

Hello NetKernel



Welcome. If you are reading this you've probably heard **NetKernel™** mentioned in some XML-related workshop, pretty much like I did back in 2006. Or maybe you have heard about **Resource Oriented Computing (ROC)™** and want to see a practical implementation. Or maybe ... Whatever your reasons for reading it, this book intends to take your hand show you the wonderful world of NetKernel¹. Are you ready ?

Audience

This book is intended for beginning and intermediate ROC-ers². There is a learning curve to ROC and NetKernel and this book will help you along that curve.

Conventions

The book was made with OpenOffice 3.2.1, all formats/fonts mentioned below are available in that editor.

The standard font for this book is **Verdana, 12pt, Black**

Chapter titles are **18pt, Turquoise 6**

Subtitels are **14pt, Turquoise 5**

Headings are **12pt, Bold, Black**

Operators (verbs indicating an action) in the text are underlined, followed by an operand (the thing acted upon) in italic.

Push *this*

Shake *that*

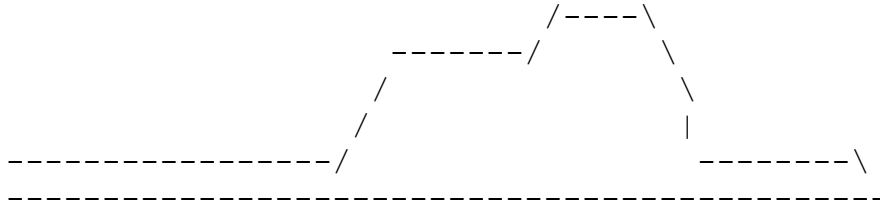
The font for operating system output in this book is **Courier New, 12pt**, instruction lines have a **Grey (10%)** background, the instructions themselves are put in **Bold**.

```
your_user@ubuntumachine:~$ aptitude -vvvv moo
Okay, okay, if I give you an Easter Egg, will you go away?
```

```
your_user@ubuntumachine:~$ aptitude -vvvvv moo
All right, you win.
```

¹ 1060, NetKernel, Resource Oriented Computing, ROC are respectively registered trademark and trademarks of 1060 Research Limited

² Puns with *ROC(ks)* are encouraged in all use of NetKernel. It does make asking for a drink with ice at a NetKernel convention a rather tricky thing to do though ...



```
your_user@ubuntumachine:~$ aptitude -vvvvvv moo
What is it?  It's an elephant being eaten by a snake, of
course.
```

Content

Chapter 1	Hello NetKernel In what remains of this chapter I'll walk you through the NetKernel history, give you a 10000 feet overview of ROC and tell you just why you should bother with NetKernel and ROC.
Chapter 2	Stacking the deck Preparation is half the battle. In this chapter I'll make sure both your environment and your brain are primed to ... <i>roc-and-roll</i> ³ . You'll also make your first NetKernel application.
Chapter 3	Incision ... right here
Chapter 4	...
Chapter 5	...
Chapter 6	...

³ I did warn you about the puns !

Content (continued)

Appendix A	Getting and Installing NetKernel In this appendix I show you how and where to download NetKernel and how to do a basic install. The appendix concludes with doing an Apposite update to make your installation current.
Appendix B	Setting up your own Apposite Repository Some NetKernel instances will not have access to the internet. They will still require updates though. This appendix explains how to set up your own Apposite Repository.
Appendix C	Running NetKernel as a service / daemon started at boottime A basic install is fine for a development workstation, but your production servers will require NetKernel to start at boottime. This appendix shows you how to accomplish that.
Appendix D	Locking down your NetKernel instance (Linux specific) The internet is a wonderful place. Quite a bit jungle-like, now I think of it. In this appendix a couple of simple precautions will teach you how to avoid being eaten while still enjoying the wildlife (NetKernel).
Appendix E	Version Control Yes, you do want version control of your sources ! And no, we are not going to have a discussion about that. This appendix explains how to set one up.

Don't know much about history ...

1060 Research was founded as a spin-out of original research (codenamed *dexter*) undertaken at Hewlett-Packard labs. The 1060 team created and implemented the **Resource Oriented Computing™** model in what you and I know as **NetKernel™**. In the mean time this technology is more than 10 years old and has proven itself in sectors ranging from Telecoms, Insurance, Banking and Military.



At the time of writing this book (September 2010) my source at Hewlett-Packard labs tells me that the current research – something that can be used to automate your garden management (amongst other things) – is codenamed *dharmā*⁴. Now, aren't you glad you just found out that bit of information ? Personally I just wonder if they work alphabetically.



If you haven't lost your sense of humour by now, how is your knowledge of Roman numerals⁵ ? Perfect ? Then try this :
X + M + L = ?

10000 feet view of Resource Oriented Computing

If you want to a more complete (and correct) explanation than you'll get here, have a look at ***Introduction to Resource-Oriented Computing, part I***, an excellent document that you can find under the heading *Technical Whitepapers* on <http://www.1060research.com/netkernel/roc/>. Be warned though that it is a tough read and that intimate knowledge of both *Plato* and *Jack and the beanstalk* are required.

Let me tell you what I think ROC is (after working on and off with it for +/- 4 years) :

1. Everything is a resource. That includes your code, my code, data on the database on my workstation and data encrypted steganographically in an image of the National Art Gallery in Kuala Lumpur.
2. Once you grasped 1. (and that may take a while) let go of worrying about where and how those resources are implemented.

4 <http://en.wikipedia.org/wiki/Dharma>

5 http://en.wikipedia.org/wiki/Roman_numerals

3. Instead, focus on what you want to do with the resources, doing that in small simple services that you can string together to as complex a system as you can imagine (and probably way beyond that).
4. Stand amazed at how your system scales in exactly the same way that the internet scales.

Why bother ?

When I was in school (*Anno Domini Nostris Iesu*⁶ 1992) training to become an IT Bachelor, we got an introduction session on the NeXTSTEP platform. The platform that was going to make us (IT professionals) obsolete within the next couple of years. And after the impressive session most of us (including myself) believed it.

18 years later I'm still in IT. People still use Cobol, PL1 and CICS (as they – well, obviously not the same people ... I hope – did before I was born). NeXTSTEP only survives in the Apple OS somewhere.

If you've been around in IT for a while, you'll have such a story of your own. There's always the **next** best thing that will take away all the pain of software development (the human factor mostly) completely. And managers will always love it.

But if one really has to say what technology **has worked** in the last 15 years, one would have to say ... the internet. It has grown beyond imagination.

ROCTM combines the core ideas of the internet, the core ideas of Unix⁷ and REST⁸ into a new and potent whole :

From Unix - borrow the idea of using simple tools that share a common interoperable data model (e.g., awk, grep, sed, etc.) to build solutions.

From REST - address everything (resources, services and code) with a URI to loosely couple the *internals* of your software making it as flexible as the Web.

NetKernelTM brings all this to an infrastructure near you !

Don't take my word for it. In retrospect I'd have liked to hear the guy showing us NeXTSTEP say that. In fact, our class did meet him again as a mime (doing a robot-impersonation) on some IT gathering later that same year. It turned out that he was out of a job only weeks after giving us the presentation.

As of [Chapter 2](#) each chapter will contain a bit of ROC-talk (indicated by the 'Rolling Stones' image) and a lot of hands-on NetKernel-stuff you **can** try at home. In fact, I'm counting on you to try it at home !

6 In

7 <http://en.wikipedia.org/wiki/Unix>

8 http://en.wikipedia.org/wiki/Representational_State_Transfer

Stacking the deck

Prerequisites

NetKernel

Installed and running. [Appendix A](#) explains how to accomplish that.



A brain⁹

These come in all shapes and sizes. Mine – for example – is not special in any way. Brain-flexibility is required though. ROC is not *difficult*, just *different*. Remember that – contrary to common belief – new braincells can be added and new pathways through your brain can be created. And after you've worked with ROC for a while both those statements will become fact !

A texteditor

There are many good plaintext-editors. And if you've written any code at all you'll probably have a favorite one. I personally use [SciTE](#) because it is lightweight, portable and very customisable. In the documentation that comes with your NetKernel instance (I'll show you where you can find that documentation a bit further on), [IntelliJ IDEA](#) is suggested.

Stick with whatever makes you productive. If your editor has code-highlighting for the most common stuff (xml, html, css, java, javascript, ...), you'll be fine. Only if the default Windows notepad is your top-of-the-line texteditor I would strongly suggest you to follow the above suggestion¹⁰.

⁹ My brain related knowledge comes from <http://pragprog.com/titles/ahptl/pragmatic-thinking-and-learning>, an excellent (very readable, even for the technically inclined) book on the matter.

¹⁰ No, I will not start a flame war over this. If you are happy with notepad, kudos to you !



Whenever you see the 'Rolling Rocks'-sign in the book we are going to have a bit of ROC-talk. Don't worry, I'll try to keep it clear and short at all times. Every sign will also be a link to the next ROC-talk, so if you want to walk through all of them at once, you can (in an electronic version of this book that is) ...

ROC has its own principles and terminology. This ROC-talk will cover the first three (of seven) principles :

1. *A resource is an abstract set of information*
Example : There is a book called 'Hello NetKernel'.
2. *Each resource may be identified by one or more logical identifiers*
Example :
 - The book 'Hello NetKernel' is the book referred to [here](#).
 - The book 'Hello NetKernel' is the book described [here](#).
 - The book 'Hello NetKernel' is the book I am writing right now.
3. *A logical identifier may be resolved within an information context to obtain a physical resource-representation*
Example :
 - As [the odt-file](#) I am editing right now.
 - As [the pdf-file](#) you can read.The verbs (editing and read) form the **information context**, the hyperlinks are **logical identifiers**, what you get on screen (even if you get an error, as you probably will in the case of the file, since that is locally on my machine) is the **physical resource-representation**.

Now, that was not difficult at all, was it ? And we got some terminology done as well, excellent. Do not worry if you can not *place* this information yet, you soon will.

Setup

We are going to dive straight in. Let us first have a look at what we've got and explain a couple of things along the way.

Layout NK4 installation

In what follows I'm labeling the *location* of your NK4 installation as **[install]**. I'm also going to use forward slash to indicate a directory. I know this is different in Windows¹¹, you'll soon notice however that within NetKernel configuration files (regardless of the operating system) the forward slash is used and I did not want a mix. So adjust for Windows where necessary. I'm also going to call your running NetKernel the **[instance]**.



When you look into your [install] directory you'll see these subdirectories :

bin	startup script and startup configuration files
etc	[instance]-wide configuration files
javadoc	generated documentation
lib	[instance]-wide libraries
log	loggings
modules ¹²	the NetKernel batteries (applications and tools)

There are a couple of others. These are volatile directories for caching and for the H2¹³ databasefiles used by the [instance] itself.

[install]/bin

Only the scripts to manually start a NetKernel instance and the configuration files containing the parameters for those scripts can be found here.

[install]/etc

There's some interesting stuff here. The *kernel.properties* file contains the parameters that govern your NetKernel instance. Very interesting stuff, but do not touch unless you have a very good reason (and know what you are doing). Besides, you can change all of these parameters from the Backend GUI.

The *modules.xml* file contains which modules (applications, tools) get loaded (and in which order). You will modify this file. Either manually or through the Backend GUI, but this is where you will add your own modules. In case it was not clear yet, you'll now – by looking at *modules.xml* – realize that NetKernel is build up from modules that run ... in NetKernel.

11 A functional guy in my company put in a request to our Windows System Administration team to adjust all backward slashes to forward slashes. They forwarded the issue to Microsoft Support. No answer has come from Redmond so far.

12 Module is the generic name for application or tool within NetKernel

13 <http://www.h2database.com>

[install]/javadoc

Statements :

1. NetKernel is developed in Java.
2. Java is *one of the languages* you can use to develop modules in NetKernel.

Right, that's out of the way ! I'm by no means a Java-guru (I prefer Python, sorry). The NetKernel developers used Java, the modules that make up NetKernel are Java modules. These modules can generate javadocs that put their documentation in the javadoc-directory. If you develop your own modules for NetKernel in Java, so can they.



People are always giving me a hard time with the "What is NetKernel ?"-question. "Is it an application-server ?", they ask. "Well, no, not really", I answer. "O, then it is an alternative for Java.", they retort and end the discussion ... (which should actually just start then). For me this just proofs some people should stay as far away from IT as possible.

[install]/lib

This directory contains the libraries used to boot the NetKernel instance itself. Note that when I state *[instance]-wide* I mean you'll have a similar directory for each of your own applications as well (*application-wide* in other words). The libraries in *[install]/lib* provide the actual ROC-functionality.

[install]/log

Guess what, this directory contains the loggings of your NetKernel instance. No need to study them here, the Backend GUI contains a very nice logviewer.

[install]/modules

It is very hip to say something is *batteries included*. Python seems to be. Haskell too. I guess Ruby couldn't stay behind. But what does it mean ? What does it mean to say *NetKernel is batteries included* ?

Well, while having the ROC-functionality at your fingertips is surely very nice, it means absolutely nothing to me. I need documentation about it, I need to be able to see it, I need to be able to use it. Those are the batteries ! And that is what the *[install]/modules* directory contains, applications and tools that use the ROC-functionality and open it up to a simple soul like me.

So, if I want to use Python to ROC in NetKernel, I can. Some Saxon for XML processing ? No problem ! Ant ? Sure ! Those and many other things are available ... Some are installed by default, others can be gotten from the repositories.

[install]/project-modules

Is not in the list. Create it now. This is where you'll put your own applications. Using *project-modules* as directory-name is not mandatory (you can choose whatever you like but please do not put blanks in it), I'll stick with *project-modules* for the rest of this book.

```
cd [install]
mkdir project-modules
```

Glossed over

You – readers of this book – are not stupid. You no doubt noticed that I glossed over a couple of things :

1. I did not give my answer to the "What is NetKernel ?"-question.
2. I did not explain what ROC-functionality means.
3. I keep referring to wonderful stuff in the Backend GUI, but I do not show any of it.
4. ...

The reason is that I want you to be able **to do something** (almost there) before I get to page 100 or so. As for the answer to 1., one of the purposes of this book is that you're able to formulate an answer to that yourself. So bear with me, all will be explained.

Hammer Time – your first module

Directory

We are going to create a directory for your module underneath *[install]/project-modules*. You could name this directory anything you like. I'm going to follow the URN¹⁴ method used in the *[install]/modules* directory. Note that for a directory or file, you can not use colons. We use points instead. So these could be possible directory-names for the application :

```
urn.com.colruyt.tutorial.firstmodule-1.0.0
urn.org.tomgeudens.tutorial.firstmodule-1.0.0
```

I also add a version number in the directory name. This too is optional, just makes it easier if I want to have two versions of the same module running side by side.

14 http://en.wikipedia.org/wiki/Uniform_Resource_Name

My employer would love for you all to use the first option whereas I would of course love to become immortal by having a directory with my name in it on your harddrive. But I'll stay modest and go with this directory name :

```
urn.org.netkernelbook.tutorial.firstmodule-1.0.0
```

Create *the directory* for your module in *[install]/project-modules*.

```
cd [install]/project-modules
mkdir urn.org.netkernelbook.tutorial.firstmodule-1.0.0
```



I'm not differentiating between Windows and Linux when stating the commands, remember to use the non-superuser **dexter** though to execute the commands on Linux.

Module definition

Every module has (must have) a *module.xml* file in the root of its directory. Create *[install]/project-modules/urn.org.netkernelbook.tutorial.firstmodule-1.0.0/module.xml* (with the texteditor of your choice).

And here's the content (don't worry about what it all means right now, we'll go into exhaustive detail later on) :

```
<module version="2.0">
  <meta>
    <identity>
      <uri>urn:org:netkernelbook:tutorial:firstmodule</uri>
      <version>1.0.0</version>
    </identity>
    <info>
      <name>firstmodule</name>
      <description>Tutorial to create a first module</description>
    </info>
  </meta>

  <system>
    <dynamic />
  </system>

  <rootspace>

    <fileset>
      <regex>res:/etc/system/SimpleDynamicImportHook.xml</regex>
    </fileset>
```

```
<mapper>

  <config>

    <endpoint>
      <grammar>res:/netkernelbook/firstmodule/hello</grammar>
      <request>
        <identifier>active:dpml</identifier>
        <argument name="operator">
          res:/resources/dpml/hello.dpml
        </argument>
      </request>
    </endpoint>

  </config>

  <space>
    <fileset>
      <private />
      <regex>res:/resources/.*</regex>
    </fileset>
    <fileset>
      <private />
      <regex>res:/etc/.*</regex>
    </fileset>
    <import>
      <private />
      <uri>urn:org:netkernel:lang:dpml</uri>
    </import>
    <import>
      <private />
      <uri>urn:org:netkernel:ext:layer1</uri>
    </import>
  </space>

</mapper>

</rootspace>

</module>
```

Dynamic Import

Again something we'll discuss in detail later, it basically means that we are making our module accessible from *outside* (outside NetKernel itself that is, so we can access it in our webbrowser).

Create directory *[install]/project-modules/urn.org.netkernelbook.tutorial.firstmodule-1.0.0/etc*

Remember that I described *[install]/etc* as the *[instance]*-wide directory for configurations ? Typically every module has its own etc directory with module-wide configurations as well.

Create directory *[install]/project-modules/urn.org.netkernelbook.tutorial.firstmodule-1.0.0/etc/system*

Create *[install]/project-modules/urn.org.netkernelbook.tutorial.firstmodule-1.0.0/etc/system/SimpleDynamicImportHook.xml*

And here's the content:

```
<connection>
  <type>HTTPFulcrum</type>
</connection>
```

The hello endpoint

This is the actual program, written in DPML, NetKernel's own scripting language. Whenever I can use DPML I will do so in this book. That way I avoid discussion over which language is best as well as leveling the playingfield.

Create directory *[install]/project-modules/urn.org.netkernelbook.tutorial.firstmodule-1.0.0/resources*

Create directory *[install]/project-modules/urn.org.netkernelbook.tutorial.firstmodule-1.0.0/resources/dpml*

Create directory *[install]/project-modules/urn.org.netkernelbook.tutorial.firstmodule-1.0.0/resources/html*

Create *[install]/project-modules/urn.org.netkernelbook.tutorial.firstmodule-1.0.0/resources/dpml/hello.dpml*

And here's the content :

```
<sequence >
  <request assignment="response">
    <identifier>res:/resources/html/hello.html</identifier>
  </request>
</sequence>
```

As you can see, the program uses another resource, which we haven't got yet, so *create* `[install]/project-modules/urn.org.netkernelbook.tutorial.firstmodule-1.0.0/resources/html/hello.html`

And here's the content :

```
<html>
  <head>
    <title>NetKernelbook First Module - Hello</title>
  </head>
  <body>
    <h1>NetKernelbook First Module - Hello</h1>
    <p>Your module has been sucessfully generated and deployed.</p>
  </body>
</html>
```

Registering the module

NetKernel has to be made aware of our new module. The place to do that is in `[install]/etc/modules.xml`

Add the following entry just before the `</modules>` endtag in `[install]/etc/modules.xml` (entry is on one line, the split below is due to the limited linesize in this book) :

```
<module runlevel="7">project-
modules/urn.org.netkernelbook.tutorial.firstmodule-1.0.0</module>
```

Testing

NetKernel should have discovered your module now. To make sure it did :

- fire up your favorite webbrowser
- enter <http://localhost:8080/netkernelbook/firstmodule/hello>

Source Version Control

It does seem ridiculous to bring this up here, but this is a good point to set up source version control. Every module you write – even one as small as this one – should have it. If you are not familiar with it, [Appendix E](#) will guide you. If you are ... use it.

Well done

You are probably not very impressed yet with the results. Rome was not build in one chapter either and trust me, we've covered a lot of ground already.

Conclusion

[Chapter 2](#) was aimed at you getting your first module up and running as quickly as possible. [Chapter 3](#) will take the scalpel to the same module and explain it in both ROC-terms and technical-terms.

<Title Chapter 3/>

<chapter 3 text/>



4. *Computation is the reification¹⁵ of a resource to a physical resource-representation*

Example :

- Opening [this odt-file](#) in Open Office
- Requesting [this pdf-file](#) in a webbrowser

The verbs (opening, requesting) are the **computation**, the hyperlinks are **logical identifiers**, what you get on screen (even if you get an error, as you probably will in the case of the file, since that is locally on my machine) is the **physical resource-representation**.

5. *Resource-representations are immutable*

Example :

- I'd be very surprised if I opened the above file and found an explanation of how to grow peas. So would you if you requested the above url.
- As I update this book you will time to time get a newer version (= another resource-representation) of the book when you request the same url.

6. *Transreption is the isomorphic¹⁶ lossless transformation of one physical resource-representation to another*

Example :

- D:\In Progress\hello_netkernel_nk4.**odt**
- http://temp.1060research.com/2010/09/hello_netkernel_nk4.pdf

7. *Computational results are resources and are identified within the address space*

Example : In 'The Complete Works of Tom Geudens' (soon available in a shop near you) you'll find the 'Hello NetKernel' book.

¹⁵ According to my dictionary this means *bringing into being* or *turning concrete*.

¹⁶ Again according to my dictionary this is either a difficult mathematical concept or just means ... *mapping*

Getting and Installing NetKernel

Prerequisites

To run NetKernel you must have a computer and operating system capable of running Java 1.5 or 1.6. NetKernel is platform neutral and has been deployed successfully on Windows 2000, Windows 7, Windows XP, Windows Vista, Windows Server 2003, Apple Mac OS X, Linux (Redhat, Suse, Debian, Ubuntu) and Solaris.

The above comes straight from the install notes. I just want to add that although a JRE (java runtime environment) is sufficient to run NetKernel with all its features, I strongly advise you to install a JDK on machines where you do NetKernel development.

Download

This book deals with the (open source) NetKernel Standard Edition and the versions for that can be found under <http://download.netkernel.org/nkse/> :

- Select the download for the 4.1.x. version.
- Pick a mirror.
- Save the *1060-NetKernel-4.1.x.jar* file to your system.
- While the download is running, read the *install notes*.

Installation

Installation is very easy and pretty much identical on any platform. Below you'll find the transcripts of an installation on **Windows 7** and **Ubuntu 10.04 LTS - the Lucid Lynx**.

Running the downloaded jar – Windows

First position yourself in the directory above the one where you want to install NetKernel (I'm going to install in D:\NK4, so I position in D:, I also put the downloaded jar there for ease of use).

```
C:\Users\your_user>d:
D:\>java -jar 1060-NetKernel-SE-4.1.1.jar
Expanding urn.com.ten60.core.boot-1.13.22
Expanding urn.com.ten60.core.cache.se-1.2.11
Expanding urn.com.ten60.core.layer0-1.31.57
Expanding urn.com.ten60.core.module.standard-1.20.29
Expanding urn.com.ten60.core.netkernel.api-4.1.5
Expanding urn.com.ten60.core.netkernel.impl-4.13.24
I 17:11:40 Kernel
Starting 1060-NetKernel-SE
...
```



```

I 17:11:45 Kernel          NetKernel Ready, accepting requests...
I 17:11:45 ModuleManager System now at RunLevel [2]
*****
* JAR BOOT NOTES
* -----
* NetKernel is now running an HTTP server on port 1060
*
* To start using NetKernel open a web browser
* and go to:  http://localhost:1060/
*****

```

Running the downloaded jar - Ubuntu

Starting the downloaded jar on Linux is exactly the same as on Windows, but we are going to do a bit of preparation in advance, this will make things easier later on.

```

your_user@ubuntumachine:~$ sudo groupadd --gid 1060 dexter
your_user@ubuntumachine:~$ sudo useradd --uid 1060 --gid 1060 -m \
-d /home/dexter -s /bin/bash -c 'NetKernel software' dexter
your_user@ubuntumachine:~$ sudo passwd dexter
your_user@ubuntumachine:~$ sudo mkdir /usr/NK4
your_user@ubuntumachine:~$ sudo chown dexter:dexter /usr/NK4

```

You can of course pick your own groupname, username and directorynames, the rest of this installation procedure will go with the values used above.

Change to the newly created user, position yourself in the directory above the one where you want to install NetKernel and start the downloaded jar.

```

dexter@ubuntumachine:~$ cd /usr
dexter@ubuntumachine:/usr$ java -jar 1060-NetKernel-SE-4.1.1.jar
Expanding urn.com.ten60.core.boot-1.13.22
Expanding urn.com.ten60.core.cache.se-1.2.11
Expanding urn.com.ten60.core.layer0-1.31.57
Expanding urn.com.ten60.core.module.standard-1.20.29
Expanding urn.com.ten60.core.netkernel.api-4.1.5
Expanding urn.com.ten60.core.netkernel.impl-4.13.24
I 17:22:40 Kernel
Starting 1060-NetKernel-SE
...I 17:22:42 Kernel          NetKernel Ready, accepting requests...
I 17:22:42 ModuleManager System now at RunLevel [2]
*****
* JAR BOOT NOTES
* -----
* NetKernel is now running an HTTP server on port 1060
*
* To start using NetKernel open a web browser
* and go to:  http://localhost:1060/
*****

```

Verification – all environments

If all is well, you can now :

- fire up your favorite webbrowser
- enter `http://localhost:1060`

And you should get this screen :



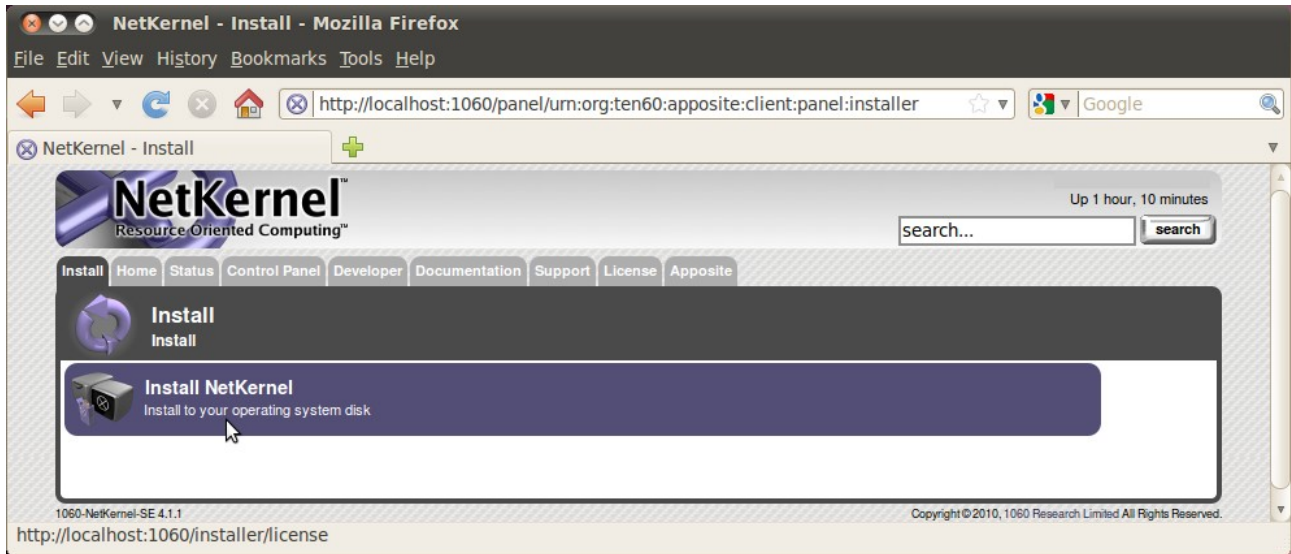
If your machine has internet access you will get news items underneath *NetKernel News*.



The NetKernel interface will perform well on any **modern up-to-date** webbrowser.

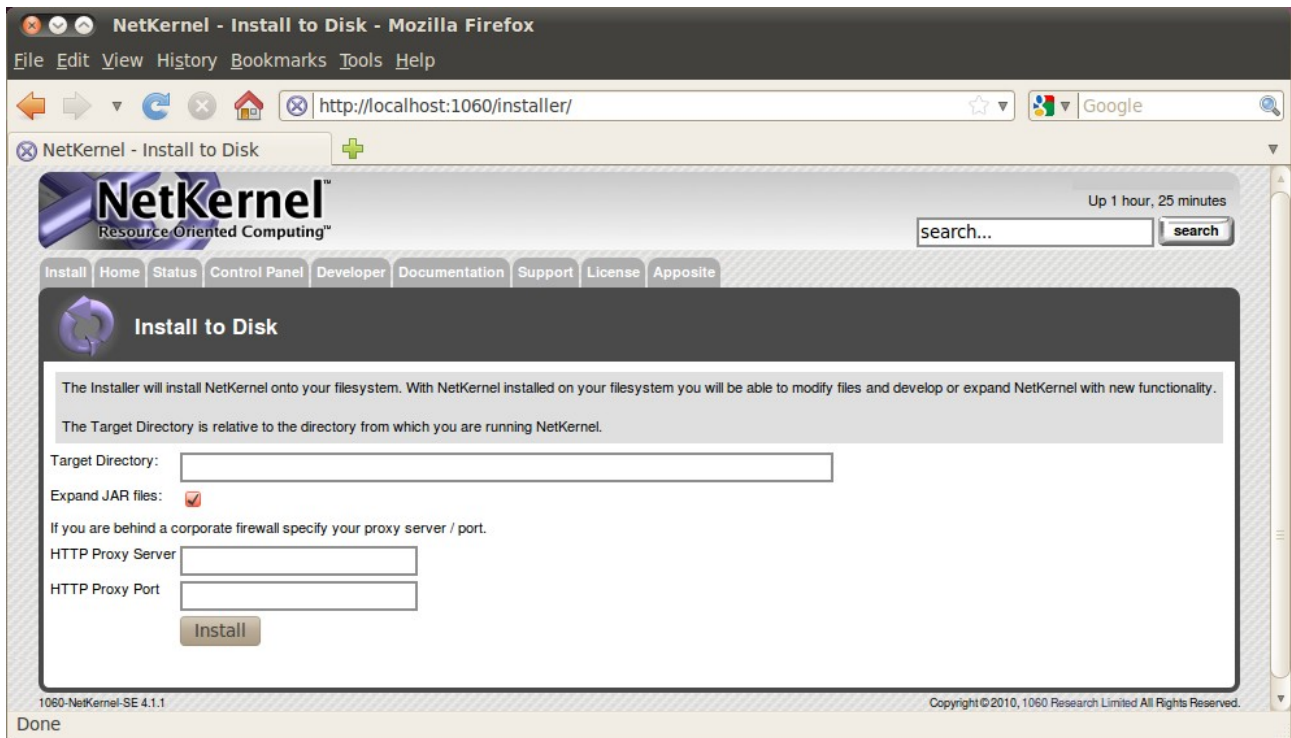
Installation – all environments

By all means, browse through the tabs and check stuff (you probably did not read that *Readme first*, did you ?). When you're ready for the installation to disk, select the *Install* tab.



I do wonder what that option does. Lets find out and press it !

Do read the license you'll see next. This book is concerned only with the (open source) NetKernel Standard Edition. **If you cannot comply with the public license terms you must obtain a commercial license from www.1060research.com.**



All the earlier positioning pays off here, for by entering NK4 in the *Target Directory* field the installation will go where I want it (as well on Windows as on Linux).

Press Install ...

Verification – Windows

The following message will show :

*NetKernel was successfully installed onto your filesystem at **D:\NK4***

Check this visually, there should be five new subdirectories underneath D:\NK4.

```
D:\NK4>dir
Volume in drive D is xxxx
Volume Serial Number is xxxx-xxxx

Directory of D:\NK4

04/09/2010  21:18    <DIR>          .
04/09/2010  21:18    <DIR>          ..
04/09/2010  21:18    <DIR>          bin
04/09/2010  21:18    <DIR>          etc
04/09/2010  21:18    <DIR>          lib
04/09/2010  21:18    <DIR>          log
04/09/2010  21:18    <DIR>          modules
                0 File(s)                0 bytes
                7 Dir(s)   x bytes free
```

Verification – Ubuntu

The following message will show :

*NetKernel was successfully installed onto your filesystem at **/usr/NK4***

Check this visually, there should be five new subdirectories underneath /usr/NK4.

```
dexter@ubuntumachine:/usr/NK4$ ls -la
total 28
drwxr-xr-x  7 dexter dexter 4096 2010-09-04 21:16 .
drwxr-xr-x 14 root    root   4096 2010-09-04 17:20 ..
drwxr-xr-x  2 dexter dexter 4096 2010-09-04 21:16 bin
drwxr-xr-x  4 dexter dexter 4096 2010-09-04 21:16 etc
drwxr-xr-x  4 dexter dexter 4096 2010-09-04 21:16 lib
drwxr-xr-x  2 dexter dexter 4096 2010-09-04 21:16 log
drwxr-xr-x 36 dexter dexter 4096 2010-09-04 21:16 modules
```

Stopping downloaded jar – all environments

You are now almost ready for your first run. Press CTRL-C in the window where you are running the downloaded jar. This will stop the installation-run.

```
...
^CI 16:30:58 Kernel          NetKernel Pausing, flushing pending
requests, new requests queued...
I 16:30:58 HTTPTransport~ Decommissioning HTTP Transport
I 16:30:58 HTTPTransport~ Graceful shutdown {}
```

First run from disk – Windows

```
C:\Users\your_user>d:
D:\>cd NK4
D:\NK4>bin\netkernel.bat
```

```
I 16:44:58 Kernel
Starting 1060-NetKernel-SE
Resource Oriented Computing Platform
Version 4.1.1
...
I 16:45:29 Kernel          NetKernel Ready, accepting requests...
I 16:45:29 ModuleManager System now at RunLevel [7]
I 16:45:29 InitEndpoint  Init completed - system at RunLevel [7]
I 16:45:29 CronTransport Added Job [Apposite Synchronize @ Every
3rd Day] of type [crontab]
```

First run from disk – Ubuntu

```
dexter@ubuntumachine:~$ cd /usr/NK4
dexter@ubuntumachine:~$ bin/netkernel.sh
```

```
I 17:40:15 Kernel
Starting 1060-NetKernel-SE
Resource Oriented Computing Platform
Version 4.1.1
...
I 17:41:08 Kernel          NetKernel Ready, accepting requests...
I 17:41:08 ModuleManager System now at RunLevel [7]
I 17:41:08 InitEndpoint  Init completed - system at RunLevel [7]
I 17:41:08 CronTransport Added Job [Apposite Synchronize @ Every
3rd Day] of type [crontab]
```

Verification – all environments

If all is well, you can now once again :

- fire up your favorite webbrowser
- enter `http://localhost:1060`

And you should get this screen :



The only visible difference with the NetKernel Management Console we saw earlier is that the *Install* tab is no longer there.

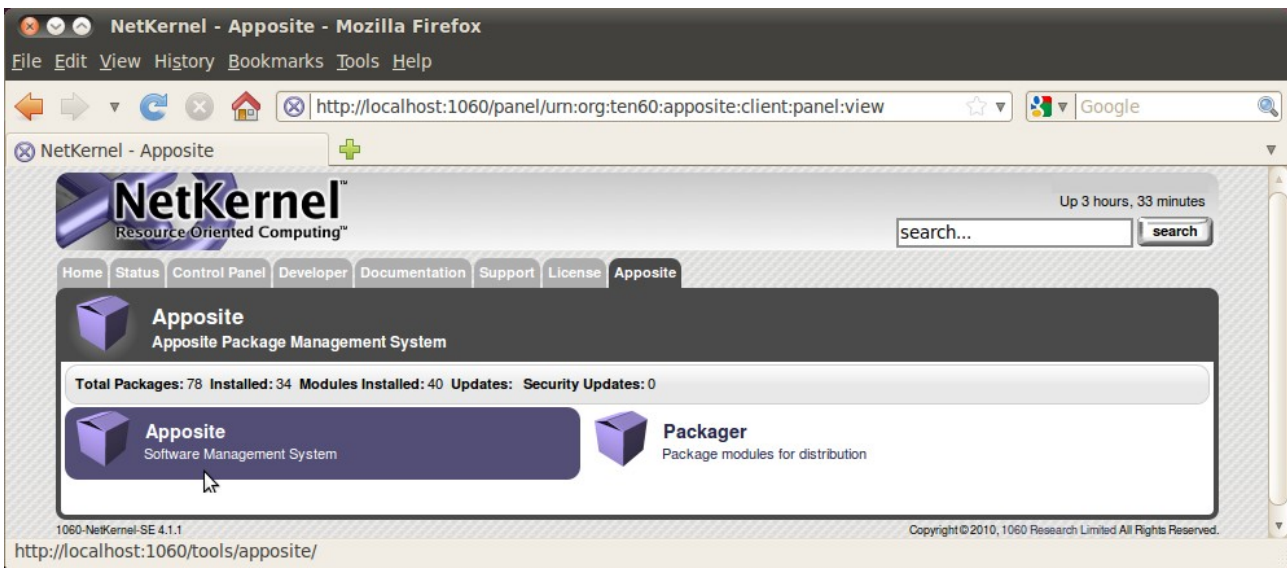
Apposite – all environments

Before you do anything else, you should update the current NetKernel modules, to make sure you have all security and other patches. NetKernel has a *Software Management System* called *Apposite* to take care of this. In fact, *Apposite* itself is managed and updated this way, as is every part of NetKernel.



The default Base URI for the Apposite repository is <http://apposite.netkernel.org/repo/>. If your NetKernel instance does not have access to the internet, you'll not be able to reach this. In that case you should **first** set up your own. [Appendix B](#) explains how to do this. Only then continue with the remainder of [Appendix A](#).

Select the *Apposite* tab.



Packaging is discussed elsewhere in this book. Select and press *Apposite*.



You should see **orange** ! There should be updates available, very likely (you can see this if you scroll down the page) for *Apposite* itself. If there are no updates available at this point ... something went wrong in an earlier step.

The action to take suggests itself rather clearly ... press *Select All Updates*.

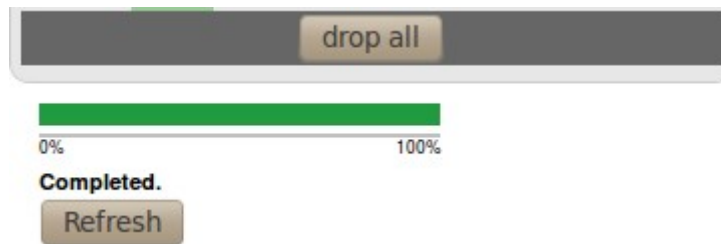
Selections		
	Action	Package
drop	update	apposite - 1.20.1
drop	update	coremeta - 1.3.1
drop	update	database-relational - 1.4.1
drop	update	http-client - 1.4.1
drop	update	http-server - 1.10.1
drop	update	kernel - 1.11.1
drop	update	lang-dpml - 1.10.1
drop	update	lang-groovy - 1.5.1
drop	update	layer0 - 1.26.1
drop	update	layer1 - 1.12.1
drop	update	module-standard - 1.16.1
drop	update	nkse-control-panel - 1.14.1
drop	update	nkse-cron - 1.6.1
drop	update	nkse-dev-tools - 1.16.1
drop	update	nkse-doc-content - 1.17.1
drop	update	nkse-docs - 1.10.1
drop	update	nkse-http-fulcrum-backend - 1.2.1
drop	update	nkse-search - 1.6.1
drop	update	nkse-visualizer - 1.6.1
drop	update	nkse-xunit - 1.4.1
drop	update	pds-core - 1.4.1
drop	update	system-core - 0.11.1
drop	update	wiki-core - 1.4.1
drop	update	xml-core - 1.6.1

drop all

Apply Selections

A *Selections* list will appear. You can see the list I get for version 4.1.1. above, yours may differ. I do know that you immediately want to add other stuff (Python for example) as well, but **don't** ! Take the logical next step, press *Apply Selections*.

Be patient, depending on which repository you use this may take a minute or so. Underneath the Selections list you'll get an update of what's going on. When finished you'll see a *Refresh* button appear there ... like this :



Guess what you have to do next. That's right ... press *Refresh*. If all goes well you should get the *Apposite* screen back, with all updated packages showing their new version number and all **orange** gone !

Conclusion

Installing NetKernel is – considering what you get in return – pretty simple and uniform across platforms. For a production system you might want to run NetKernel as a service or a daemon that gets started at boottime. [Appendix C](#) deals with setting that up.

Setting up your own Apposite Repository

Prerequisites

There is a bit of a 'chicken and the egg'-problem¹⁷ here. The reason you would need your own Apposite Repository is that you do or will not allow your NetKernel instance access to the internet. However, in order to set up your own Apposite Repository, you will need access to the internet. No way around it I'm afraid. It does however not have to be from the machine you run your NetKernel instance on !

I'll discuss a setup via the **rsync**-utility on **Windows 7** and **Ubuntu 10.04 LTS - the Lucid Lynx**. For Ubuntu this utility is present by default, for Windows 7 we'll use the one available in the **Cygwin** package. Don't worry if Cygwin means nothing to you, I'll discuss the setup for that as well.

In fact, that's what I'm going to do first ...



Yes, I do know that there are other rsync ports available for Windows. Feel *free* to use them ... most of them are not (free to use that is). Some of the others are limited to specific usages. Trust me, it will do you no harm to have a Linux-like shell with lots of Linux-utilities available on your Windows machine. You can thank me later !

Preparation

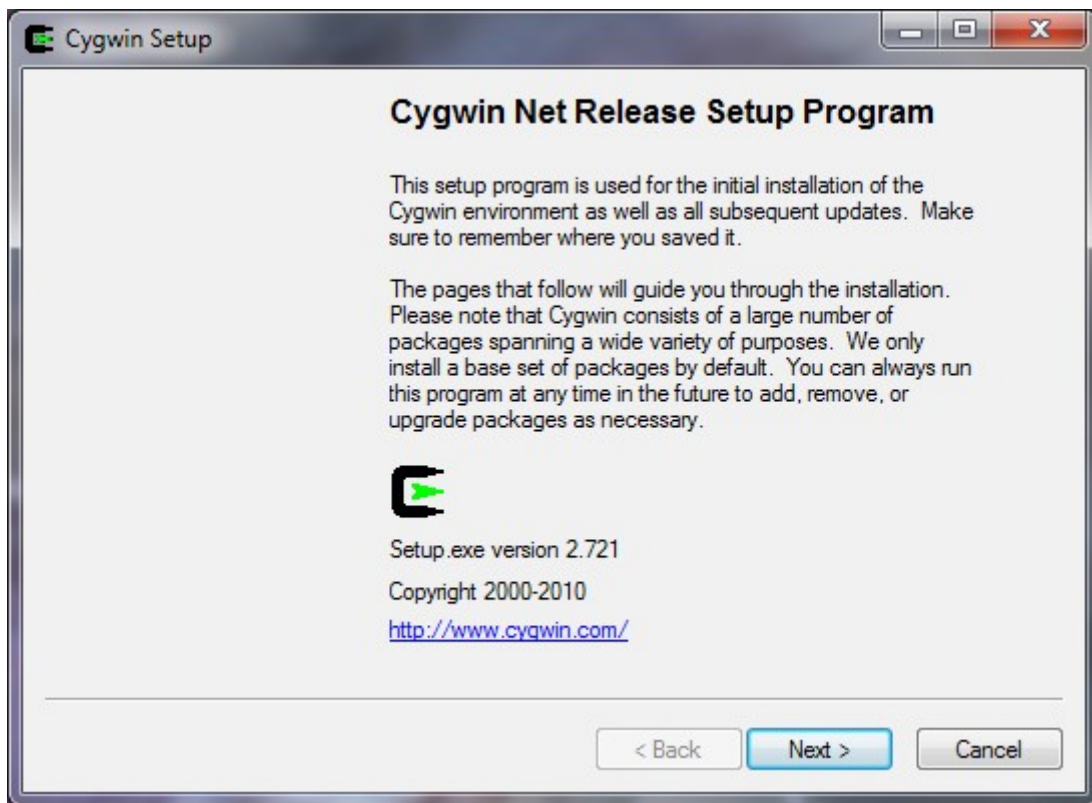
Getting Cygwin – Windows 7 only

You can get the Cygwin setup file at <http://www.cygwin.com/setup.exe>. Download it. Note that not only the first install is done with this file, but also all subsequent updates (or installation of new utilities you may require). So download it to a place where you can find it again (I keep it on my desktop in fact).

¹⁷That one has been solved by science, the chicken came first. Something to do with a certain protein.

Installing Cygwin – Windows 7 only

Start the downloaded *setup.exe*.



Read *the text* (told you about keeping the setup.exe, didn't I ?)

Press *Next*

Select *Install from Internet*

Press *Next*

Enter *the location* and *the users* for Cygwin.

I entered **D:\cygwin** and selected the recommended user option.

Press *Next*.

Enter *the location* you want Cygwin to download its packages to

I entered **D:\cygwin\downloads**.

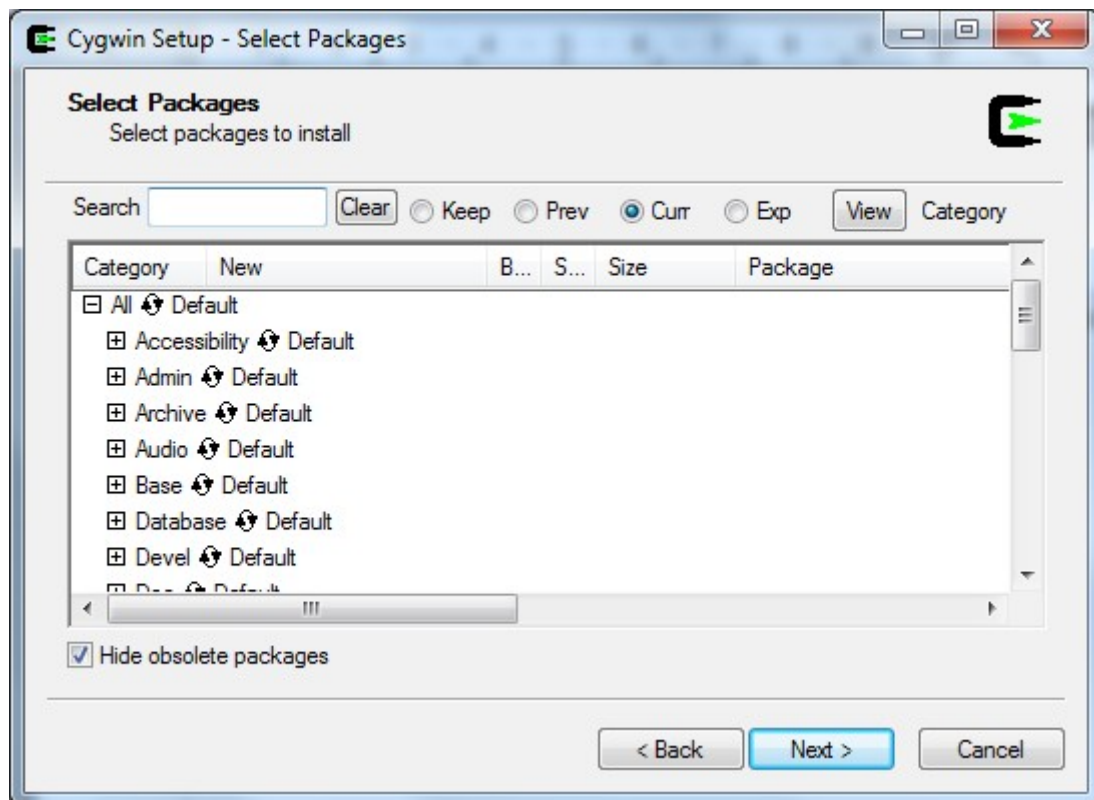
Press *Next*

I use a *Direct Connection* to the Internet, your connection settings may differ ...

Press *Next*

Choose a *mirror* near you.

Press *Next*.



Finally we are getting to the packages (Linux utilities) that are going to be installed. For the most part the defaults are fine, but there are two packages that you want to select extra under the **Net**-heading (expand that heading and click on the skip in front of the packages ... the skip will be replaced by a version number) :

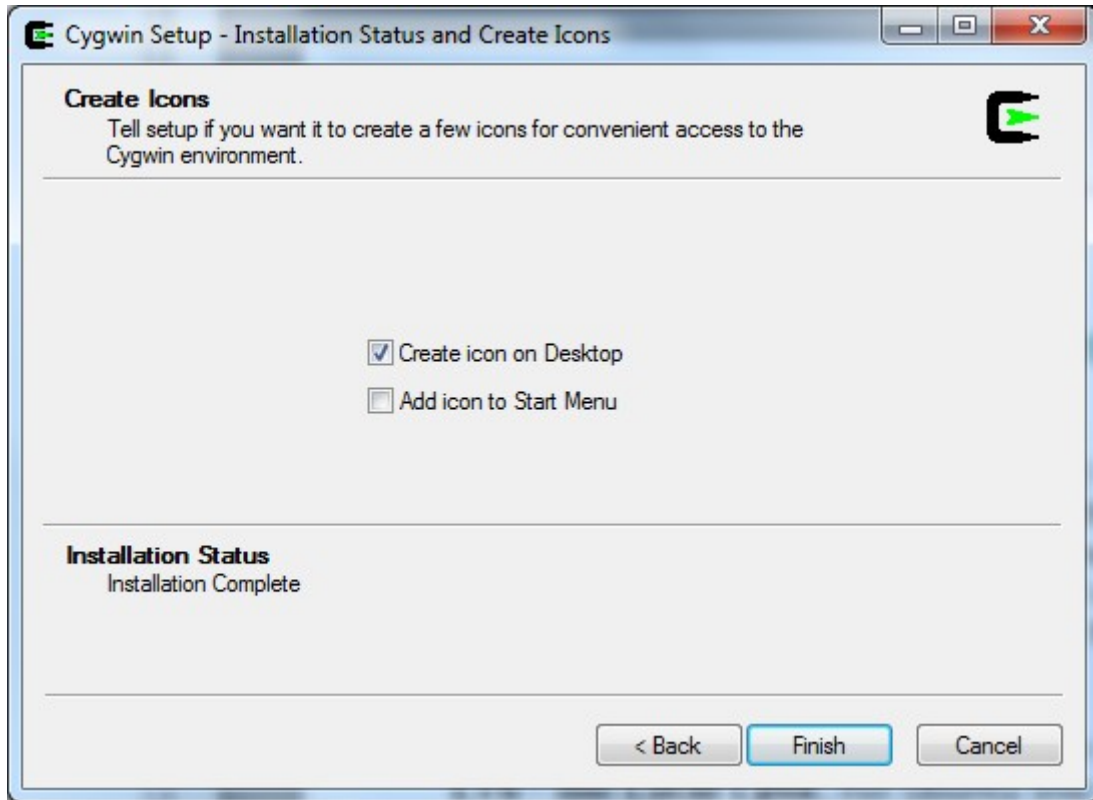
- openssh
- rsync

Press Next

Confirm that you want to select the packages that resolve the dependencies.

Press Next

The installation will now run for a bit ...



Select how you want to be able to reach Cygwin.
Press *Finish*

Congratulations ! You are now the proud owner of a quite decent Linux environment on your Windows machine.

Non-root user – Ubuntu only

If the Linux machine for the Apposite Repository differs from the NetKernel machine, you will benefit from creating the same non-root user we created for the NetKernel machine.

```
your_user@ubuntumachine:~$ sudo groupadd --gid 1060 dexter
your_user@ubuntumachine:~$ sudo useradd --uid 1060 --gid 1060 -m \
-d /home/dexter -s /bin/bash -c 'Apposite Repository' dexter
your_user@ubuntumachine:~$ sudo passwd dexter
```

Synchronization

Creating the repository – Windows 7

Create a directory to hold the repository:

```
C:\Users\your_user>mkdir d:\repo
```

Creating the repository – Ubuntu

Create a directory to hold the repository:

```
your_user@ubuntumachine:~$ sudo mkdir /repo
```

```
your_user@ubuntumachine:~$ sudo chown dexter:dexter /repo
```

Synchronizing the repository – Windows 7

Start your *Cygwin shell* (doubleclick the icon that was created on your desktop).

```
your_user@windowshost ~
```

```
$ mkdir /repo
```

```
your_user@windowshost ~
```

```
$ mount d:/repo /repo
```

```
your_user@windowshost ~
```

```
$ rsync -rv rsync://apposite.netkernel.org/download/repo/ /repo/
```



If you are using a firewall (you should) it will now ask you if **rsync** is allowed access to the Internet. Grant that access. A second later it will come back to ask if rsync may act as a server. It may.

The synchronization will take a while, the repository is (September 2010) about 250Mb.

```
...
```

```
packages/Y/
```

```
packages/Z/
```

```
sent 10335 bytes  received 265302027 bytes  304082.94 bytes/sec
```

```
total size is 265230673  speedup is 1.00
```

```
your_user@windowshost ~
```

```
$ umount /repo
```

```
your_user@windowshost ~
```

```
$ rmdir /repo
```

```
your_user@windowshost ~
```

```
$ exit
```

Synchronizing the repository – Ubuntu

Log on to the system as the non-root user we created earlier.

```
dexter@ubuntumachine:~$ rsync -rv \  
rsync://apposite.netkernel.org/download/repo/ /repo/
```

The synchronization will take a while, the repository is (September 2010) about 250Mb.

...

packages/Y/

packages/Z/

```
sent 10335 bytes  received 265302027 bytes  318311.17 bytes/sec  
total size is 265230673  speedup is 1.00
```

Verification – Windows 7

You should see two directories in the repository.

```
C:\Users\your_user>dir d:\repo  
Volume in drive D is xxxx  
Volume Serial Number is xxxx-xxxx
```

Directory of d:\repo

```
09/09/2010  21:17    <DIR>          .  
09/09/2010  21:17    <DIR>          ..  
09/09/2010  21:17    <DIR>          netkernel  
09/09/2010  21:17    <DIR>          packages  
                0 File(s)                0 bytes  
                4 Dir(s)  x bytes free
```

Verification – Ubuntu

You should see two directories in the repository.

```
dexter@ubuntumachine:~$ ls -la /repo  
total 16  
drwxr-xr-x  4 dexter dexter 4096 2010-09-09 21:31 .  
drwxr-xr-x 23 root    root  4096 2010-09-09 21:06 ..  
drwxr-xr-x  3 dexter dexter 4096 2010-09-09 21:31 netkernel  
drwxr-xr-x 38 dexter dexter 4096 2010-09-09 21:31 packages
```

Use

There are several ways you can go about this. You can **serve** the Apposite Repository to the NetKernel instance(s) through a webserver. In Chapter 3 we do exactly that when I show how you can use NetKernel as a webserver.

Another option is to **map** the Apposite Repository over your internal network to the NetKernel instance(s). Windows has several options for just that and with **Samba**¹⁸ you can easily map a Linux directory to a Windows machine (the other way around remains a tricky thing though).

For the rest of this Appendix I assume that you have **manually copied** the Apposite Repository to the machine that is running the NetKernel instance.

Whatever option you take, remember to frequently resynchronize with the central repository on the internet ! Once a month for example will not hurt at all.



Automate this task or have it automated. It is all very well to be closed off from the *evil* internet, and no, you do not always need the latest and the greatest, but you do need security patches and the occasional new functionality. If you have to do it manually you'll forget after a while.

Activating your personal Apposite Repository – Windows 7

So, the assumptions are as follows :

- You are running the NetKernel instance on this machine.
- You've copied the synchronized Apposite Repository to this machine, in my case that is to **D:\repo**.

Navigate your browser to the NetKernel Apposite screen (<http://localhost:1060> and so on, remember ?).

Press the Admin button.

Edit the Base URI of the repository so the screen looks like this :

Name	Base URI	Path	PublicKey	Trusted	Sets
NetKernel.org	file:///D:/repo/	1060-NetKernel-SE/4.1.1/		<input checked="" type="checkbox"/>	main universe multiverse

¹⁸<http://www.samba.org>

Yes, the Base URI is now **file:///D:/repo/**. Windows loves slashes !
Make sure to test the connection !

Activating your personal Aposite Repository – Ubuntu

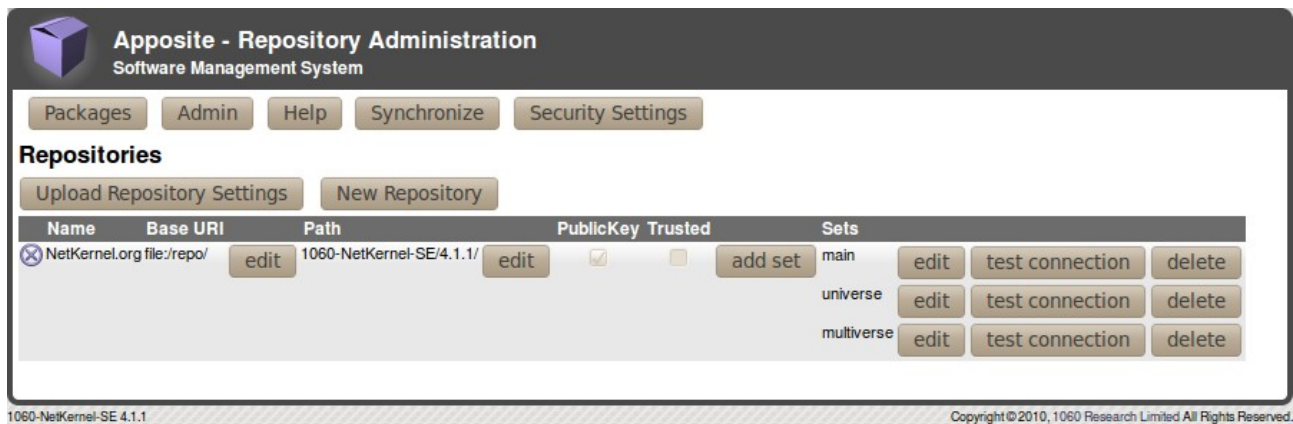
So, the assumptions are as follows :

- You are running the NetKernel instance on this machine.
- You've copied the synchronized Aposite Repository to this machine, in my case that is to **/repo**.

Navigate your browser to the NetKernel Aposite screen (http://localhost:1060 and so on, remember ?).

Press the Admin button.

Edit the Base URI of the repository so the screen looks like this :



The screenshot displays the 'Aposite - Repository Administration' web interface. At the top, there are navigation buttons: 'Packages', 'Admin', 'Help', 'Synchronize', and 'Security Settings'. Below this is the 'Repositories' section, which includes 'Upload Repository Settings' and 'New Repository' buttons. A table lists the repository details:

Name	Base URI	Path	PublicKey	Trusted	Sets
NetKernel.org file:/repo/	file:/repo/	1060-NetKernel-SE/4.1.1/	<input checked="" type="checkbox"/>	<input type="checkbox"/>	main universe multiverse

Each repository entry has an 'edit' button next to the Base URI and Path, and an 'add set' button next to the Trusted checkbox. The Sets column lists 'main', 'universe', and 'multiverse', each with its own 'edit', 'test connection', and 'delete' buttons. The footer of the interface shows '1060-NetKernel-SE 4.1.1' on the left and 'Copyright © 2010, 1060 Research Limited All Rights Reserved.' on the right.

So, the Base URI is now **file:/repo/**.
Make sure to test the connection !

Conclusion

Setting up your own **Aposite Repository** is not hard at all. I would even dare to say it is an advisable thing to do :

- Your security team – if you have one – will be pleased.
- If you have multiple NetKernel instances running, your updates will be much swifter from a local (local as in 'on your local network') repository.

However, I would also advise to :

- Automate the synchronization with the official repository.
- Update your instances frequently.

One more thing to note ... the packages in a local repository are no less secure than those on the official repository. I quote :

The public apposite repository only has signed official releases of 1060 authorized packages. Both the individual packages and the complete repository metadata are signed. When you have a local copy inside your firewall the NKSE apposite client still performs full repository and package authentication and verification before permitting anything from the the mirror to be installed. So even though the library is local you can still treat it as the authentic trustworthy source of NKSE libraries and updates.

And that's all I have to say about that.

Running NetKernel as a service / daemon started at boottime

Cover Story

For servers this is expected behavior, but imagine this : "You get up in the morning, you boot your desktop/laptop at home¹⁹ and NetKernel is automatically started. Bliss."

As you probably know, attaining the state of bliss is (in most religions) not one of the lighter matters. However, Peter Rodgers²⁰ has provided all that you require to run NetKernel as a service/daemon (on your server or on your home desktop) right here :

<http://www.netkernel.org/forum/topic/649/1>

If you feel comfortable with the instructions mentioned in that post, by all means follow them and ignore the rest of this Appendix. I'm going to do almost the same (you'll have to read further to see where I take a different approach ... muhahaha²¹).

Note that I only vouch for the platforms that I tested the procedures on. At this time those are **Windows 7** and **Ubuntu 10.04 LTS - the Lucid Lynx**. That means they will **probably** work on most current Windows platforms and Linux platforms. If you have been able to test on another platform (and want an honorable mention), let me know !

Preparation

Getting and installing YAJSW – Windows 7 only

No, I'm not using the Tanuki wrapper which you can find at <http://wrapper.tanukisoftware.com> and that is used in Peters post. Do not get me wrong, Tanuki is the authority on this task and their Community Edition is excellent. I was however very sorry to read this :

Release footnotes:

**1: 64-bit Windows versions of the Java Service Wrapper are not currently being made available in the Community Edition.*

¹⁹If this is indeed the first thing you do after getting up (it is for me), get a life ... and tell me where you found one !

²⁰<http://www.1060research.com/company/management/>, Peter is the guy at *the top* ...

²¹My 'evil laugh' imitation.

Very strange because the Linux 64-bit version is available. However, it does disqualify Tanuki for the time being. Somebody *used the force and read the Tanuki-source* however and the result (**Yet Another Java Service Wrapper**) can be found at <http://sourceforge.net/projects/yajsw/files/>.

Download the zip file from there.

At this moment that is **yajsw-beta-10.3.zip**, your version may be different.

Unpack the zip file into a directory of your choice, in my case **D:\yajsw**.

Modifying YAJSW – Windows 7 only

In order to make YAJSW a bit more flexible we are going to alter the bat-files a bit. You can find those in the ... you guessed it ... bat-subdirectory, in my case

D:\yajsw\bat.

Edit setenv.bat

```
...
rem configuration file used by all bat files
IF [%1]==[] (
set conf_file="%wrapper_home%/conf/wrapper.conf"
) ELSE (
set conf_file="%1"
)
...
```

Edit installService.bat

```
call setenv.bat %1
%wrapper_bat% -i %conf_file%
pause
```

Edit startService.bat

```
call setenv.bat %1
%wrapper_bat% -t %conf_file%
pause
```

Edit stopService.bat

```
call setenv.bat %1
%wrapper_bat% -p %conf_file%
pause
```

Edit queryService.bat

```
call setenv.bat %1
%wrapper_bat% -q %conf_file%
pause
```

Edit *uninstallService.bat*

```
call setenv.bat %1
%wrapper_bat% -r %conf_file%
pause
```

Getting the netkerneld script – Ubuntu only

You can use the location mentioned in the post for this, but actually you have the script on your system already. That is, if you kept the original jar-file you used to install NetKernel with.

Did you know that any jar-file – **1060-NetKernel-SE-4.1.1.jar** in my case – is actually a zip-file ? Rename a copy of the .jar-file to a .zip-file and unzip it (in a save location, you do not want to mess up your NetKernel install in this way).

The **netkerneld** script can be found in the bin-directory in that save location.

Use

Making the wrapper configuration for YAJSW – Windows 7 only

The YAJSW scripts expect a configuration file. You can put this configuration file (now that we've made the scripts somewhat more flexible) in NetKernels main etc-directory. In my case the file is **D:\NK4\etc\wrapper.netkernel.conf** and it contains the following (some lines are split due to the limitation of the linesize, but these are all single line definitions !) :

```
wrapper.java.command=java
wrapper.working.dir=D:\\\\NK4
wrapper.java.app.mainclass=BootLoader
wrapper.console.loglevel=INFO
wrapper.console.title=NetKernel
wrapper.ntservice.name=NetKernel
wrapper.ntservice.displayname=NetKernel
wrapper.ntservice.description=NetKernel
wrapper.java.classpath.1 =
D:\\\\NK4\\lib\\urn.com.ten60.core.boot-1.13.22.jar
wrapper.app.parameter.1 = D:\\\\NK4
wrapper.java.additional.1 = -Xmx128m
wrapper.java.additional.2 = -Xms128m
wrapper.java.additional.2 = -XX:SoftRefLRUPolicyMSPerMB=100
wrapper.java.additional.3 =
-Djava.protocol.handler.pkgs=org.ten60.netkernel.protocolhandler
wrapper.java.additional.4 =
-Dsun.net.client.defaultReadTimeout=20000
wrapper.java.additional.5 =
-Dsun.net.client.defaultConnectTimeout=20000
wrapper.logfile =
```

You should of course alter this according to your installation !

This is little more than you'll also find in the *normal* **netkernel.bat** startscript. If you're interested in knowing each and every possible parameter, check out : <http://yajsw.sourceforge.net/YAJSW Configuration Parameters.html>



I'm not quite happy with so many hardcoded parameters. I envision one extra step to generate this configuration file based on the available NK4 installation information.

Install and start the NetKernel service – Windows 7 only

You are now ready to install the service. There is however one more thing to take into account. Exactly that in fact. The scripts have to run with *Administrator* permissions. On Windows 7 there are several ways to accomplish this (I changed the property of the Command Prompt program to "Run As Administrator" ... now I just have to remember to turn it off again), just Google for a solution and you'll come up with one that suits you.

Navigate to the YAJSW bat-directory

```
C:\Users\your_user>d:  
D:\>cd yajsw\bat  
D:\yajsw>
```

Install the NetKernel service

```
D:\yajsw\bat>installService.bat D:\NK4\etc\wrapper.netkernel.conf
```

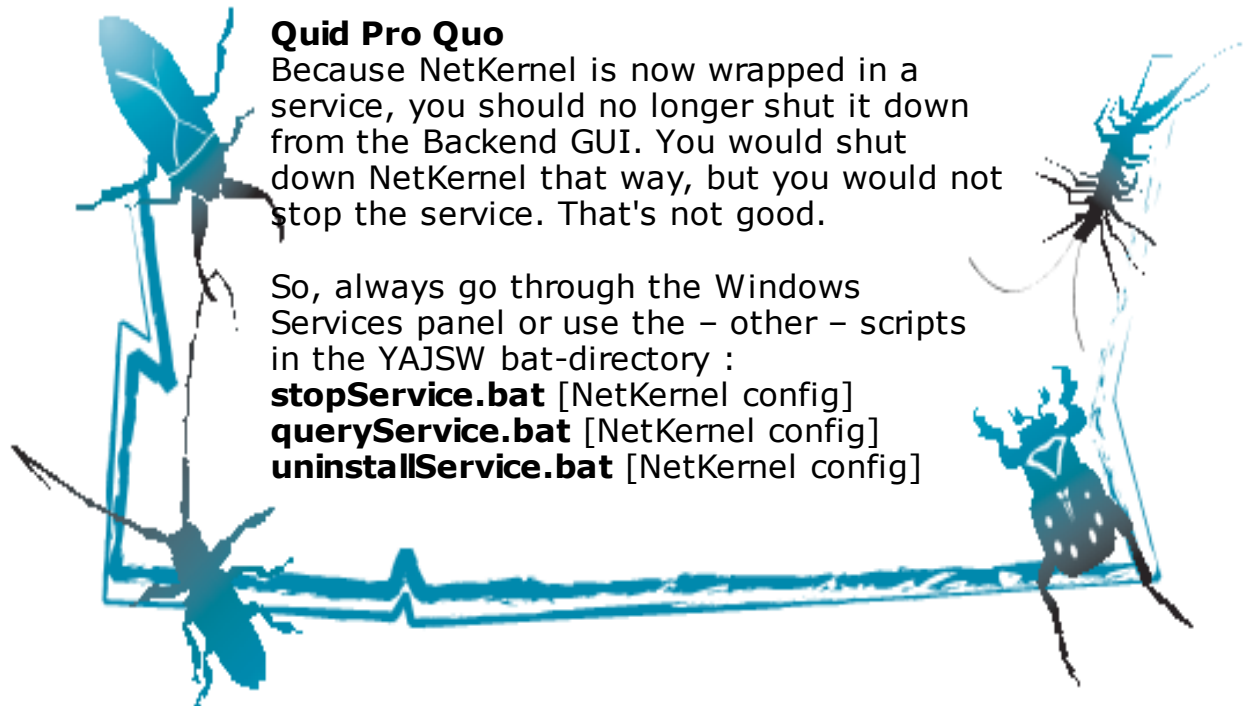
```
D:\yajsw\bat>call setenv.bat D:\NK4\etc\wrapper.netkernel.conf  
"java" -Xmx30m -jar "d:\yajsw\bat\..\wrapper.jar" -i  
"D:\NK4\etc\wrapper.netkernel.conf"  
17-Sep-2010 15:26:21 org.apache.commons.vfs.VfsLog info  
INFO: Using "C:\Users\YourUser\AppData\Local\Temp\vfs_cache" as  
temporary files store.  
Service NetKernel installed
```

That was easy, was it not ? You still have to start the service though ...

```
D:\yajsw\bat>startService.bat D:\NK4\etc\wrapper.netkernel.conf
```

```
D:\yajsw\bat>call setenv.bat D:\NK4\etc\wrapper.netkernel.conf  
"java" -Xmx30m -jar "d:\yajsw\bat\..\wrapper.jar" -t  
"D:\NK4\etc\wrapper.netkernel.conf"  
17-Sep-2010 15:39:34 org.apache.commons.vfs.VfsLog info  
INFO: Using "C:\Users\YourUser\AppData\Local\Temp\vfs_cache" as  
temporary files store.  
service started
```

And thats it. The NetKernel service will be restarted when you reboot your machine.



Quid Pro Quo

Because NetKernel is now wrapped in a service, you should no longer shut it down from the Backend GUI. You would shut down NetKernel that way, but you would not stop the service. That's not good.

So, always go through the Windows Services panel or use the - other - scripts in the YAJSW bat-directory :

stopService.bat [NetKernel config]

queryService.bat [NetKernel config]

uninstallService.bat [NetKernel config]

Modify and install the netkerneld script – Ubuntu only

Note

- You'll require superuser abilities to do the installation of the script.
- The wget-utility is required. Most Linux/Unix systems have this, if yours does not, install it first

Before we do anything else with the script, we're first going to make a few small adjustments.

Verify that *the parameters* at the top are correct

```
HOMEDIR=/usr/NK4
```

```
NK_USER=dexter
```

```
APPNAME=NetKernel
```

```
#User Editable variables
```

```
STARTGREPPID="ten60.pid=1"           #Must match the ten60.pid value  
set in the netkernel.sh script
```

```
BACKENDPORT="1060"                   #HTTP port of backend fulcrum
```

Comment out *this instruction* (just put a # in front of it) in the script

```
chown -R $NK_USER $HOMEDIR
```

It is an understandable but dangerous instruction. I personally prefer the daemon to fail at startup because a certain file *suddenly and magically* (yeah, right) has the wrong ownership. Maybe you prefer a flawless startup.

Now, we are ready to do the install. Move the netkerneld script from wherever you got it to /etc/init.d/ directory.

Change the ownership and the permissions of the script

```
your_user@ubuntumachine:~$ cd /etc/init.d
your_user@ubuntumachine:/etc/init.d$ sudo chown root:root
netkerneld
your_user@ubuntumachine:/etc/init.d$ sudo chmod 755 netkerneld
```

Verify if the daemon works

```
your_user@ubuntumachine:/etc/init.d$ sudo service netkerneld
Usage: /etc/init.d/netkerneld {start|stop|reboot|restart|status|
kill}
```

The next step is to link the daemon into the startup and shutdown of the machine. On Ubuntu this is very easy :

```
your_user@ubuntumachine:/etc/init.d$ sudo update-rc.d netkerneld
defaults
```

```
update-rc.d: warning: /etc/init.d/netkerneld missing LSB
information
update-rc.d: see <http://wiki.debian.org/LSBInitScripts>
Adding system startup for /etc/init.d/netkerneld ...
/etc/rc0.d/K20netkerneld -> ../init.d/netkerneld
/etc/rc1.d/K20netkerneld -> ../init.d/netkerneld
/etc/rc6.d/K20netkerneld -> ../init.d/netkerneld
/etc/rc2.d/S20netkerneld -> ../init.d/netkerneld
/etc/rc3.d/S20netkerneld -> ../init.d/netkerneld
/etc/rc4.d/S20netkerneld -> ../init.d/netkerneld
/etc/rc5.d/S20netkerneld -> ../init.d/netkerneld
```

You may safely ignore the LSB warning (unless you are a Debian purist, in which case you'll have rewritten the netkerneld script completely already).

Most Linux/Unix systems have these *runlevels*, so even if they do not have the **update-rc.d** tool, you can make the links yourself, for example :

```
your_user@ubuntumachine:~$ ln -sf /etc/init.d/netkerneld
/etc/rc0.d/K20netkerneld
```

All that remains now is to start the daemon. You can do that by rebooting the machine or by starting the service yourself :

```
your_user@ubuntumachine:/etc/init.d$ sudo service netkerneld start
```


Verification – all environments

If all is well, you can now :

- fire up your favorite webbrowser
- enter `http://localhost:1060`

It pays to check that all works as expected. So do reboot your machines to see NetKernel rise and shine again !

Locking down your NetKernel instance

Prerequisites

<to be continued/>

Version Control

Prerequisites

<to be continued/>