HiSPARC Installer

Implementation Notes

Robert Hart
Nikhef, Amsterdam, Netherlands

NIKHEF

Contents

1 Introduction

The HiSPARC installer is generated by and based upon the "**Nullsoft Scriptable Install System**" (NSIS). It consists of three installers:

1. **adminUpdater**: the *administrator* installer which requires administrator rights.
2. **userUnpacker**: the *user* part of the HiSPARC software. No administrator rights are required.
3. **hisparcInstaller**: the overall or *main* installer of HiSPARC. Besides its own part, it includes the `adminUpdater` and `userUnpacker` as well. They are an integral part of it. Executing the `hisparcInstaller` requires administrator rights.

Both the `adminUpdater` and `userUnpacker` have their own version number (independent of each other). Furthermore the installer has an independent release number. The version number of the main installer is a combination of these numbers. Suppose version *X* for the `adminUpdater`, version *Y* for the `userUnpacker` and release number *Z*, then the following executables (installers) are generated (compiled):

1. **adminUpdater_v*X*.exe**
2. **userUnpacker_v*Y*.exe**
3. **hisparcInstaller_v*X.Y.X*.exe**

The reason to have three separate installers is to distribute updates in some kind of automatic way. The *updater* process (as part of the HiSPARC system) checks at regular intervals the central installer repository for newer versions of the `adminUpdater` and `userUnpacker`. If a more recent version is found, it is copied and in case of the `userUnpacker` automatically executed. In case of the `adminUpdater` a message box is presented indicating an `admin` update is available. In the latter case an administrator should execute manually the latest version of the `adminUpdater`.
Initially, one starts with the combined or main `hisparcInstaller`. Practically you do not need the other two.

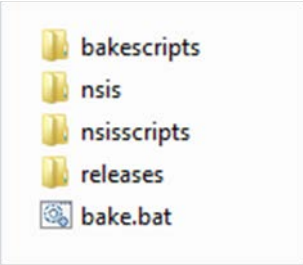
2 Creating the Installer

The installer is created by a set of batch, Python and NSIS scripts.
The following list of remarks explains how it is implemented.

1. The folder containing the HiSPARC software[1] contains the folder: `bake`.
   NB: It also contains (and requires) the folders `admin`, `persistent` and `user`!

[1] See Appendix 6.1: Setting up a HiSPARC Development Area

2. The `bake` folder itself contains the following files/folders:



The `releases` folder does not exist initially. It is created as soon as an installer is made. The `bake.bat` batch file is the starting point of creating the installer. Double-clicking it will start the creation procedure. The three installers are stored in the `releases` folder. The `bakescripts` folder contains a couple of Python sources to initiate and support the creation. The `nsis` folder contains version 2.46 of NSIS and the `nsisscripts` folder the NSIS sources of HiSPARC.

3. The contents of `bake.bat` is shown in Script 2-1. It simply calls a Python script: `bakescripts\bake.py` which continues the execution. It uses the Python executable from the `user` part of the installer.
NB: `"%~dp0"` denotes the current directory of the script itself.

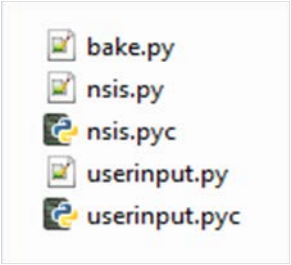
```
@echo off

:: call the python bake script
call "%~dp0..\user\python\python.exe" "bakescripts\bake.py"

pause
```

Script 2-1: bake.bat

4. The `bakescripts` folder contains the following files:



The `bake.py` scripts is called by the `bake.bat` script. It uses two supporting Python scripts: `nsis.py` and `userinput.py` (see Appendix 6.2). The two `.pyc` files (compiled Python) are irrelevant. They are generated only once.

5. The contents of the Python script `bake.py` is shown in Script 2-2. When executed it checks whether the `releases` folder already exists. If not, it is created. Then it continues asking which version number of the administrator installer and user installer are to be created. Finally it asks for a release number. The Python script `userinput.py` is used to get these numbers. If the corresponding version already exists (inside the `releases` folder), then the generation of it is skipped. The main installer is always generated (compiled).

```
# bake.py
# Create the HiSPARC installer.

import os
import sys

from datetime  import datetime
from userinput import *
from nsis      import *

#files created will always be put in the "\bake\releases" directory
RELEASE_DIRECTORY = "./releases"

input = userInput()
nsiHandling = nsiHandling()

print "\nWelcome to the HiSPARC bake script!\n"
adminVersion  = input.getVersion("administrator")
userVersion   = input.getVersion("user")
releaseNumber = input.getVersion("release")

tNow = datetime.today()
releaseDate = "%d%02d%02d_%02d%02d%02d" % (tNow.year, tNow.month, tNow.day, tNow.hour, tNow.minute,
tNow.second)

#check if the RELEASE_DIRECTORY exists, if not create it
if not os.access(RELEASE_DIRECTORY, os.F_OK):
    os.makedirs(RELEASE_DIRECTORY)

#compile the administrator software first
if os.path.exists("%s/adminUpdater_v%s.exe" % (RELEASE_DIRECTORY, adminVersion)):
    print "Administrator installer already exists, not creating a new one!"
else:
    try:
        nsiHandling.compileNSI("./nsisscripts/adminupdater/admininstaller.nsi",
        ["ADMIN_VERSION=%s" % adminVersion])
    except:
        print "ERROR: Compilation could not be finished!"
        sys.exit

#compile the user software
if os.path.exists("%s/userUnpacker_v%s.exe" % (RELEASE_DIRECTORY, userVersion)):
    print "User unpacker already exists, not creating a new one!"
else:
    try:
        nsiHandling.compileNSI("./nsisscripts/userunpacker/userunpacker.nsi",
        ["USER_VERSION=%s" % userVersion])
    except:
        print "ERROR: Compilation could not be finished!"
        sys.exit

#compile the main installer
try:
    nsiHandling.compileNSI("./nsisscripts/maininstaller/hisparcinstaller.nsi",
    ["ADMIN_VERSION=%s" % adminVersion]+["USER_VERSION=%s" % userVersion]+["RELEASE=%s" %
releaseNumber]+["RELEASE_DATE=%s" % releaseDate])
except:
    print "ERROR: Compilation could not be finished!"
    sys.exit

print "\nFinished compilation of version %s.%s.%s.\n" % (adminVersion, userVersion, releaseNumber)
```

Script 2-2: bake.py

6. In case all installers are created the `bake.py` calls the `nsis` builder three times with the following arguments (example: administrator version = 6, user version = 11, release number = 4):

   - adminUpdater

Prog	`nsis\makensis.exe`
Arg1	`/V1`
Arg2	`/DADMIN_VERSION=6`
Arg3	`nsisscripts\adminupdater\admininstaller.nsi`

   - userUnpacker

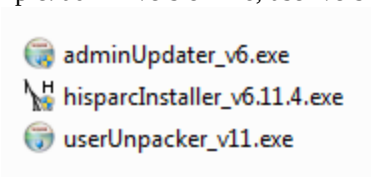
Prog	`nsis\makensis.exe`
Arg1	`/V1`
Arg2	`/DUSER_VERSION=11`
Arg3	`nsisscripts\userunpacker\userunpacker.nsi`

   - hisparcInstaller

Prog	`nsis\makensis.exe`
Arg1	`/V1`
Arg2	`/DADMIN_VERSION=6`
Arg3	`/DUSER_VERSION=11`
Arg4	`/DRELEASE=4`
Arg5	`/DRELEASE_DATE=20130129_125929`
Arg6	`nsisscripts\maininstaller\hisparcinstaller.nsi`

   NB: Arg1: /V1: verbosity level 1 = errors only.

7. The results are placed in the `releases` folder and after completion it should contain three files (example: admin version = 6, user version = 11, release number = 4):



   The `hisparcInstaller_v6.11.4.exe` is the final installer and is the combined `adminUpdater` and `userUnpacker` installer, including the `main` part. The `admin` part and `user` part are not created, if they already exist in this folder. For a clean build, delete all files in this folder first.

8. Figure 2.1 shows the complete output of the execution of `bake.bat`. In this case the creation of the HiSPARC installer version 6.11.4. A complete rebuilt takes about 15 minutes.

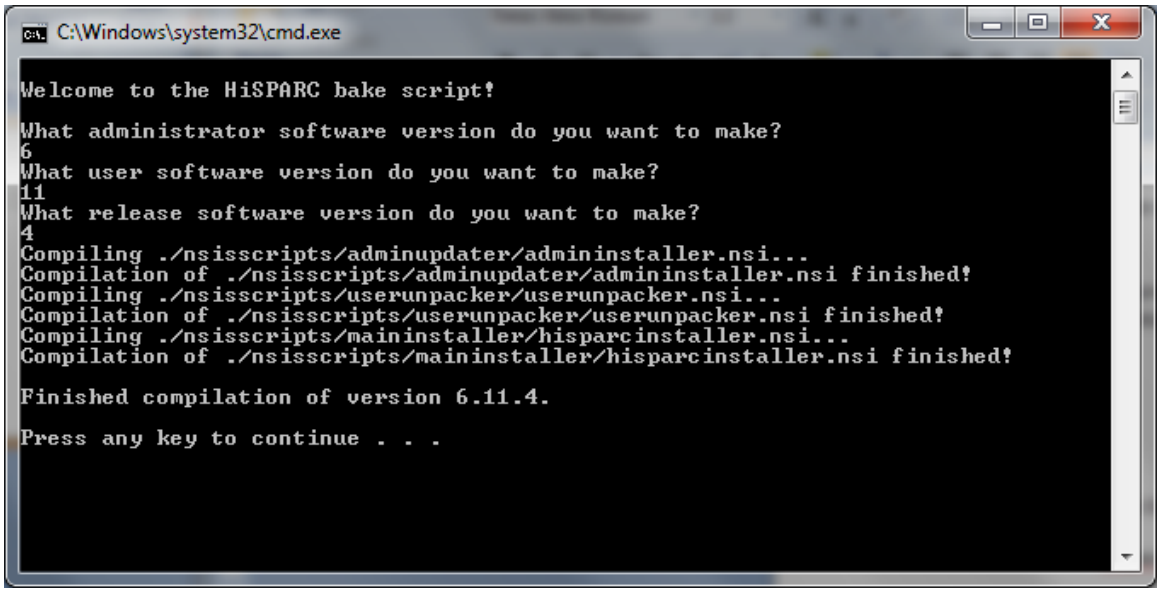


Figure 2.1: Output HiSPARC installer v6.11.4

9. Each installer has its own folder with NSIS sources. Below the contents of the folders are shown:

hisparcInstaller	*adminUpdater*	*userUnpacker*
header.bmp	admininstaller.nsi	interface.nsh
hisparc.ico	firewall.nsh	userunpacker.nsi
hisparcinstaller.nsi	install.nsh	
interface2.nsh	interface.nsh	
startinstall.ini	uninstall.nsh	
uninstaller.nsh	variables.nsh	
userinput.nsh		
userinput1.ini		
userinput2.ini		
userinput3.ini		
variables.nsh		
welcome.bmp		

All installers have their own `.nsi` file (the default extension for NSIS), which is called by the `bake.py` script to start the creation. The `.nsi` files contain the `.onInit` function, by which the NSIS generation is started. The supporting `.nsh` files (NSIS header files) take care of a specific part of the installer. The main installer folder contains also a couple of picture files (`.ico` and `.bmp`) for decoration purposes. Furthermore it contains a couple of `.ini` files which define the input boxes of the main variables.
The three installers share a common include file `hs_def.nsh` containing definitions used by all of them, mainly registry definitions. This file resides one level up inside the `nsisscripts` folder.

3 Executing the HiSPARC installer

Each installer has its own set of tasks and for that it uses its own specific folder, which is created inside the HiSPARC installation folder. Table 3.1 shows which folder is used by which installer.

Installer	Nickname	Folder
hisparcInstaller	*main*	`persistent`
adminUpdater	*administrator*	`admin`
userUnpacker	*user*	`user`

Table 3.1: Installer folder

The following chapters explain in more detail what each installer does.

3.1 hisparcInstaller

The hisparcInstaller (or *main* installer) takes care of the complete installation of HiSPARC. For that purpose it contains the `adminUpdater` and `userUnpacker` as well. The following list shows what it does:

1. The `.onInit` function inside the file `hisparcinstaller.nsi` starts with some checks:

    - Platform check: the installation is aborted if the PC is of type Windows ME, 95 or 98. At the moment only WXP and W7 PCs are supported.
    - Administrator check: the installation is aborted if the user has no administrator privileges.
    - Win32 check: if it is not a 32-bit processor (probably a 64-bit processor), a warning is given and the user is asked to continue or not.
    - Check registry if HiSPARC is already installed. If installed, the user is asked to continue or not.

2. The NSIS modern user interface 2.0 is started (from `interface2.nsh`). Its task is to get and set the main parameters. The first mandatory parameter is the installation path:

    ```
    C:\Program Files\HiSPARC
    ```

    NB: On a 64-bit machine the installation path is: `C:\Program Files (x86)\HiSPARC`

3. The other selectable parameters are:
    - Station number (mandatory)
    - Station password (mandatory)
    - Security certificate (mandatory)
    - Web address local database (optional)
    - Connected detectors:
        - HiSPARC detector (seems obvious)
        - Weather station
        - Magnetic field detector
        - Lightning detector

4. The folder `hisparc` is created inside the installation path. This folder, known as `$HisparcDir`, is the home area for the HiSPARC system.

5. The entire `persistent` folder, as found in the HiSPARC base area besides the `bake` folder, is copied into the `$HisparcDir` folder.

6. The folder `downloads` is created inside the folder `$HisparcDir\persistent`. The *adminUpdater* and the *userUnpacker* (with the proper version numbers) are copied into this folder.

7. The folder `$HisparcDir\persistent\uninstallers` is created. Each installer has its own uninstaller. They reside in this folder.

8. The security certificate file (a zip file) is copied into the folder:
   `$HisparcDir\persistent\configuration`

9. The configuration file `$HisparcDir\persistent\configuration\config.ini` is updated with the following parameters:

   - Station number
   - Station password
   - Security certificate (name of location of the zip file)
   - Web address local database
   - Connected detectors: HiSPARC, weather, magnetic field and lightning

10. The HiSPARC system has its own registry entry which, at this point, is created with the following key and name:

`HKLM`	`SOFTWARE\HiSPARC`

    One of the entries, which is created and written, is the `Path` variable, containing `$HisparcDir.` Others are the version numbers and the parameters mentioned in the previous item. The main variables are stored both in the configuration file and registry. See also Appendix 6.3.
    NB: The version number of the adminUpdater and userUnpacker are written by the installer itself, not by the main installer!

11. The environment (system) variable `HISPARC_ROOT` is created and set to `$HispardDir`.

12. The main installer now calls the **adminUpdater** (chapter 3.2), as found in
    `$HisparcDir\persistent\downloads`

13. The main installer now calls the **userUnpacker** (chapter 3.3), also found in
    `$HisparcDir\persistent\downloads`

14. Two local user accounts are created (the passwords are not shown):

    - **hisparc**: the user logged on when HiSPARC is running. No administrator privileges, the account never expires and its password cannot be changed.
    - **admhisparc**: user with administrator privileges. The account never expires. This user may occasionally be necessary when updates have to be installed which require administrator rights.

    Finally all accounts are set so that their passwords never expire.

15. Enable automatic logon at reboot. The following registry variables (Table 3.2) with key `HKLM` and path '`SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon`' are set:

Variable	Value
DefaultUserName	`hisparc`
DefaultPassword	`!Usr4hisp`
AutoAdminLogon	`1`
ForceAdminLogon	`0`
DefaultDomainName	`$CpuName`

Table 3.2: Autologon variables

The `$CpuName` variable contains the name of the computer. If the name could not be determined, then the value "`this computer`" is used.

16. Several HiSPARC commands are added and grouped as shortcut to the '`All Programs`' list. They reside inside the group HiSPARC, which contains the following shortcuts (Table 3.3):

Shortcut link	Program
StartHiSPARCSoftware	`$HisparcDir\persistent\startstopbatch\StartUserMode.bat`
Expert/DSPMon	`$HisparcDir\user\dspmon\DSPMon.exe`
Expert/HiSPARC_DAQ	`$HisparcDir\user\hisparcdaq\run_hisparcdaq.bat`
Expert/Uninstall	`$HisparcDir\persistent\uninstallers\mainuninst.exe`
Status/LocalDiagnosticTool	`$HisparcDir\user\diagnostictool\run_diagnostictool.bat`
Status/Registry	`$HisparcDir\persistent\startstopbatch\HiSPARC_Registry.exe`
Status/RunStatus	`$HisparcDir\persistent\startstopbatch\RunStatus.bat`
HiSPARC_ROOT	`%windir%\explorer.exe`

Table 3.3: 'All Programs' HiSPARC commands

The StartHiSPARCSoftware link starts the entire HiSPARC system, including the LabVIEW application (detector and/or weather), the MySQL daemon, the HiSPARC *monitor* and *updater*.

17. The following shortcut is added to the general **Startup** folder:

StartHiSPARCSoftware	`$HisparcDir\persistent\startstopbatch\StartUp.bat`

As a result the `StartUp.bat` file is executed each time a user logs on. If the user is not `hisparc` it does nothing. If the user is `hisparc` it executes the StartHiSPARCSoftware link as found in Table 3.3, hereby executing the HiSPARC system. In combination with the automatic logon of `hisparc` at startup, it somehow guarantees that the entire HiSPARC system is always running.

18. The *main* uninstaller, `mainuninst.exe`, is created and stored into the folder: `$HisparcDir\persistent\uninstallers.` The `Uninstall` shortcut from the HiSPARC command group points to this uninstaller. Furthermore a separate registry entry is created for the HiSPARC system with the following key and name:

`HKLM`	`SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\HiSPARC`

Several variables are created and written. As a result, the HiSPARC system is added to the list of removable packages/programs found in the Windows Control Panel.

19. The security and properties of the entire HiSPARC area (i.e. `$HisparcDir`) is set to "**FullAccess**" for all users.

20. The installation is finished. The user is asked to restart the computer or not. If yes, the following command is executed: "`shutdown –r –f –t 0`". Due to the automatic logon facility and the `Startup.bat` program at startup, the HiSPARC system will be up and running within a few minutes.

3.2 adminUpdater

The **adminUpdater** is responsible for the installation of several packages, such as:

- OpenVPN
- TightVNC
- NAGIOS
- ODBC
- LabVIEW (Runtime Engines only)
- FTDI drivers

OpenVPN, TightVNC and NAGIOS are handled as *service* (in the Windows context).
The `adminUpdater` is part of the `hisparcInstaller` and executed by it. However, the `adminUpdater` can also be executed separately. If installed separately, it uses a simplified GUI, without any decoration. The following sequence is carried out by the `adminUpdater`:

1. The `.onInit` function of the `adminUpdater`, located in the `adminInstaller.nsi` file, starts with several checks:

   - Administrator check: the installation is aborted if the user has no administrator privileges.
   - Registry check: the installation is aborted if the HiSPARC registry entry path does not exist or is empty (`HKLM, SOFTWARE\HiSPARC\Path`). If it exists the variable `$HisparcDir` is assigned to it.
   - HiSPARC folder check: the installation is aborted if the folder `$HisparcDir` does not exist.
   - Configuration file check: the installation is aborted if the configuration file `$HisparcDir\persistent\configuration\config.ini` does not exist.
   - Certificate check: from the configuration file the location of the security certificate file is determined. The file name is assigned to the variable `$CertZip`. The installation is aborted if this (zip) file does not exist.

2. The variable `$AdminDir` is declared and assigned to: `$HisparcDir\admin`.

3. A check is performed whether the PC has a 32-bit processor. If yes, the variable `$OpenVpnDir` is assigned to `$AdminDir\openvpn32`. Otherwise it is assumed that the PC has a 64-bit processor and the variable `$OpenVpnDir` is assigned to `$AdminDir\openvpn64`.

4. The entire `admin` folder, as found in the HiSPARC base area besides the `bake` folder, is copied into the folder `$HisparcDir`.

5. The variable `$AdminDir` is declared and assigned to: `$HisparcDir\admin`. The installation is aborted if one of the following folders or files does not exist:

   - `$AdminDir`
   - `$OpenVpnDir\bin\tapinstall.exe`
   - `$OpenVpnDir\bin\openvpnserv.exe`
   - `$AdminDir\tightvnc\tvnserver.exe`
   - `$AdminDir\nsclientpp\NSClient++.exe`
   - `$AdminDir\odbcconnector\Install_HiSPARC.bat`
   - `$AdminDir\niruntimeinstaller\setup.exe`
   - `$AdminDir\nirte2012\setup.exe`
   - `$AdminDir\ftdi_drivers\dpinst.exe`

6. Setup and install **OpenVPN**.

   - Register OpenVPN into its registry (`HKLM,SOFTWARE\OpenVPN`). Several variables are set and defined.
   - Install the *tap* driver by executing: `$OpenVpnDir\bin\tapinstall.exe` with the following 3 arguments:
     - `Install`
     - `$OpenVpnDir\driver\OemWin2k.inf`
     - `tap0901`
   - Install the OpenVPN server by executing:
     `$OpenVpnDir\bin\openvpnserv.exe –install`
   - Unzip (extract) the certificate zip file `$CertZip` into the folder:
     `$OpenVpnDir\config`

7. Setup and install **TightVNC**.

   - Register TightVNC into its registry (`HKLM,SOFTWARE\TightVNC`). A lot of variables are set and defined.
   - Install the TightVNC server by executing:
     `$AdminDir\tightvnc\tvnserver.exe –install –silent`

8. Install **NAGIOS** (`nsclient++`) by executing:
   `$AdminDir\nsclientppNSClien++.exe /install`

9. Install and setup **ODBC.**

   - Install ODBC by executing: `$AdminDir\odbcconnector\Install_HiSPARC.bat`
   - Register ODBC into its registry (`HKLM,SOFTWARE\ODBC\ODBC.INI\buffer`). A couple of variables are set and defined.

10. Install the **LabVIEW** Runtime Engines version 8.2.1 and 2012 by executing:
    `$AdminDir\niruntimeinstaller\setup.exe` with the following 4 arguments:
    - `/acceptlicenses`
    - `yes`
    - `/r:n`
    - `/q`

    and `$AdminDir\nirte2012\setup.exe` with the same 4 arguments.

11. Install the **FTDI** (USB) drivers by executing: `$AdminDir\ftdi_drivers\dpinst.exe /q`

12. The 3 *services* (OpenVPN, TightVNC and NAGIOS) are set to *automatic* and *started*.

13. Adjust some firewall rules by opening specific ports. The following NSIS commands (from `firewall.nsh` and based on the `SimpleFC` plugin) are executed:

```
SimpleFC::AddPort  5666 "HiSPARC Nagios" 6 2 2 194.171.82.1 1
SimpleFC::AddPort 12489 "HiSPARC Nagios" 6 2 2 194.171.82.1 1
SimpleFC::AddPort  5900 "HiSPARC VNC"    6 2 2 172.16.66.0/24 1
SimpleFC::AllowDisallowIcmpInboundEchoRequest 1
```

Script 3-1: firewall rules

14. Update the `adminUpdater` version number into the configuration file and HiSPARC's registry entry: `SOFTWARE\HiSPARC\AdminVersion`

15. Read the NI installation folder variable (`$NIdir`) from the registry. Key is `HKLM`, variable is: `SOFTWARE\National Instruments\Common\Installer\NIDIR.` The `$NIdir` folder is granted "**FullAccess**" for all users.

16. The *admin* uninstaller, `adminuninst.exe`, is created and stored into the folder: `$HisparcDir\persistent\uninstallers`

3.3 userUnpacker

The **userUnpacker** is responsible for the installation of supporting programs and tools, like the *monitor* and *updater*. They are mainly Python scripts. It is part of the `hisparcInstaller` and executed by it. However, the `userUnpacker` can also be executed separately. If installed separately, it uses a simplified GUI, without fancy decorations. The following sequence is carried out by the `userUnpacker`:

1. The `.onInit` function of the userUnpacker, located in the `userunpacker.nsi` file, starts with several checks:

   - Registry check: the installation is aborted if the HiSPARC registry entry path does not exist or is empty (`HKLM, SOFTWARE\HiSPARC\Path`). If it exists, the variable `$HisparcDir` is assigned to it.
   - HiSPARC folder check: the installation is aborted if the folder `$HisparcDir` does not exist.
   - Configuration file check: the installation is aborted if the configuration file `$HisparcDir\persistent\configuration\config.ini` does not exist.

2. The entire `user` folder, as found in the HiSPARC base area besides the `bake` folder, is copied into the folder `$HisparcDir`.

3. The variable `$UserDir` is declared and assigned to: `$HisparcDir\user`. The installation is aborted if the folder `$UserDir` does not exist.

4. Update the `userUnpacker` version number into the configuration file and HiSPARC's registry entry: `SOFTWARE\HiSPARC\UserVersion`

5. The *user* uninstaller, `useruninst.exe`, is created and stored into the folder: `$HisparcDir\persistent\uninstallers`

4 Executing the HiSPARC uninstaller

Like the installer the uninstaller consists of three parts. One for the *main* part, one for the *admin* part and finally one for the *user* part. The main uninstaller contains and executes the admin and user part as well. It is recommended not to execute the admin or user uninstaller separately. The following list shows what the uninstaller, as started by the '`All Programs\HiSPARC\Expert\Uninstall`' shortcut, does:

1. The `.un.onInit` function of the main uninstaller starts with some checks:

   - Administrator check: the removal of HiSPARC is aborted if the user has no administrator privileges.
   - Registry check: the removal is canceled if the HiSPARC registry entry path does not exist or is empty (`HKLM, SOFTWARE\HiSPARC\Path`). If it exists, the variable `$HisparcDir` is assigned to it.
   - HiSPARC folder check: the removal is aborted if the folder `$HisparcDir` does not exist.
   - The user is now asked whether he wants to remove HiSPARC completely or not. If not, the removal is canceled. If confirmed, the removal of HiSPARC is granted.

2. The removal continues with the execution of the following batch program:
   `$HisparcDir\persistent\startstopbatch\StopAdminMode.bat`
   This batch program calls a Python script to stop the Admin-Mode applications: TightVNC, NAGIOS and OpenVPN and a Python script to stop the User-Mode applications: LabVIEW, MySQL, the *monitor* and *updater*. Figure 4.1 shows the output.

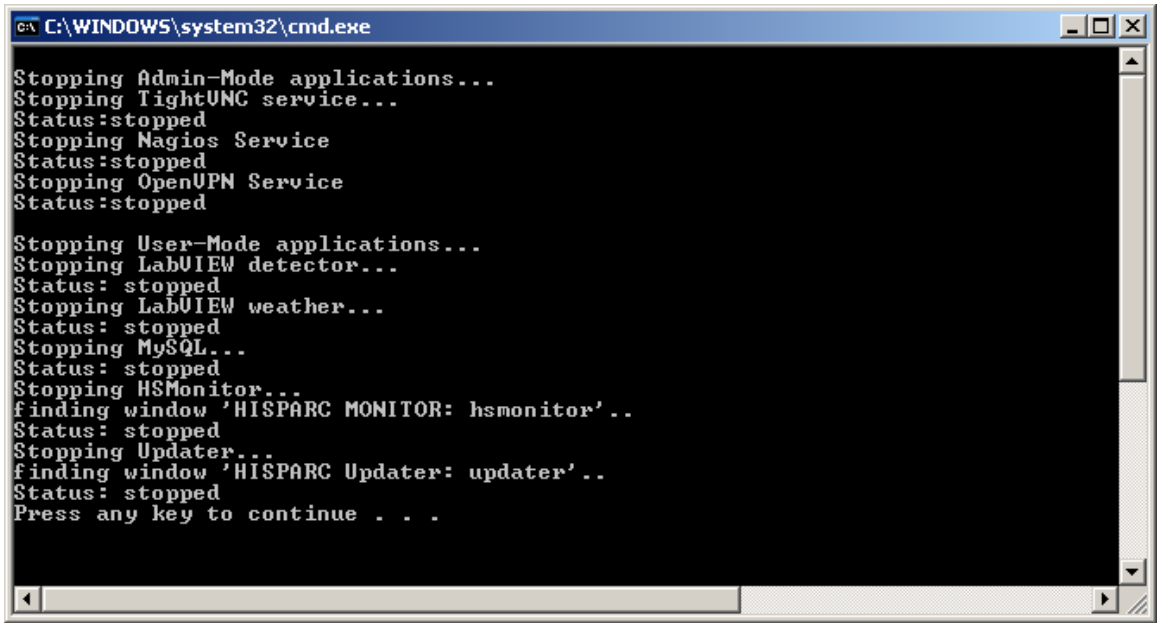


Figure 4.1: Output of StopAdminMode

   As a result all services are stopped and the entire HiSPARC system is not longer running. Press any key to continue.

3. The main uninstaller now starts the admin uninstaller by executing:
   `$HisparcDir\persistent\uninstallers\adminuninst.exe /S`
   NB: /S runs the uninstaller silently.

   - Because this uninstaller can also be called directly, the user is checked again if he has administrator privileges. If not, exit.

- Assign `$AdminDir` to `$HisparcDir\admin`.
- Assign `$OpenVpnDir` to `$AdminDir\openvpn32` if it is installed on a 32-bit processor and to `$AdminDir\openvpn64` if it is not.
- Uninstall OpenVPN
  - Remove OpenVPN by calling: `$OpenVpnDir\openvpnserv.exe –remove`
  - Remove registry key: `SOFTWARE\OpenVPN`
  - Remove the *tap* device by calling: `$OpenVpnDir\bin\tapinstall.exe remove tap0901`
  - Remove the folder: `$AdminDir\openvpn`
- Uninstall TightVNC
  - Remove TightVNC by calling: `$AdminDir\tightvnc\tvnserver –remove`
  - Remove registry key: `SOFTWARE\TightVNC`
  - Remove the folder: `$AdminDir\tightvnc`
- Uninstall NAGIOS
  - Stop NAGIOS by calling: `$AdminDir\nsclientpp\NSClient++.exe /stop`
  - Uninstall NAGIOS by calling: `$AdminDir\nsclientpp\NSClient++.exe /uninstall`
  - Remove the folder: `$AdminDir\nsclientpp`
- Uninstall ODBC
  - Remove the registry key: `SOFTWARE\ODBC\ODBC.INI\buffer`
  - Clear the registry variable: `SOFTWARE\ODBC\ODBC.INI\ODBC Data Sources\buffer`
  - Uninstall ODBC by calling: `$AdminDir\odbcconnector\Uninstall_HiSPARC.bat`
  - Remove the folder: `$AdminDir\odbcconnector`
- Uninstall the NI Runtime engines (optional)
  The user is asked whether to remove the *NI Runtime Environment* or not. If yes, the following actions are carried out:
  - Read the NI installation folder variable (`$NIdir`) from the registry. Key is `HKLM`, variable is: `SOFTWARE\National Instruments\Common\Installer\NIDIR`.
  - Remove the engines by calling: `$NIdir\Shared\NIUninstaller\uninst.exe /qb /x /all`
  - Remove the folder: `$NIdir`

  As a result both Runtime Engines (version 8.2.1 and 2012) are removed.
- The entire folder `$AdminDir` is removed.
- The 3 *services* (OpenVPN, TightVNC and NAGIOS) are stopped and removed from the list of services. Stopping the services may seem redundant, because they have been stopped already in step 2 of the un-installation.
- The final step of the admin uninstaller is to remove itself, so the following file is deleted: `$HisparcDir\persistent\uninstallers\adminuninst.exe`

4. The main uninstaller now starts the user uninstaller by executing:
   `$HisparcDir\persistent\uninstallers\useruninst.exe /S`
   NB: /S runs the uninstaller silently.

   - Assign `$UserDir` to `$HisparcDir\user.`
   - The entire folder `$UserDir` is removed.
   - The final step of the user uninstaller is to remove itself, so the following file is deleted: `$HisparcDir\persistent\uninstallers\useruninst.exe`

5. The two local user accounts, `hisparc` and `admhisparc`, are removed.

6. The registry values for the automatic logon are cleared.

7. Delete the HiSPARC registry key: `SOFTWARE\HiSPARC.`

8. Delete the HiSPARC uninstall registry key:
   `SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\HiSPARC`

9. The group of HiSPARC shortcuts of ‘All Programs’ is removed.

10. The shortcut to `Startup.bat` inside the general (i.e. All Users) Startup folder is removed.

11. The main installer is removed, so the following file is deleted:
    `$HisparcDir\persistent\uninstallers\mainuninst.exe`

12. The user is asked whether he wants to keep the HiSPARC program folder or not. Remember that the `admin` and `user` folder are removed already! If confirmed the folder `$HispardDir` is removed.

13. The final step of the main uninstaller is whether the user wants to reboot the computer or not. If yes, the following command is executed: `shutdown –r –f –t 0`

5 Release Notes & Version History

5.1 Document Version 2.1 and HiSPARC 6.9

HiSPARC version 6.9 was made with an installer which was thoroughly checked and improved in November 2011. A lot of additional checks were added and where possible redundant code removed. The following remarks are applicable to this and later versions of the installer and is related to version 2.1 of this document:

- The installer is now completely in English. Not only the appearance of the installer but also the NSIS code.
- The mandatory installation path is now: `C:\Program Files\HiSPARC`. There is no choice anymore. Previous versions used as default a ridiculous long name: `C:\Program Files\HiSPARC Client Application.`
- The configuration file `$HisparcDir\persistent\configuration\config.ini` is used by a lot of supporting programs (even between the 3 installers) to get and exchange certain variables. The new installer publishes these variables into the HiSPARC registry. The installer itself already makes use of these facility. In future, after all programs have been modified, the use of this configuration file will become obsolete.
- The HiSPARC software made extensively use of a virtual drive, which mapped the `$HisparcDir` to a single letter, mostly **Z**:. This symbolic link is used as a shortcut to the HiSPARC folder. The installer itself did the same, but any reference to this virtual drive is removed. Version 6.11 of the HiSPARC software, June 2012, has no reference anymore to this virtual drive.
- In January 2012 the entire HiSPARC software repository has been moved from *Bazaar* to *GitHub.*

5.2 Document Version 2.2 and HiSPARC 6.11.4

Version 2.2 of this document (January 2013) is related to HiSPARC version 6.11.4. Compared with 2.1 (June 2012) the following modifications have been applied:

- The version number of the installer has been extended with a release number.
- If executed not on a 32-bit platform, the installer will not quit anymore, but may be continued if wanted. It makes it possible to run it on a 64-bit machine (which has not been tested yet!).
- The version of the FTDI driver set is upgraded from 2.08.08 to 2.08.24 (x86, 32-bit).
- An additional NI Runtime-engine is added to the installer. Apart from version 8.2.1 also version 2012 is installed. Old and new versions of LabVIEW executables may now run in parallel.
- The NI run-time-engines do not require a license and serial number. The initialization file (`hisparcspec.ini`) containing this information has been removed.
- After installation of the NI Runtime engines, their original folders inside the `admin` folder are not removed anymore.

5.3 Document Version 2.3 and HiSPARC 6.11.9

Version 2.3 of this document (May 2013) is related to HiSPARC version 6.11.9. Compared with 2.2 (January 2013) the following modifications have been applied:

- The applied version of OpenVPN has been upgraded from 2.1.4 to 2.2.2. It makes it possible to use OpenVPN on 64-bit processors as well. The *tap* installer requires a 64-bit binary for a 64-bit machine and a 32-bit binary for a 32-bit machine. The installer contains 2 instances of version 2.2.2. One for 32-bit processors (`openvpn32`) and one for 64-bit processors (`openvpn64`).
- The station number and password are now mandatory (step 3 of the `hisparcInstaller`). It is not allowed anymore to leave it empty. The station number allows only digits.
- The `Startup.bat` file (step 17 of the `hisparcInstaller`)has been modified. If the user `hisparc` logs on the registry variable `ScreenSaveActive` is disabled.
- The installation results of OpenVPN, TightVNC, NAGIOS, ODBC and the NI RunTime engines are checked. If not correct a message-box will appear containing the error code. The installation is suspended. It resumes when the error is confirmed by pressing the OK button.

6 Appendix

6.1 Setting up a HiSPARC Development Area

The following sequence describes how to get the HiSPARC software and what modifications are necessary in order to create an installer. It is assumed that *GitHub*[2] (and inclusively *git* version 1.7.9) is installed on your PC.

1. Run the command prompt of *git*: `All Programs` → `Git` → `Git Bash`
   It behaves like a (Bourne Again) Shell.

2. Go to the folder you want the software to reside.

3. Execute: `git clone https://github.com/HiSPARC/station-software`

4. The checkout location should now contain the folders `station-software` and inside this folder it should contain (at least): `admin`, `bake`, `db_buffer`, `persistent` and `user`.

5. Several passwords, like the one for the database buffer and the HiSPARC admin-user `admhisparc`, are not committed in the HiSPARC repository. Before you can make an installer this information has to be supplied. This classified information can be obtained from a password protected PDF file:

   http://www.nikhef.nl/~robert/docu/HiSPARC_Installer_Classified.pdf

6.2 Supporting Python Scripts

```
import subprocess

NSISPATH = "./nsis"

class nsiHandling():

    def __init__(self):
        self.nsisExe = "%s/makensis.exe" % NSISPATH

    def compileNSI(self, nsiPath, defines):
        definelist = ""
        #print str(defines)
        for i in defines:
            definelist = "%s /D%s" % (definelist, i)
        #print definelist
        command = "%s /V1 %s %s" % (self.nsisExe, definelist, nsiPath)
        #print command
        print "Compiling %s..." % nsiPath
        nsiProcess = subprocess.Popen(command)
        nsiProcess.wait()
        print "Compilation of %s finished!" % nsiPath
```

Script 6-1: nsis.py

[2] GitHub: A cloud-repository to share code easily. Based on git, a distributed version control system.

```
class userInput():

    def __init__(self):
        pass

    def getVersion(self, type):
        version = ""
        while (version == ""):
            version = raw_input("What %s software version do you want to make? \n" % type)
        return version
```

Script 6-2: userinput.py

6.3 HiSPARC Registry

Table 6.1 shows the contents of the HiSPARC registry of a typical HiSPARC station.

Variable	Value
Path	C:\Program Files\HiSPARC\hisparc
VolatilePath	C:\Program Files\HiSPARC\hisparc
DisplayName	HiSPARC
HiSPARCVersion	6.11.4
AdminVersion	6
UserVersion	11
Release	4
ReleaseDate	20130129_125929
StationNumber	510
HasHiSPARC	1
HasWeather	0
HasMagnetic	0
HasLightning	0

Table 6.1: HiSPARC Registry

6.4 Package Versions

Table 6.2 shows the version numbers of the packages as used in HiSPARC version 6.11.9.

Package	Version
NI Runtime engines	**8.2.1** and **2012** (32-bit)
MySQL Server	**5.1.53** (x86, 32-bit)
MySQL ODBC	**5.1.8** (x86, 32-bit)
OpenVPN	**2.2.2**
TightVNC	**2.0.3**
FTDI	**2.08.24**
GPS-DSPMon	**1.46**
NAGIOS	**0.3.8**
Python	**2.7.1**

Table 6.2: Version numbers of the Packages

6.5 Issues and miscellaneous remarks

1. **Driver Signing**: During installation one (virtual) hardware driver is installed. The *tap* driver, part of the OpenVPN installation, creates one. On Windows XP machines a justified modal notification warning will appear. To suppress this warning in advance, perform the following:
   `Start → Control Panel → Driver Signing → Ignore`
   It was impossible to make this mechanism part of the installer. Multiple tap installs yield multiple warnings. A single tap de-install clears them all (without modal warnings).
   On Windows 7 machines, there is no such thing as Driver Signing. It is disabled.

2. **Return values of package installation**: On success TightVNC, ODBC, NAGIOS and OpenVPN return with the value 0. The FTDI driver installation always returns with the value 512. The NI Runtime engines return with the value: 3010 (0xBC2), which means according to the NI web-pages: ERROR_SUCCESS_REBOOT_REQUIRED.

3. **ODBC installation**: The installation of this package is rather nested. One `.bat` file after the other. The return value in case of success is passed by an "`EXIT /B 0`" command. But if it is no success (`≠0`) then in older installers a `PAUSE` was executed, stopping the installation or de-installation. In the latest installer this is replaced by a timeout of 10 seconds in order to read the diagnostics. However, the command `TIMEOUT` is not supported on Windows XP, so a general solution to create a timeout of 10 seconds is applied: "`ping 1.1.1.1 –n 1 –w 10000 > nul`".

4. **Uninstaller**: The main installer uses the NSIS `ExecWait` primitive to run the *admin* and *user* installer. The primitive does what is says, it waits until completion. However, if used by an uninstaller, it starts the execution but does not wait for completion. According to the NSIS web-pages it is a known "feature". For the HiSPARC uninstaller it means that the *main*, *admin* and *user* uninstallers run in parallel.