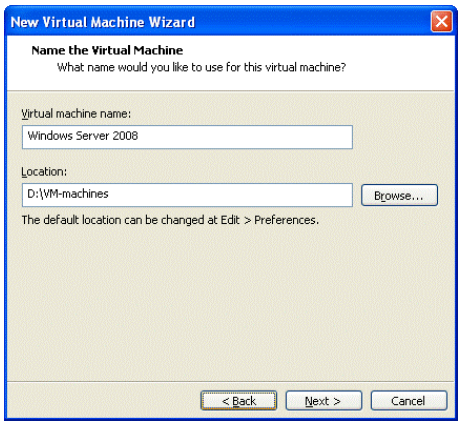

Figure 40 Windows Server 2008 Installation – Name the virtual machine

During the next step in the process you have to name the Virtual machine.

Feel free to use any name you like or which suits your environment.

Afterwards click on the [Next] button or press ALT-N to continue.

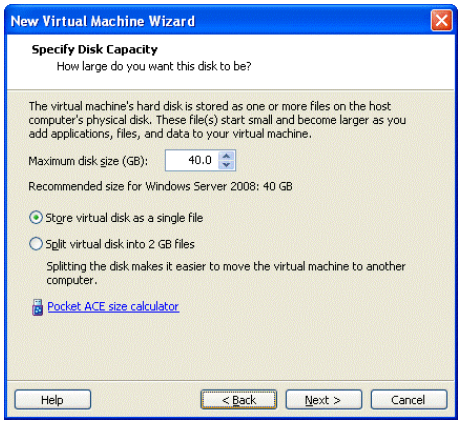

Figure 41 Windows Server 2008 Installation – Specify Disk Capacity

Here we will use the defaults recommended by VM Ware.

Click on the [Next] button or press ALT-N to continue.

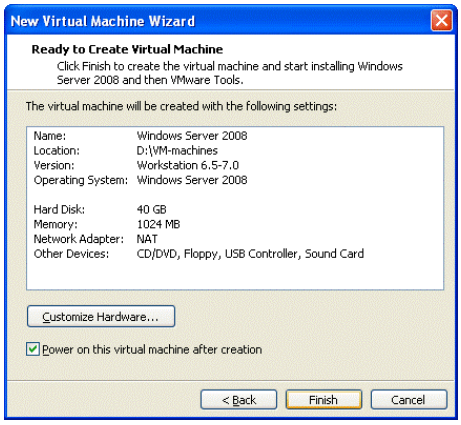

Figure 42 Windows Server 2008 Installation – Ready to create Virtual Machine

Final chance for checking the settings prior to the creation of the virtual machine and the begin of the installation process.

Click on the [Finish] button or press ALT-F to continue.

Setup is now copying the needed files from the installation media.

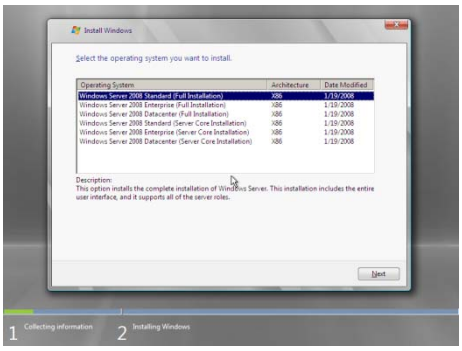

Figure 43 Windows Server 2008 Installation – Select Operating System

We will use “Windows Server 2008 Standard installation” Select this application in the menu and click on “Next or press ALT-N to continue.

After the completion of the installation and the first login the following screen will be displayed.

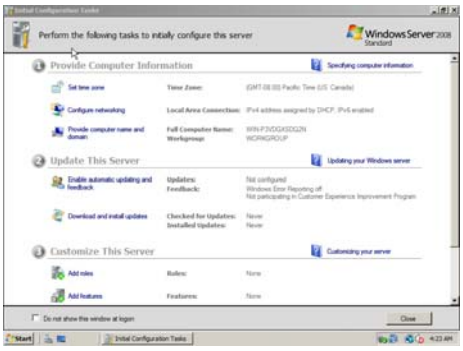

Figure 44 Windows Server 2008 Installation – Initial configuration tasks

8.3 Initial configuration and customization

8.3.1 Initial configuration of the Windows 2008 Server

Perform the following initial configuration tasks by clicking on the hyperlinks in the Window shown in Figure 44 on page 50:

- Set time zone
- Provide Computer name and domain
- Enable automatic updating
- Download and install updates (see below)

After clicking on the hyperlink you will get the following window.

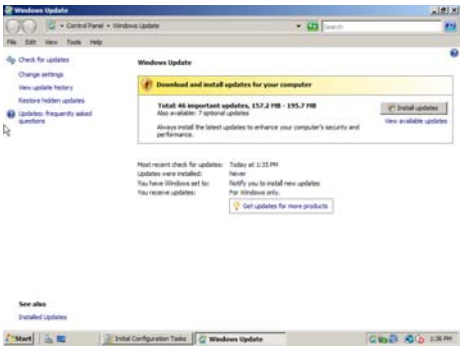

Figure 45 Windows Server 2008 Installation – Windows Update

Click on the button [Install updates] to continue.

Once the installation of updates is completed the following window will appear.

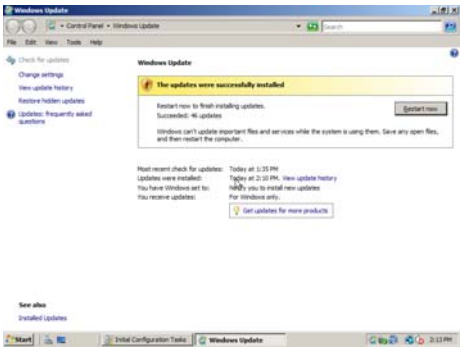

Figure 46 Windows Server 2008 Installation – Windows Update completed

Prior to continuing you might need to reboot the server by clicking on the [Restart now] or by pressing ALT-R to continue.

After the reboot no further customization is needed and the initial configuration is completed.

8.3.2 Add a second hard disc for Apache Data files

Prior to adding the device you need to shutdown the Virtual machine by shutting down the server.

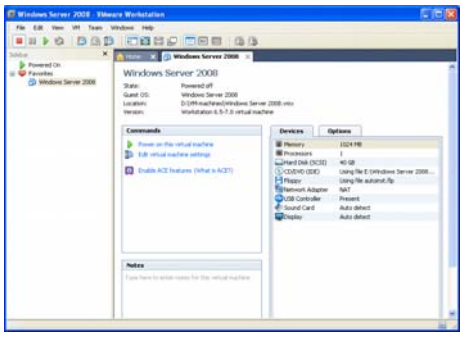

Figure 47 Windows Server 2008 Installation – VMware Workstation Main Screen

Click on the hyperlink “Edit virtual machine settings” to enter the setup screen.

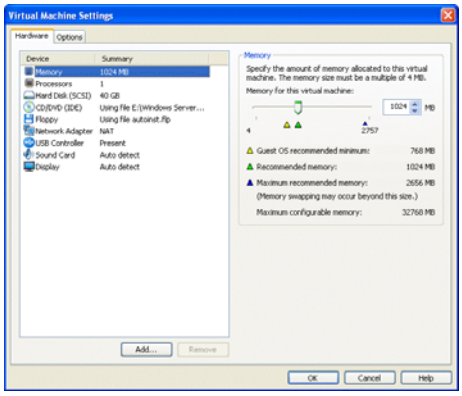

Figure 48 Windows Server 2008 Installation – Virtual Machine Setup

Click on the [Add] button to add hardware devices.

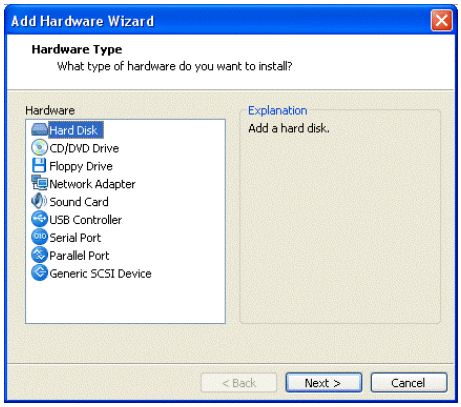

Figure 49 Windows Server 2008 Installation – Add hardware wizard

Select “Hard Disk” and click on the [Next] button to continue.

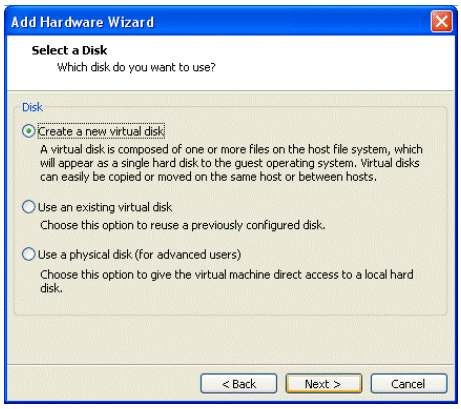

Figure 50 Windows Server 2008 Installation – Add hardware wizard (Disk) - I

Click on the [Next] button to continue

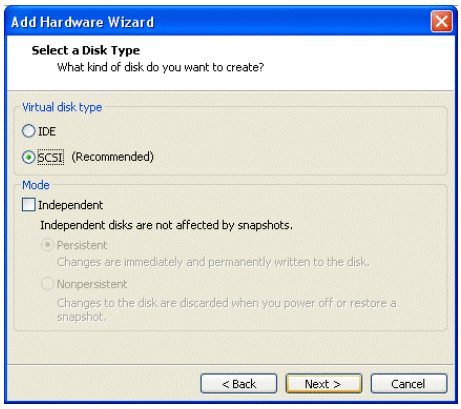

Figure 51 Windows Server 2008 Installation – Add hardware wizard - II

Click on the [Next] button to continue

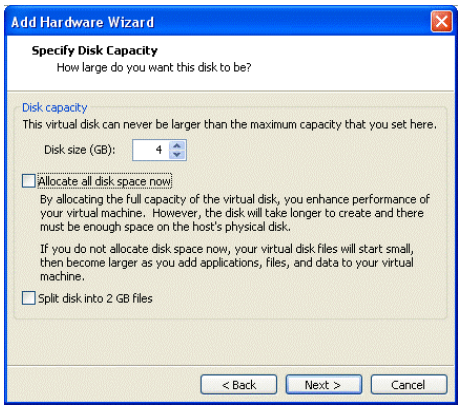

Figure 52 Windows Server 2008 Installation – Add hardware wizard – Specify capacity

Adjust the disk size to your needs. We will use 4GB which is more than enough in our case.

Click on the [Next] button to continue

Figure 53 Windows Server 2008 Installation – Add hardware wizard – specify disk file

Click on the [Finish] button to create the disk.

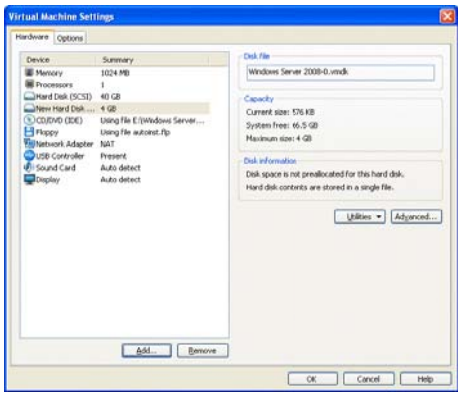

Figure 54 Windows Server 2008 Installation – VMware Workstation with second disk

Afterwards the application will return to the main window showing you the new hard disk there.

Click on the [OK] button to finish the setup.

Finally we need to make the disk available to the system.

For this we need to power on the virtual machine again by pressing Ctrl+B or clicking on the hyperlink “Power on this virtual machine”.

After logging in go to Start / Run / Control Panel / Administrative Tools

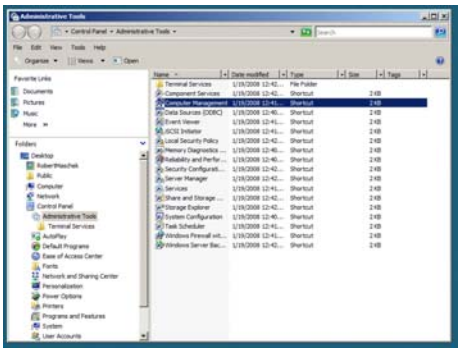

Figure 55 Windows Server 2008 Installation – Administrative Tools

Open the Computer Management by doubleclicking on the menu entry.

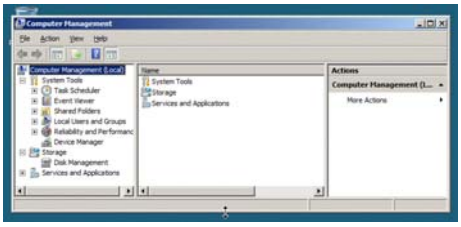

Figure 56 Windows Server 2008 Installation – Computer Management

In the Computer Management click on Storage / Disk Management. The Initialize Disk window appears automatically.

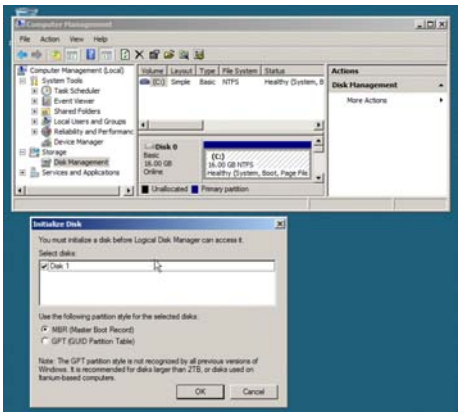

Figure 57 Windows Server 2008 Installation – Initialize disk

Click on the [OK] button to continue.

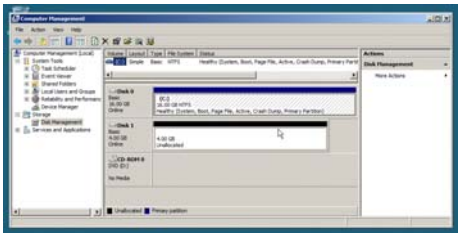

Figure 58 Windows Server 2008 Installation – Volume creation

Right Click on Disk 1 on select New Simple Volume from the appearing menu.

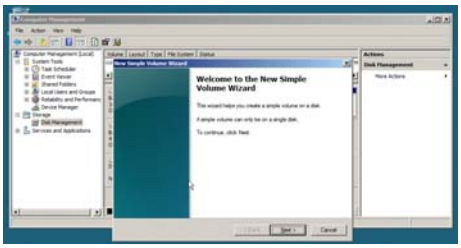

Figure 59 Windows Server 2008 Installation – New simple volume wizard

Click on the [Next] button or press ALT-N to continue.

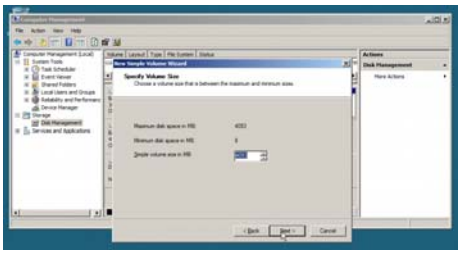

Figure 60 Windows Server 2008 Installation – System volume size

Click on the [Next] button or press ALT-N to continue.

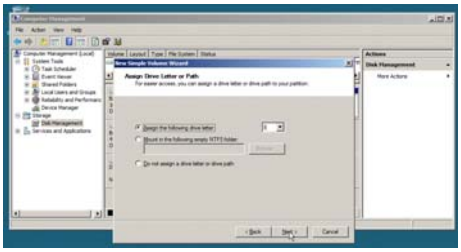

Figure 61 Windows Server 2008 Installation – Drive letter

Change the drive letter so that it suits your need and click on the [Next] button or press ALT-N to continue.

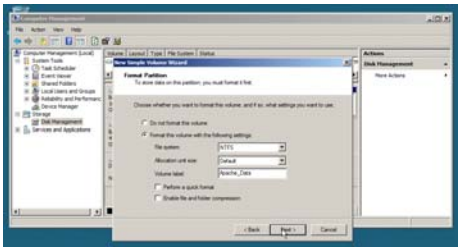

Figure 62 Windows Server 2008 Installation – Format partition

Click on the [Next] button or press ALT-N to continue.

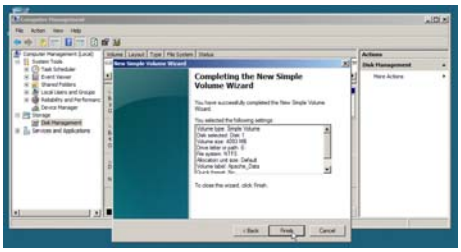

Figure 63 Windows Server 2008 Installation – Completion

Click on the [Finish] button to complete the addition of the disk.

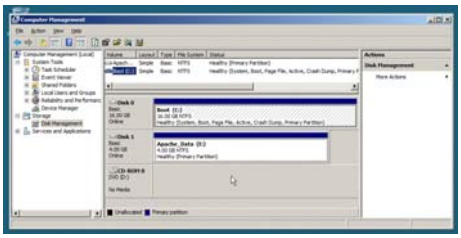

Figure 64 Windows Server 2008 Installation – Volume added

8.3.3 Activation of the Operating system

Since Windows 2008 is not Open Source or freeware you need to activate it to use all features.

To do this there are a lot of different ways depending on the license you are having. For this reason we are not able to care about this.

9 Appendix C: Setup Virtual machines: Fedora 12

9.1 Prerequisites

You can download Fedora 12 from the homepage of the Fedora project using the following hyperlink:
http://download.fedoraproject.org/pub/fedora/linux/releases/12/Fedora/i386/iso/Fedora-12-i386-DVD.iso

Prior to the installation it is recommended that you verify the integrity of the downloaded file.
For more information how to do this visit: http://docs.fedoraproject.org/readme-burning-isos/en_US/sn-validating-files.html

9.2 Installation

Start the application by doubleclicking on the icon

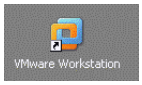

Figure 65 VMware Workstation – Icon

or under All programs / VMware / VMware Workstation.

The main window of the application (see Figure 35 on page 46) will appear.

To start the creation of a new virtual machine select File / New Virtual Machine from the menu or pres Ctrl+N

Figure 66 Fedora 12 installation – Creation of virtual machine

Click on the [Next] button or press ALT-N to continue.

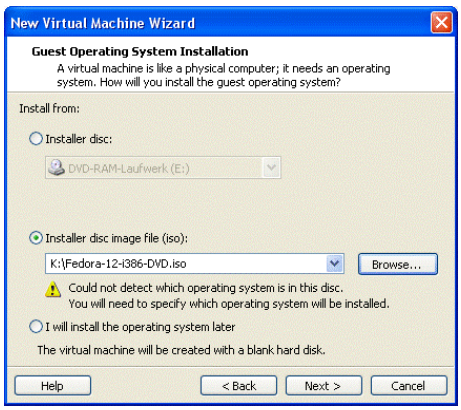

Figure 67 Fedora 12 Installation – Guest Operating System Sources

Choose the location of the installation directory by either browsing to the appropriate directory using the [Browse] button or typing the pathname.

Click on the [Next] button to continue.

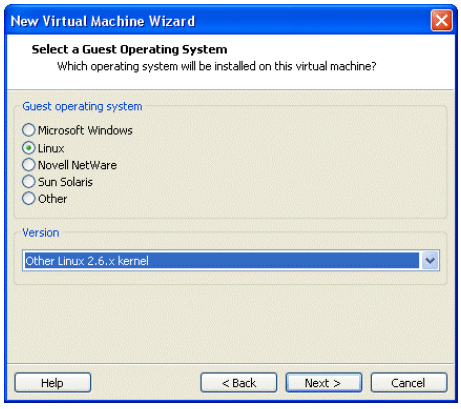

Figure 68 Fedora 12 Installation – Select Guest Operating System

Click on the [Next] button to continue.

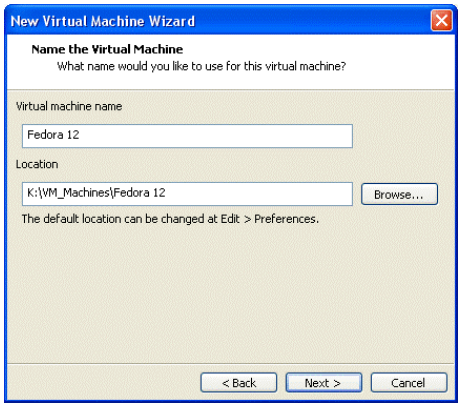

Figure 69 Fedora 12 Installation – Name virtual machine

Click on the [Next] button to continue.

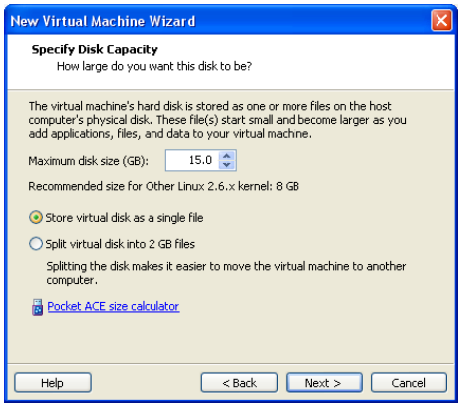

Figure 70 Fedora 12 Installation – Specify disk capacity

According to the system requirements Fedora 12 needs up to 9GB. So we change the disk size to 12GB.

Click on the [Next] button to continue.

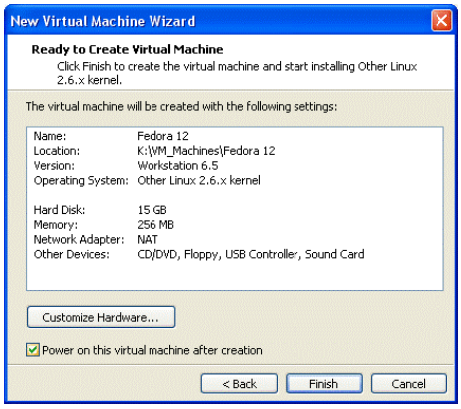

Figure 71 Fedora 12 Installation – Summary screen

Click on the [Customize Hardware] button to change the memory settings to 512MB

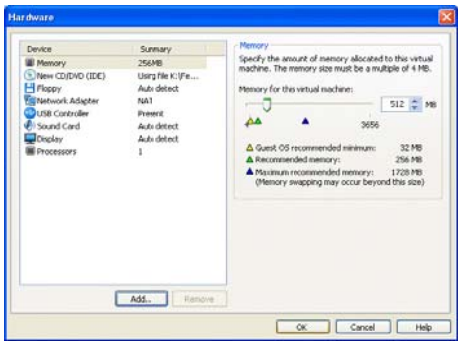

Figure 72 Fedora 12 Installation – Change memory settings

Click on the [OK] button to finalize the change and check the changed settings in the summary screen.

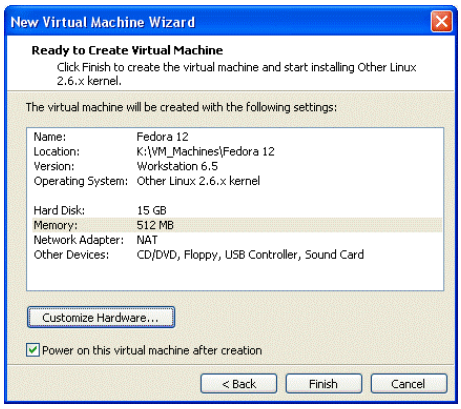

Figure 73 Fedora 12 Installation – Ready to create virtual machine

Click on the [Finish] button to start the creation of the virtual machine and continue with the installation.

Figure 74 Fedora 12 Installation – Setup welcome screen

Choose “Install or upgrade an existing system” and press [Enter] or wait until the installation starts.

Figure 75 Fedora 12 Installation – Disc Found

Press the [Skip] button to continue.

Figure 76 Fedora 12 Installation – Installation start

Click on the [Next] button or press ALT-N to continue.

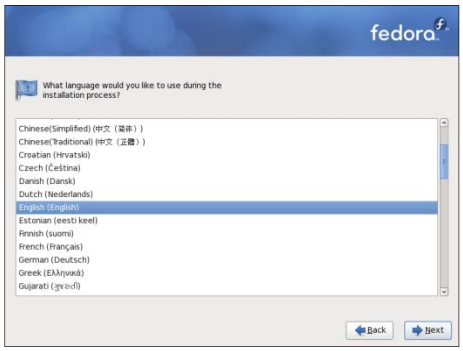

Figure 77 Fedora 12 Installation – Installation language

Choose your language for the setup process and click on the [Next] button or press ALT-N to continue.

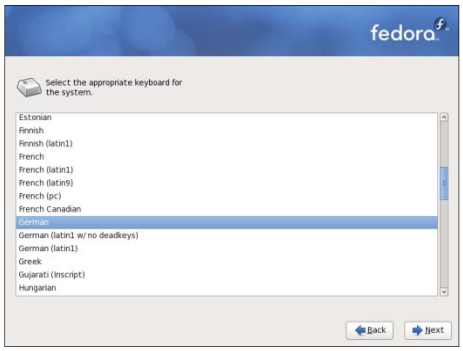

Figure 78 Fedora 12 Installation – Installation keyboard layout

Select the appropriate keyboard and click on the [Next] button or press ALT-N to continue.

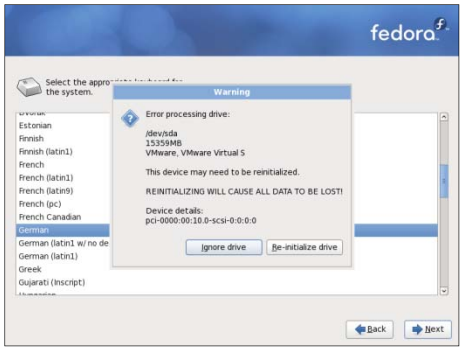

Figure 79 Fedora 12 Installation – Drive initizialisation

Click on the [Re-initialize drive] button or press ALT-R to continue.

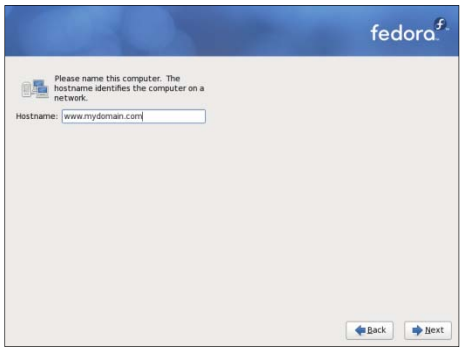

Figure 80 Fedora 12 Installation – Set hostname

Enter the full qualified name (FQN) of the machine and click on the [Next] button or press ALT-N to continue.

Figure 81 Fedora 12 Installation – Set timezone

Click on the [Next] button or press ALT-N to continue.

Figure 82 Fedora 12 Installation – Set root password

Type the root password twice and click on the [Next] button or press ALT-N to continue.

If you don’t choose an appropriate password you will get a warning

Figure 83 Fedora 12 Installation – Weak root password warning

Click on the [Cancel] button or press ALT-C to return to the Set root password menu again.

Providing weak passwords can be dangerous for the system. For this reason clicking on the [Use Anyway] button is not recommended.

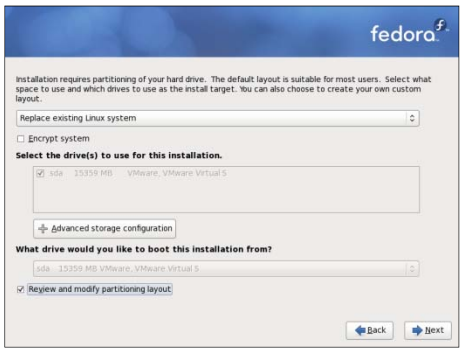

Figure 84 Fedora 12 Installation – Hard drive partitioning

Mark the checkbox “Review and modify partitioning layout” and click on the [Next] button or press ALT-N to continue.

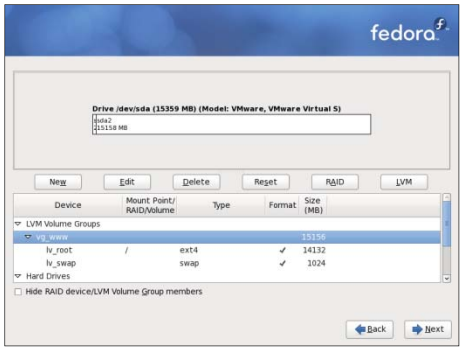

Figure 85 Fedora 12 Installation – Edit hard drive partitioning

Select the LVM volume group vg_www and click on the [Edit] button or press ALT-E to continue.

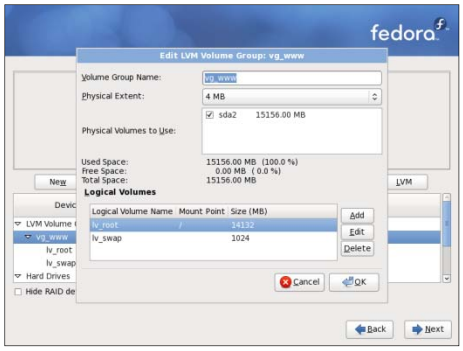

Figure 86 Fedora 12 Installation – Edit lv_root

Select the logical volume lv_root and click on the [Edit] button or press ALT-E to continue.

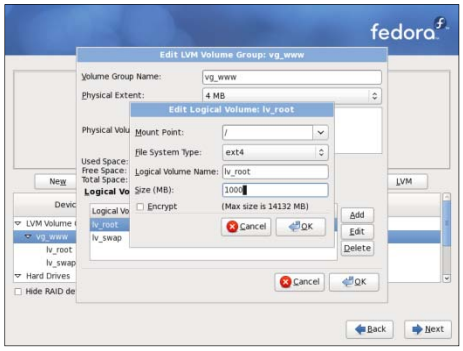

Figure 87 Fedora 12 Installation – lv_root new size

Set the Size (MB) to 10000 and click on [Ok] or press ALT-O to continue.

In the previous menu (see Figure 86 on page 65) click on the [Add] button or press ALT-A to create a new logical volume.

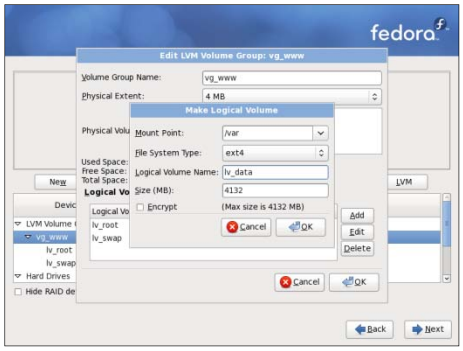

Figure 88 Fedora 12 Installation – Create a new logical volume

Set the Size (MB) to 4152 and click on [Ok] or press ALT-O to continue.

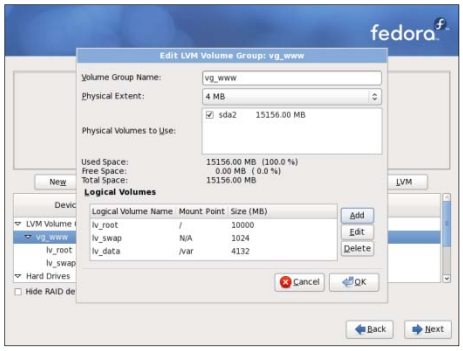

Figure 89 Fedora 12 Installation – LVM volume group: vg_www

Check the new settings and click on the [OK] button or press ALT-O to continue.

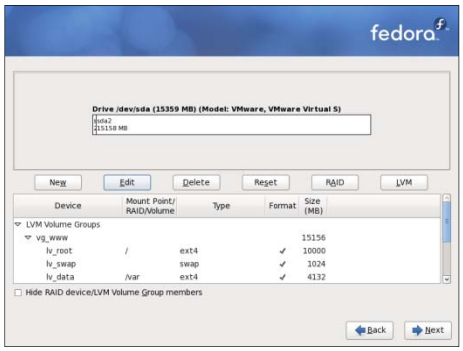

Figure 90 Fedora 12 Installation – Changed hard drive settings

Click on the [Next] button or press ALT-N to continue

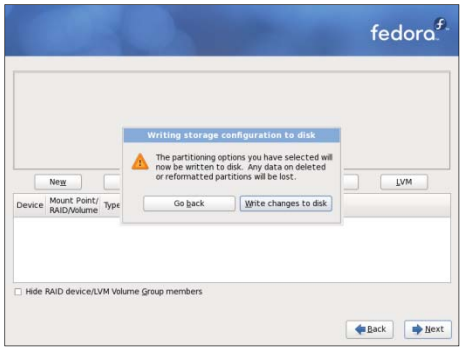

Figure 91 Fedora 12 Installation – Hard drive partitioning – write changes to disk

A warning will be displayed prior to saving the new layout. Click on the [Write changes to disk] or press ALT-W to continue.

Fedora has now created the filesystem.

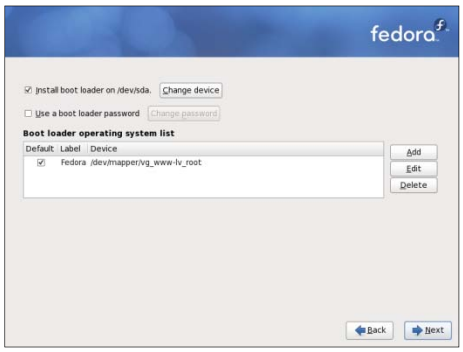

Figure 92 Fedora 12 Installation – Boot loader installation

For the Boot loader installation we will leave the settings unchanged. Click on the [Next] button or press ALT-N to continue.

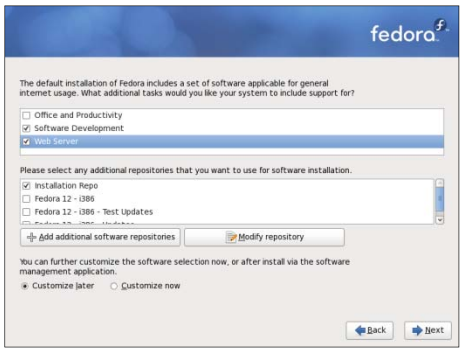

Figure 93 Fedora 12 Installation – Change install packages

From the default installation remove “Office and productivity” and add “Software Development and Webserver” by marking or unmarking the checkbox. Click on the [Next] button or press ALT-N to continue.

The installation is now started.

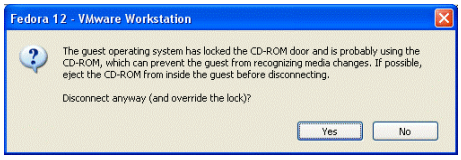

Figure 94 Fedora 12 Installation – VMware Workstation drive lock

During the installation the above shown warning message appears.

Click on the [Yes] button to continue.

The installation is now finished and a reboot of the server is needed.

9.3 Initial configuration and customization

9.3.1 Initial configuration of Fedora 12

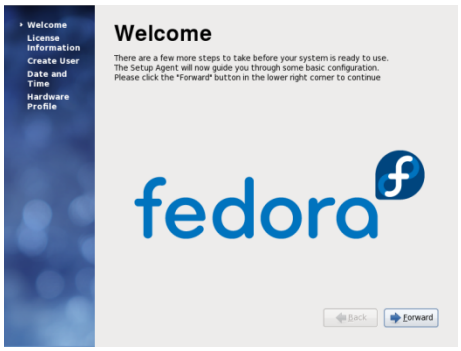

Figure 95 Fedora 12 Installation – Initial configuration

After the reboot and logging in you will see the following startup screen. Click on the [Forward] button or press ALT-F to continue.

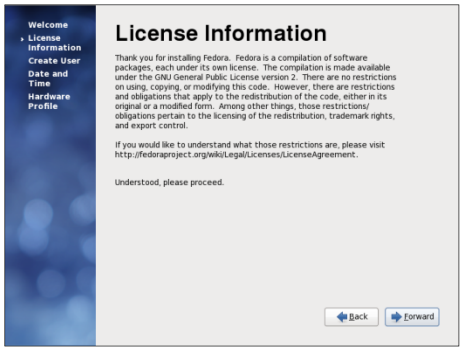

Figure 96 Fedora 12 Installation – License information

Accept the license information by clicking on the [Forward] button or press ALT-F to continue.

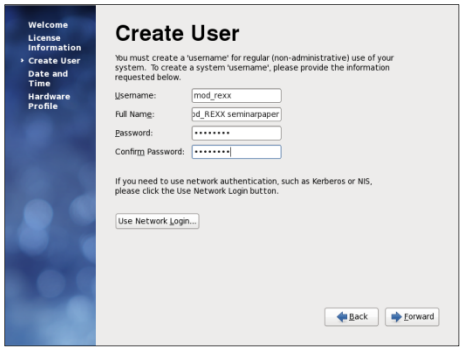

Figure 97 Fedora 12 Installation – Create User

Once you have filled in the needed fields click on the [Forward] button or press ALT-F to continue.

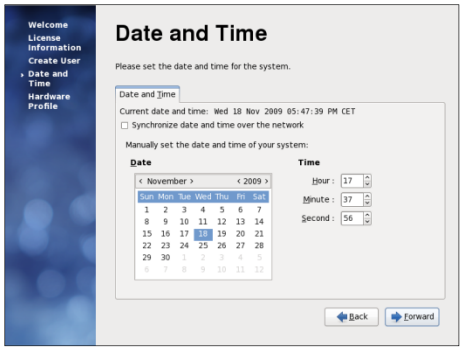

Figure 98 Fedora 12 Installation – Date and Time

Click on the [Forward] button or press ALT-F to continue.

Figure 99 Fedora 12 Installation – Hardware profile

Mark the checkbox “Do not send profile” and click on the [Finish] button or press ALT-F to continue.

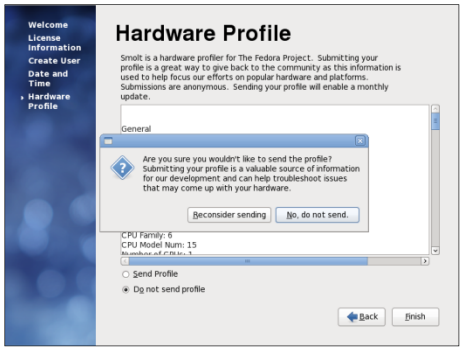

Figure 100 Fedora 12 Installation – Hardware profile sending

We still haven't changed our mind. So click on the [No, don not send] button or press ALT-N to continue.

Now the final configuration takes place. The system needs to be restarted afterwards.

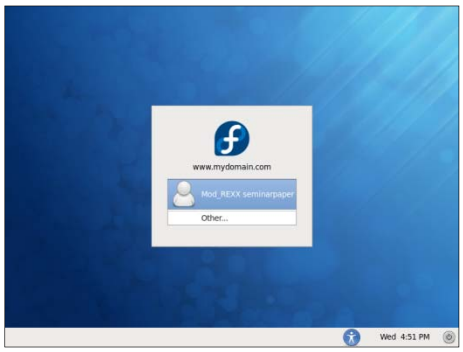

Figure 101 Fedora 12 Installation – Initial configuration finished

After the reboot you will see the following login screen.

9.3.2 Enable Network Connection

The network connection for the system is disabled by default and needs to be enabled now.

Figure 102 Fedora 12 Installation – Desktop

After logging in with the created account (see Figure 97 on page 69)

Click on System / Preferences / Network Connections

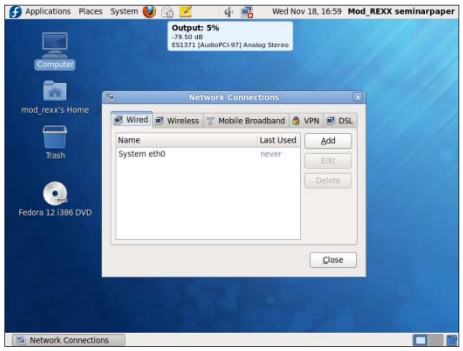

Figure 103 Fedora 12 Installation – Network connections

Click on “System eth0” and the [Edit] button or press ALT-E to continue.

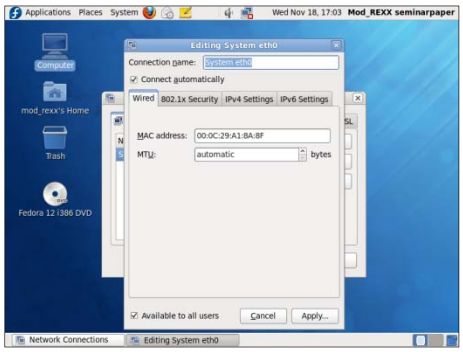

Figure 104 Fedora 12 Installation – Ethernet connection

Mark the checkbox “Connect automatically” and Apply (there is no keyboard shortcut for this).

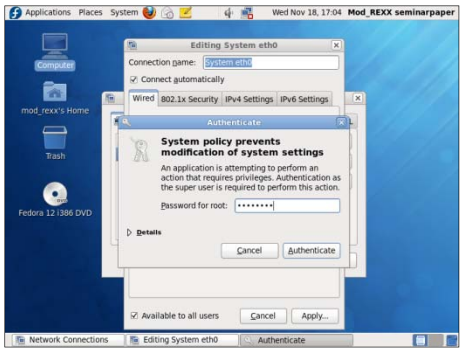

Figure 105 Fedora 12 Installation – Root user authentication needed

This change needs to be done using the root user. So enter the root password and click on the [Authenticate] button or press ALT-A to continue.

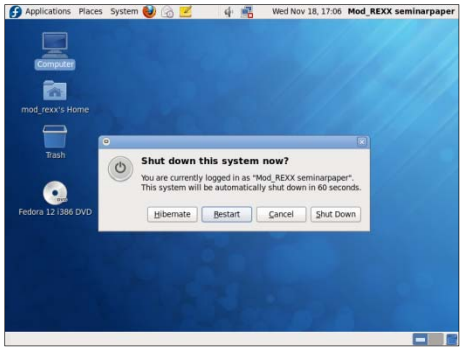

Figure 106 Fedora 12 Installation – System restart needed

After this change the system needs to be restarted once more.

Once the system is up again the installation and configuration has been finished.

10 Appendix D: Installing Apache on Windows 2008 Server

10.1 Prerequisites

10.1.1 Add Apache Website to the trusted sites

Open Tools / Internet Options / Security

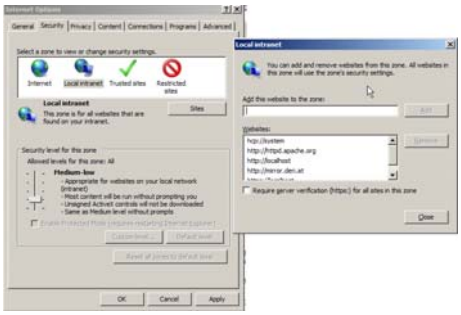

Figure 107 Apache on Windows 2008 – Adjust trusted sites

Type http://httpd.apache.org in the field “Add the website to the zone” and click on the [Add] button or press ALT-A to continue.

Otherwise the following message will be displayed.

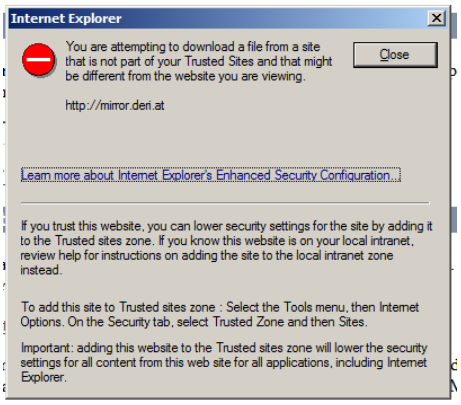

Figure 108 Apache on Windows 2008 – Adjust trusted sites warning message

10.1.2 Download the software

Download the Apache Webserver from the Apache homepage

http://httpd.apache.org/download.cgi

Click on the file “Win32 Binary including OpenSSL 0.9.8k (MSI Installer)” (Filename: apache_2.2.14-win32-x86-openssl-0.9.8k.msi) to start the download.

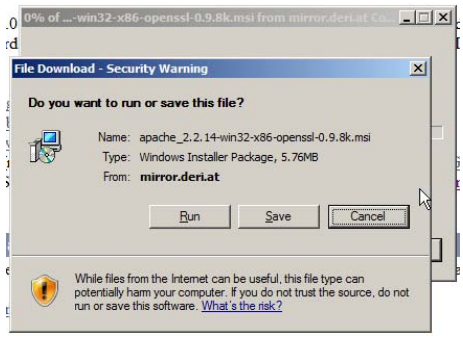

Figure 109 Apache on Windows 2008 – Download Apache software

Click on the [Save] button or press ALT-S to continue.

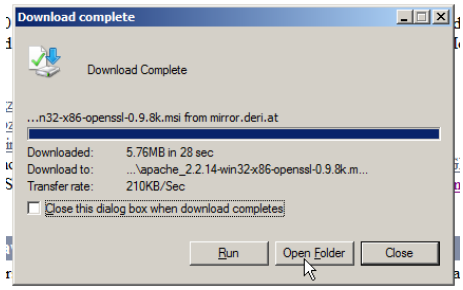

Figure 110 Apache on Windows 2008 – Open download folder

After the completion of the download click on the [Open folder] button or press ALT-F to continue.

10.1.3 Option: MD5 value check

To verify the integrity of the downloaded file we will calculate the MD5 hash value of the downloaded file. This can be done for example by using the MD5 calculator available for download at http://www.softpedia.com/get/System/File-Management/MD5-calculator.shtml

Figure 111 Apache on Windows 2008 – MD5 hash value calculator

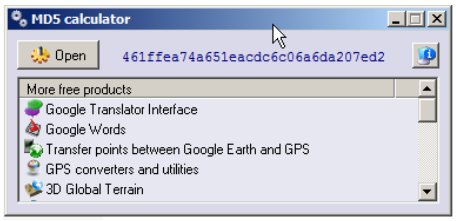

Figure 112 Apache on Windows 2008 – Apache MD5 hash value calculation

For the file apache_2.2.14-win32-x86-openssl-0.9.8k.msi the MD5 value should be 461ffea74a651eacdc6c06a6da207ed2.

10.2 Installation

Start the installation by running apache_2.2.14-win32-x86-openssl-0.9.8k.msi file either from the command prompt or by doubleclicking on the downloaded file.

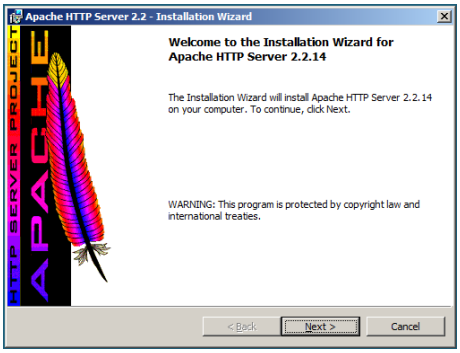

Figure 113 Apache on Windows 2008 – Installation welcome screen

Click on the [Next] button or press ALT-N to continue.

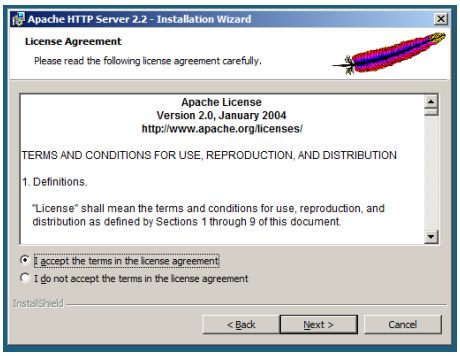

Figure 114 Apache on Windows 2008 – Apache License

Select “I accept the terms in the license agreement” and lick on the [Next] button or press ALT-N to continue.

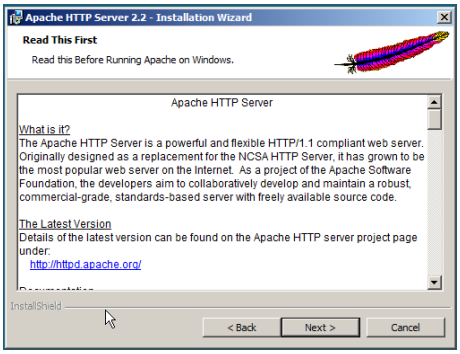

Figure 115 Apache on Windows 2008 – Read this first

Click on the [Next] button or press ALT-N to continue.

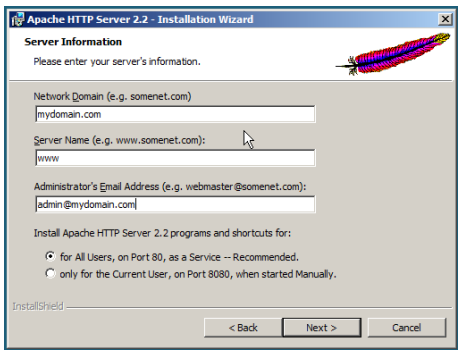

Figure 116 Apache on Windows 2008 – Server information

Type in all the needed configuration details for the server and click on the [Next] button or press ALT-N to continue.

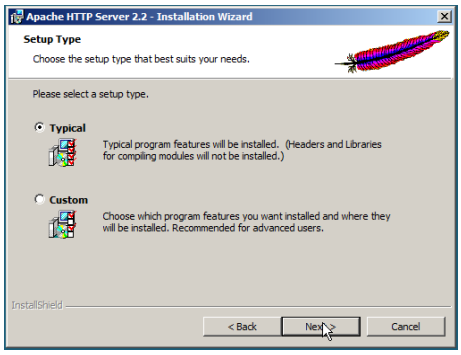

Figure 117 Apache on Windows 2008 – Installation type

Select “Typical” for the setup type and click on the [Next] button or press ALT-N to continue.

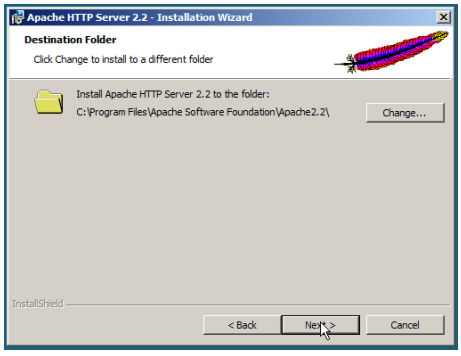

Figure 118 Apache on Windows 2008 – Destination folder

Choose the appropriate installation folder (we will use the default) and click on the [Next] button or press ALT-N to continue.

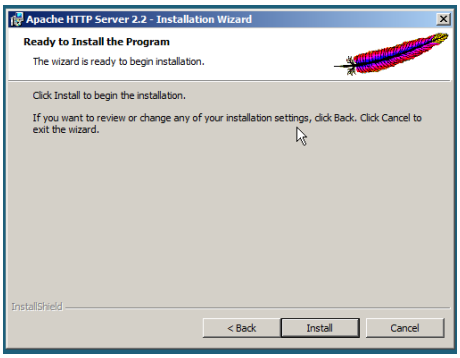

Figure 119 Apache on Windows 2008 – Ready to install

After entering the needed details we are now ready to install the software. So click on the [Install] button to start installing the application.

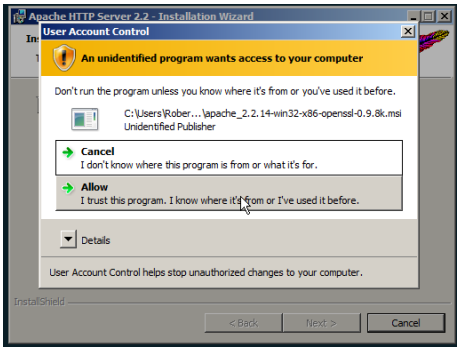

Figure 120 Apache on Windows 2008 – Access control warning I

To prevent unauthorized changes Windows comes up the “User Account Control” window. Click on Allow to continue.

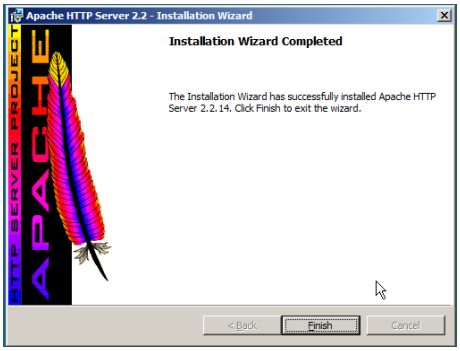

Figure 121 Apache on Windows 2008 – Installation wizard completed

The software is being installed now and the above shown window is displayed as soon as the installation is finished. Click on the [Finish] button or press ALT-F to continue.

Finally we will check that the Webserver is up and running. To do so we open the browser on the server and go to http://localhost

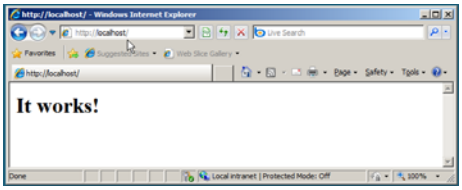

Figure 122 Apache on Windows 2008 – Test installation

If you are able to see the message shown in Figure 122 the installation has been finished successfully.

10.3 Initial installation and customization

10.3.1 Changing htdocs

The default location for the htdocs directory is located in the Apache program directory. We want to switch it to another directory for security reasons but this need to be done from and Administrator account.

Open the main configuration file of the Webserver which is located at C:\Program Files\Apache Software Foundation\Apache2.2\conf\httpd.conf.

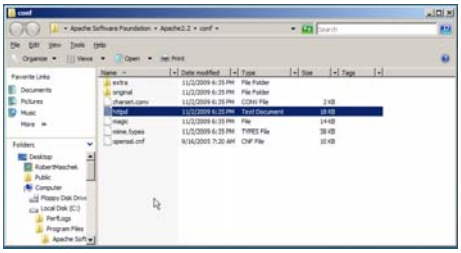

Figure 123 Apache on Windows 2008 – httpd.conf file location

Search for the following sections

```
DocumentRoot "C:/Program Files/Apache Software Foundation/Apache2.2/htdocs"
```

and change it to the new directory

```
DocumentRoot "E:/htdocs/mydomain.com/pages"
```

The same procedure for

```
#
# This should be changed to whatever you set DocumentRoot to.
#
```

```
<Directory "C:/Programme/Apache Software Foundation/Apache2.2/htdocs">
```

which should be changed to

```
<Directory "E:/htdocs">
```

as well.

Copy the content of the old htdocs Directory to the new one.

To finish change the server needs to read the httpd.conf file again. For this reason we need to restart the server by opening the Apache Servicemonitor

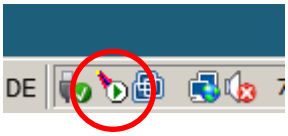

Figure 124 Apache on Windows 2008 – Apache servicemonitor

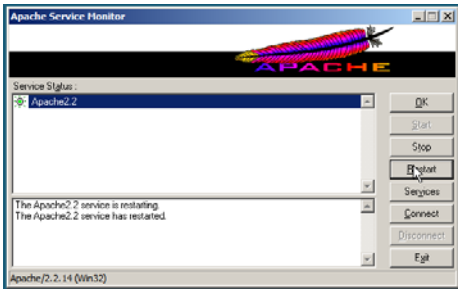

Figure 125 Apache on Windows 2008 – Server restart

and click on the [Restart] button or press ALT-R to restart the server.

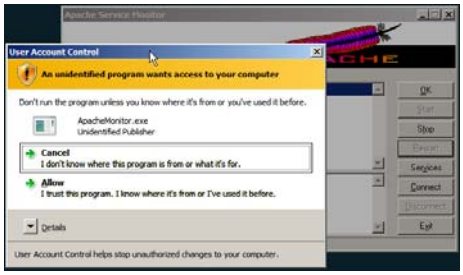

Figure 126 Apache on Windows 2008 – Access control warning II

Once more Windows tries to prevent unauthorized changes and displays the “User Account Control” window. Click on [Allow] to continue.

After the restart we will check again that the Webserver is up and running. To do so we open the browser on the server and go to http://localhost .

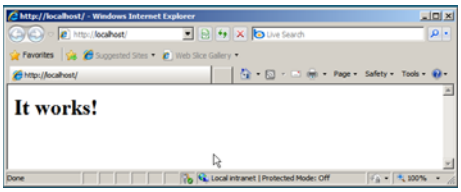

Figure 127 Apache on Windows 2008 – Test modified installation

10.3.2 Adjusting Windows Firewall

To allow external connection we need to adjust the Windows firewall.

Go to Start /Control Panel / Windows Firewall

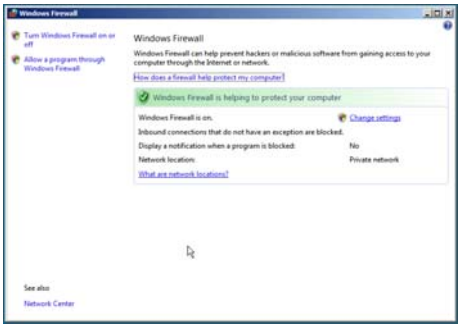

Figure 128 Apache on Windows 2008 – Windows firewall

Click on the hyperlink Change Settings

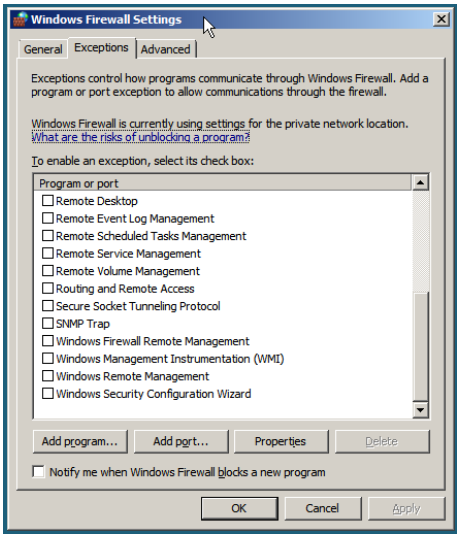

Figure 129 Apache on Windows 2008 – Windows firewall settings

and select Expections. Click on the [Add Port] button or press ALT-O to continue.

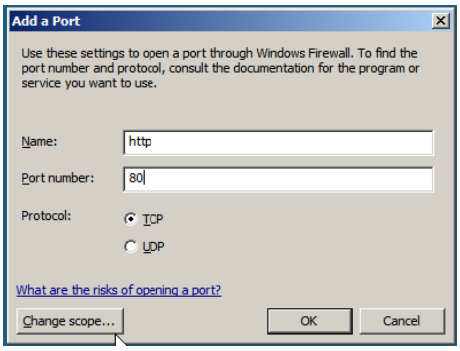

Figure 130 Apache on Windows 2008 – Add a port

Type the name (http) and the port number (80). Click on [Ok] twice to close all open windows.

Check that access to the Webserver is possible from a remote machine.

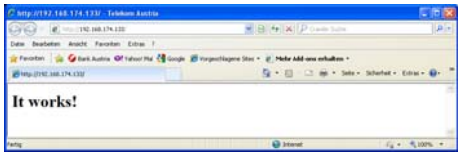

Figure 131 Apache on Windows 2008 – Testing connection from remote

If you are able to see the message shown in Figure 131 the work has been finished successfully.

11 Appendix E: Installing Apache on Fedora 12

11.1 Prerequisites

We have decided to build the software from the source code and not to use precompiled rpm packages so that it is easier to use these instructions for other UX systems as well.

11.1.1 Download the software

Login to the system and open a terminal window. Switch to the root user for download and installation

```
[mod_rexx@www ~]$ su
Password: ******
[root@www mod_rexx]# wget http://archive.apache.org/dist/httpd/httpd-
2.2.14.tar.gz
```

11.1.2 Option: MD5 value check

Download the MD5 signature file for the Apache file.

```
[root@www mod_rexx]# wget http://www.apache.org/dist/httpd/httpd-
2.2.14.tar.gz.md5
```

Read the content of the file.

```
[root@www mod_rexx]# cat httpd-2.2.14.tar.gz.md5
MD5 (httpd-2.2.14.tar.gz) = 2c1e3c7ba00bcaa0163da7b3e66aaa1e
```

Calculate the MD5 hash value of the downloaded file.

```
[root@www mod_rexx]# md5dsum httpd-2.2.14.tar.gz
2c1e3c7ba00bcaa0163da7b3e66aaa1e  httpd-2.2.14.tar.gz
```

Both values have to be identical.

11.2 Installation

Unpack the file.

```
[root@www mod_rexx]# tar xvzf httpd-2.2.14.tar.gz
```

Change to the directory with the unpacked files.

```
[root@www mod_rexx]# cd httpd-2.2.14
```

Install the software (for future need we also include some optional modules).

```
[root@www httpd-2.2.14]# ./configure
[root@www httpd-2.2.14]# make
[root@www httpd-2.2.14]# make install
```

Start the Webserver.

```
[root@www httpd-2.2.14]# /usr/local/apache/bin/apachectl start
```

Close the terminal session

```
[root@www httpd-2.2.14]# exit
[mod_rexx@www ~]$ exit
```

To allow access to the system we need to adjust the firewall of the system.

Go to System / Administration /Firewall

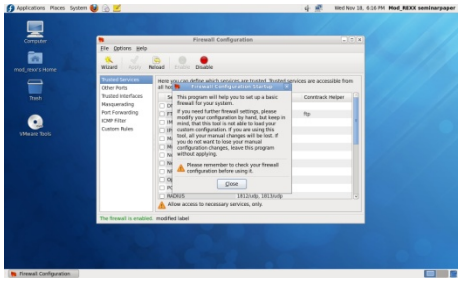

Figure 132 Apache on Fedora 12 – Firewall setup

Click on the [Close] button or press ALT-C to continue

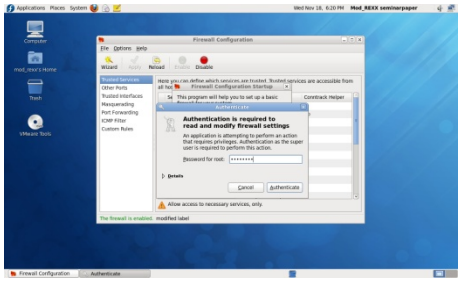

Figure 133 Apache on Fedora 12 – Authentication needed

Enter root password and click on the [Authenticate] button or press ALT-A to continue.

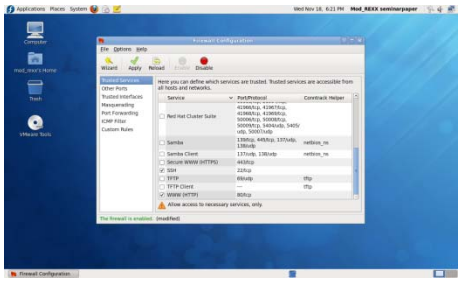

Figure 134 Apache on Fedora 12 – Firewall port opened

Scroll down and mark the checkbox by WWW (HTTP). To change the firewall ruleset click on [Apply].

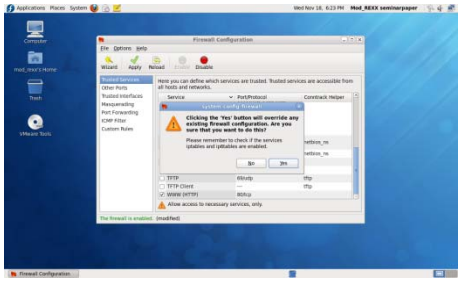

Figure 135 Apache on Fedora 12 – Firewall configuration change

Click on the [Yes] button or press ALT-Y to continue.

The Installation is finished now and we need to check that it works.

Open the Webbrowser on the server and go to http://localhost

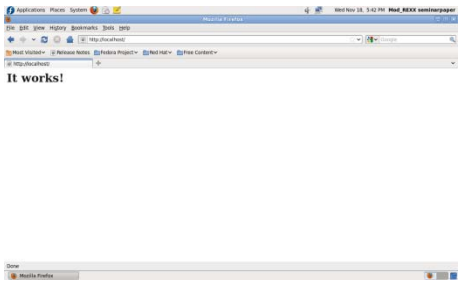

Figure 136 Apache on Fedora 12 – Connection test

You should see the screen shown in Figure 136.

11.3 Customization:

11.3.1 Adding startup

There's already a startup file for the Apache Webserver in the /etc/init.d directory but this file is for the version of the Webserver which comes with the distribution. So we need to replace the content of the existing file with the following lines:

```
#!/bin/bash
#
# httpd        Startup script for the Apache HTTP Server
#
# chkconfig: - 85 15
# description: The Apache HTTP Server is an efficient and extensible  \
#            server implementing the current HTTP standards.
# processname: httpd
# config: /usr/local/apache2/conf/httpd.conf
# pidfile: /usr/local/apache2/httpd.pid
#
### BEGIN INIT INFO
# Provides: httpd
# Required-Start: $local_fs $remote_fs $network $named
# Required-Stop: $local_fs $remote_fs $network
# Should-Start: distcache
# Short-Description: start and stop Apache HTTP Server
# Description: The Apache HTTP Server is an extensible server
#  implementing the current HTTP standards.
### END INIT INFO

# Source function library.
. /etc/rc.d/init.d/functions

if [ -f /etc/sysconfig/httpd ]; then
        . /etc/sysconfig/httpd
fi

# Start httpd in the C locale by default.
HTTPD_LANG=${HTTPD_LANG-"C"}

# This will prevent initlog from swallowing up a pass-phrase prompt if
```

```
# mod_ssl needs a pass-phrase from the user.
INITLOG_ARGS=""

# Set HTTPD=/usr/sbin/httpd.worker in /etc/sysconfig/httpd to use a server
# with the thread-based "worker" MPM; BE WARNED that some modules may not
# work correctly with a thread-based MPM; notably PHP will refuse to start.

# Path to the apachectl script, server binary, and short-form for messages.
apachectl=/usr/local/apache2/bin/apachectl
httpd=${HTTPD-/usr/local/apache2/bin/httpd}
prog=httpd
pidfile=${PIDFILE-/usr/local/apache2/logs/httpd.pid}
lockfile=${LOCKFILE-/var/lock/subsys/httpd}
RETVAL=0

# The semantics of these two functions differ from the way apachectl does
# things -- attempting to start while running is a failure, and shutdown
# when not running is also a failure.  So we just do it the way init
scripts
# are expected to behave here.
start() {
        echo -n $"Starting $prog: "
        LANG=$HTTPD_LANG daemon --pidfile=${pidfile} $httpd $OPTIONS
        RETVAL=$?
        echo
        [ $RETVAL = 0 ] && touch ${lockfile}
        return $RETVAL
}

# When stopping httpd a delay of >10 second is required before SIGKILLing
the
# httpd parent; this gives enough time for the httpd parent to SIGKILL any
# errant children.
stop() {
      echo -n $"Stopping $prog: "
      killproc -p ${pidfile} -d 10 $httpd
      RETVAL=$?
      echo
      [ $RETVAL = 0 ] && rm -f ${lockfile} ${pidfile}
}
reload() {
```

```
    echo -n $"Reloading $prog: "
    if ! LANG=$HTTPD_LANG $httpd $OPTIONS -t >&/dev/null; then
        RETVAL=$?
        echo $"not reloading due to configuration syntax error"
        failure $"not reloading $httpd due to configuration syntax error"
    else
        killproc -p ${pidfile} $httpd -HUP
        RETVAL=$?
    fi
    echo
}

# See how we were called.
case "$1" in
  start)
      start
      ;;
  stop)
      stop
      ;;
  status)
        status -p ${pidfile} $httpd
      RETVAL=$?
      ;;
  restart)
      stop
      start
      ;;
  condrestart|try-restart)
      if status -p ${pidfile} $httpd >&/dev/null; then
            stop
            start
      fi
      ;;
  force-reload|reload)
        reload
      ;;
  graceful|help|configtest|fullstatus)
      $apachectl $@
      RETVAL=$?
      ;;
```

```
  *)
      echo $"Usage: $prog
{start|stop|restart|condrestart|reload|status|fullstatus|graceful|help|conf
igtest}"
      RETVAL=3
esac

exit $RETVAL
```

Figure 137 Apache on Fedora 12: Startup script (automatically generated)

The easiest way to do this is to copy the above shown content to a file named http. Transfer this file to the system via FTP; USB-Stick; Webmail, … .
Open a Terminal and go to the directory of the file. There execute the commands shown below.

```
[mod_rexx@www Downloads]$  su
Password: *****
[mod_rexx@www Downloads]$# mv –f httpd /etc/init.d/httpd
[mod_rexx@www Downloads]$# chmod 755 /etc/init.d/httpd
[mod_rexx@www Downloads]$# chkconfig --level 2345 httpd on
[mod_rexx@www Downloads]$# chkconfig --list | grep httpd
http        0:off 1:off 2:on  3:on  4:on  5:on
```

Each time the system is restarted now Apache will start automatically.

11.3.2 Symbolic linking of the apache2 directory

For better maintenance we will use a symbolic link instead of the apache2 directory. So in future you just need to adopt the link when moving to another version. Fallback is also very easy in this case.

This is done by issuing the following commands in a Terminal session.

```
[mod_rexx@www ~]$ su
Password: *****
[root@www mod_rexx]# cd /usr/local
[root@www local]# mv apache2 apache-2.2.14
[root@www local]# ln –s apache2.2.14/ apache2
[root@www local]# exit
```

11.3.3 Changing htdocs

The default location for htdocs is in the Apache program directory. For improving security (making it more difficult to place malicious code directly in the program directory) you should switch it to another directory.

This is done by issuing the following commands in a Terminal session.

```
[mod_rexx@www ~]$ su
Password: *****
[root@www mod_rexx]# mkdir /var/htdocs
[root@www mod_rexx]# mkdir /var/htdocs/mydomain.com
[root@www mod_rexx]# mkdir /var/htdocs/mydomain.com/pages
[root@www mod_rexx]# mv /usr/local/apache2/htdocs/index.html
/var/htdocs/mydomain.com/pages/index.html
[root@www mod_rexx]# rmdir /usr/local/apache2/htdocs/
[root@www mod_rexx]# exit
```

Open /usr/local/apache2/conf/httpd.conf in your preferred editor

Step 1.)

Search for the following sections:

```
DocumentRoot " /usr/local/apache2/htdocs"
```

Change to the new directory

```
DocumentRoot "/var/htdocs/mydomain.com/pages"
```

Step 2.)

Search for the following sections:

```
#
# This should be changed to whatever you set DocumentRoot to.
#
<Directory " /usr/local/apache2/htdocs">
```

Change the directory directive to

```
<Directory "/var/htdocs/mydomain.com/pages">
```

These changes need a restart of the server.

```
[root@www mod_rexx]# /usr/local/apache2/bin/apachectl restart
```

Check again that it works by going to the webpage http://localhost.

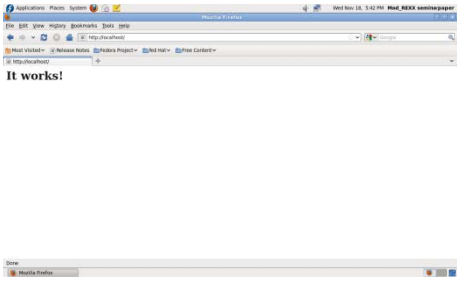

Figure 138 Apache on Fedora 12 – Connection test after reconfiguration

Check that access to the Webserver is possible from a remote machine.

Figure 139 Apache on Fedora 12 – Connection test from remote

12 Appendix F: Installing ooRexx on Windows Server 2008

12.1 Prerequisites

Prior to the installation we need to build and download the actual version of ooRexx via the ooRexx website.

Go to http://build.oorexx.org/build.html select “Windows XP i386 EXE”, enter your e-mail address (used for information) and click on the button [Create ooRexx interpreter].

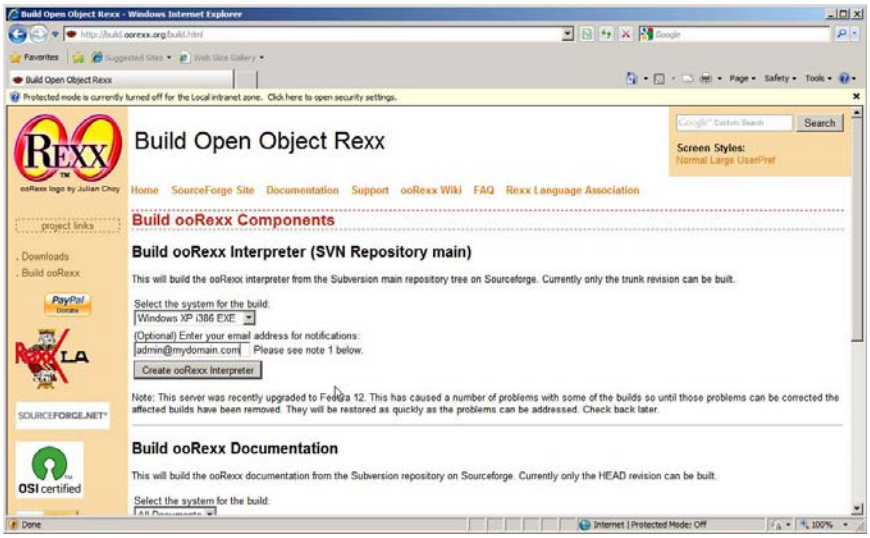

Figure 140 ooRexx on Windows Server 2008 – Create actual build

A view minutes later you will receive an e-mail notification that the build is completed and the appropriate download link.

Open the included link in your favorite webbrowser

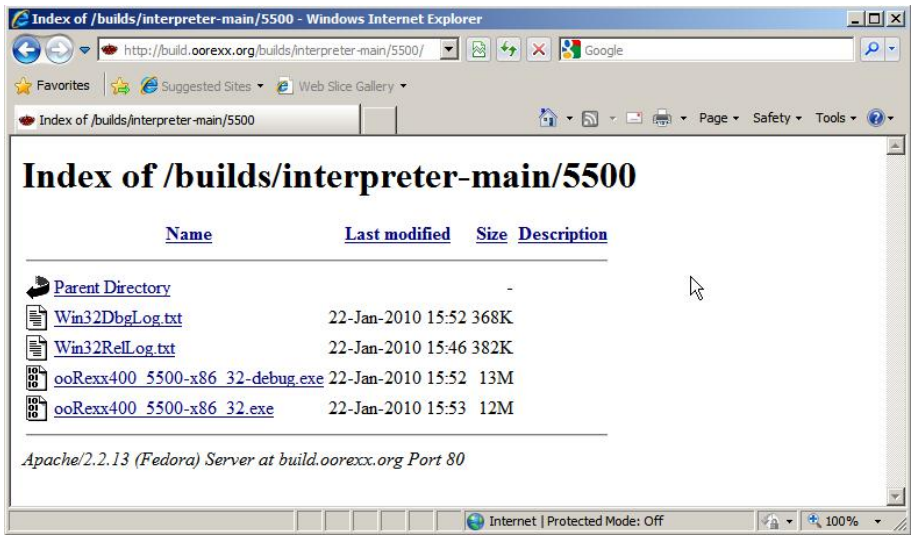

Figure 141 ooRexx on Fedora 12 – Build ready for download

Before continuing we need to check the txt file Win32RelLog.txt for errors occurred during the build process.

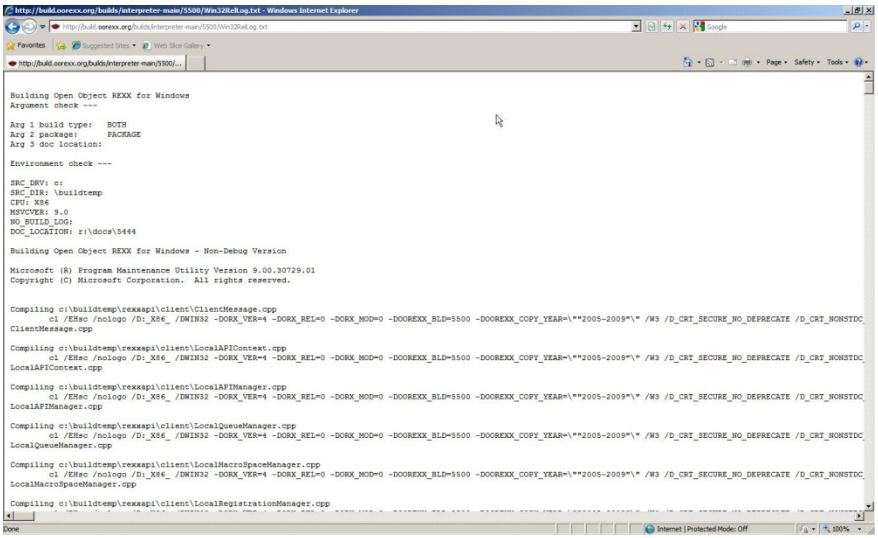

Figure 142 ooRexx on Windows Server 2008 – Check build for any errors

If there is no error message in the txt file you can download the release file (the file which ends not with debug ;-)) to the target system.

12.2 Installation

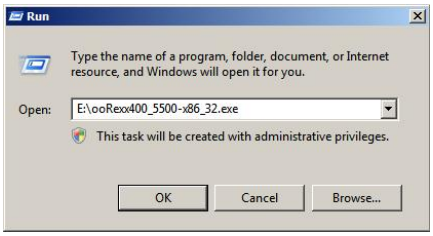

Figure 143 ooRexx on Windows Server 2008 - Start installation

Click on Start / Run and type the filename or browse to its location. Click on [OK] to continue.

Figure 144 ooRexx on Windows Server 2008 - Security warning

By default you will get the security warning shown above. Click on [Run] to continue.

Figure 145 ooRexx on Windows Server 2008 - Setup welcome screen

Click on [Next] to continue.

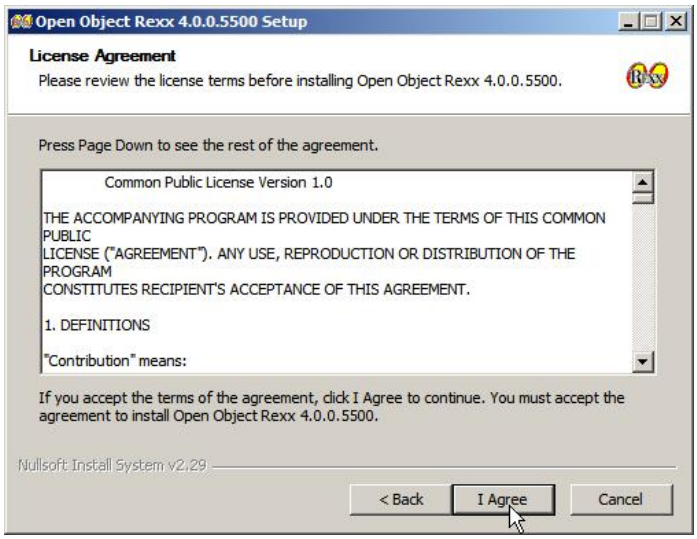

Figure 146 ooRexx on Windows Server 2008 - License agreement

Read through the License Agreement and click on [I agree] to continue as soon as you have understood it.

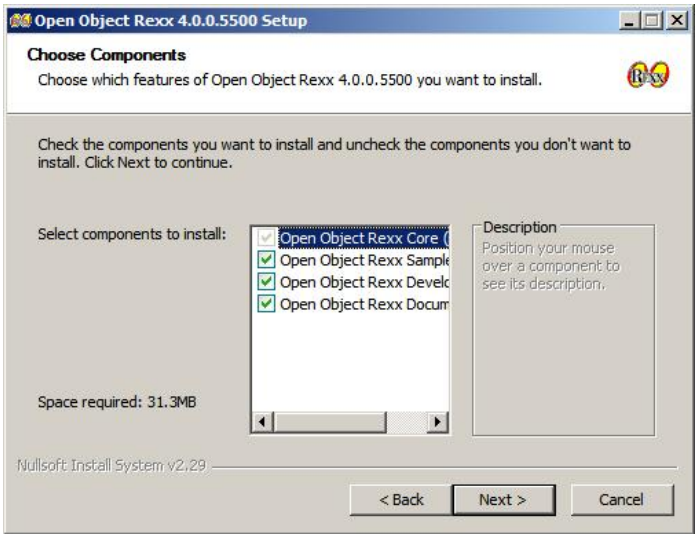

Figure 147 ooRexx on Windows Server 2008 - Choose components

We will use the default and install all components. So just click on the [Next] button to continue.

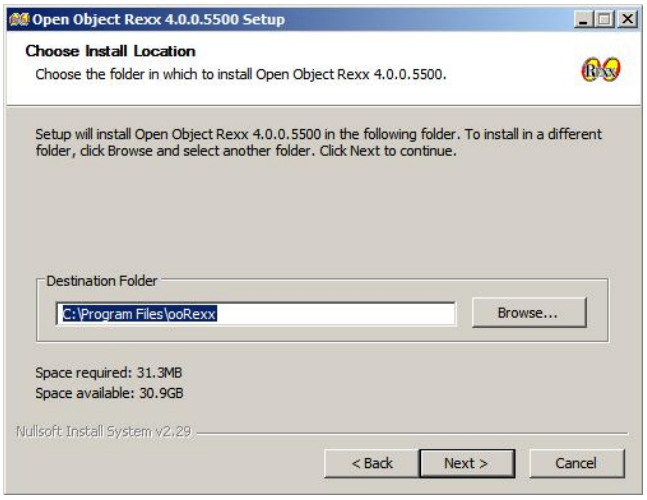

Figure 148 ooRexx on Windows Server 2008 - Install location

Now we choose the location for the installation of ooRexx. Normally the default is o.k. so we click on the [Next] button to continue.

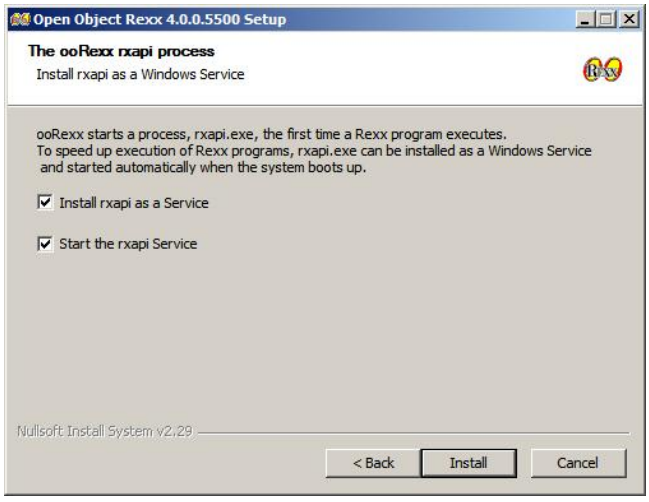

Figure 149 ooRexx on Windows Server 2008 - rxapi process

We will install the rxapi as a service. So leave the default values unchanged and click on the [Install] button to start the installation.

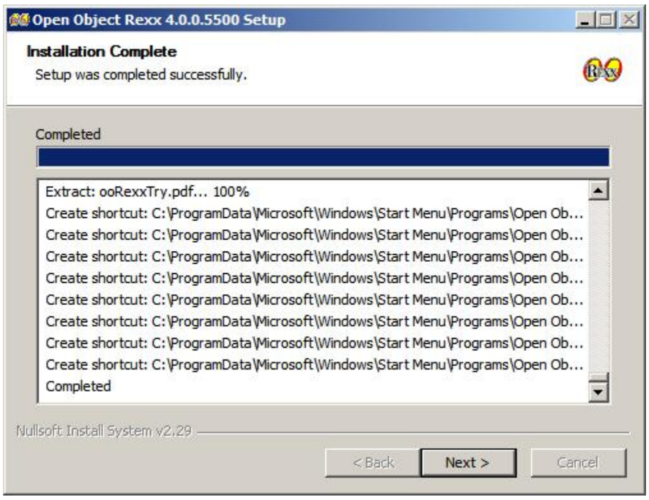

Figure 150 ooRexx on Windows Server 2008 - Installation completed - I

Once the installation is completed you will see the above shown screen. Check for any error message by scrolling through the installation log. Once you are finished click on the [Next] button.

Figure 151 ooRexx on Windows Server 2008 - Installation completed - II

The installation is completed. Click on the [Finish] button to complete the installation.

To check that ooRexx works we will use Rexxtry. Rexxtry is a utility for interactive trying of REXX statements.

Open a DOSbox and type

```
E:\> rexx rexxtry
```

Then issue the simple Rexx command "say "This is a test".

You should see a screen as shown in Figure 152.

Figure 152 ooRexx on Windows Server 2008 - Test successful

After this check the installation is completed.

13 Appendix G: Installing ooRexx on Fedora 12

13.1 Prerequisites

Prior to the installation we need to build and download the actual version of ooRexx via the ooRexx website.

Go to http://build.oorexx.org/build.html select “Fedora 12 i386 rpm” enter your e-mail address (used for information) and click on the button [Create ooRexx interpreter].

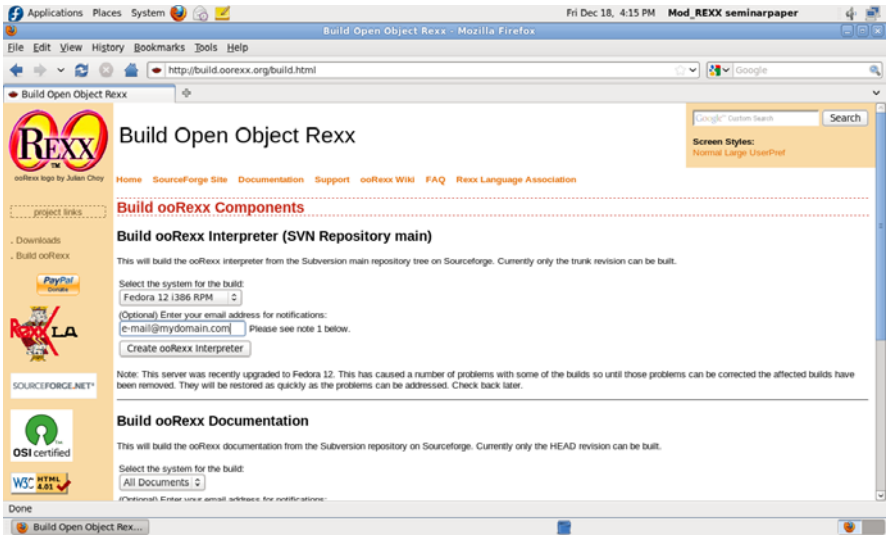

Figure 153 ooRexx on Fedora 12 – Create actual build

A view minutes later you will receive an e-mail notification that the build is completed and the appropriate download link.

Open the included link in your favorite webbrowser

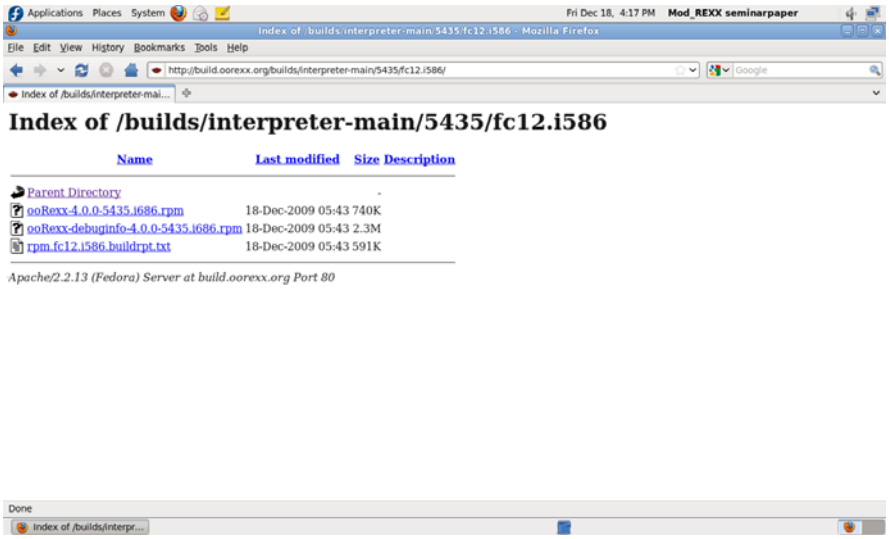

Figure 154 ooRexx on Fedora 12 – Build ready for download

Before continuing we need to check the txt file for errors occurred during the build process

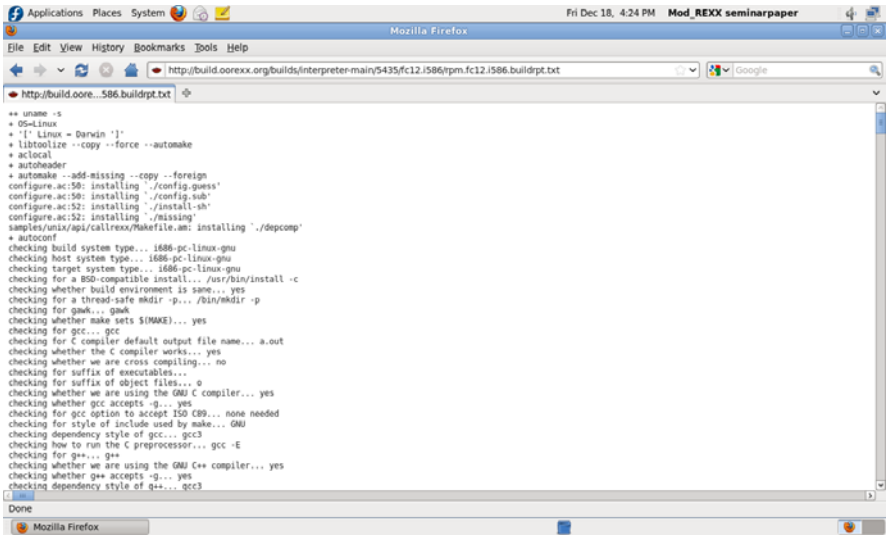

Figure 155 ooRexx on Fedora 12 – Check build for any errors

13.2 Installation

Open a terminal session on the target system and issue the following commands

```
[mod_rexx@www ~]$ su
Password: *****
```

Download the build using the wget command.

```
[root@www mod_rexx]# wget http://build.oorexx.org/builds/interpreter-main/5435/fc12.i586/ooRexx-4.0.0-5435.i686.rpm
```

Install and check the rpm file on the target system.

```
[root@www mod_rexx]# rpm -i ooRexx-4.0.0-5435.i686.rpm
[root@www mod_rexx]# rpm -qa | grep ooRexx
ooRexx-4.0.0-5435.i686
```

Leave the administrator account and check that ooRexx works by using Rexxtry.

```
[root@www mod_rexx]# exit
[mod_rexx@www ~]$ rexx rexxtry
REXX-ooRexx_4.0.0(MT) 6.03 18 Dec 2009
  rexxtry.rex lets you interactively try REXX statements.
    Each string is executed when you hit Enter.
    Enter 'call tell' for a description of the features.
  Go on - try a few...            Enter 'exit' to end.
say "This is a test"
This is a test
  ..... rexxtry.rex on LINUX
exit
[mod_rexx@www ~]$
```

After this check you have completed the installation.