

Welcome to the Nao Personal Webserver



Requirements:

- Windows Win32/64 (Tested on Windows 7)
- [Python for Windows](#) (Tested with Python DLL: 2.7.3)
- [Python SDK 2.7 1.12.5 Windows 32bit x32](#) (Aldabaran Software Development Kit)

* 1) Installation

- NPWS Folder structure

* 2) Startup and checking system

- Python Installation Check
- Execute a test script on the Nao
- Set the user login information and start the Server
 - Set the user login information

* 3) Start the server

- Using the webbrowser client menu
 - Login Screen
 - Client Main Menu
 - Client System Menu
 - Client Script Menu
 - Client Script Menu with parameters

* 4) Create your own Client Sub Menu's and execute scripts on the server.

- How does it work..

1) Installation:

Extract the zipfile on a disk and create a shortcut to the "NaoPersonalWebServer.exe" or just dbl-click on this executable to start the program. The executable is using the network connection, so expect messages from firewall applications.

NPWS Folder structure:

* css (npws style sheets)

* images (npws images)

* js (npws javascripts)

- jquery
 - images
 - themes
 - ui
- jquery.mobile-1.0

* scripts (user and npws scripts)

* templates (npws html templates)

2) Startup and checking system:

If the NPWS startup without any error messages, two checks must be done to see if the program can see the Python installation and if the program can execute a Python script on the Nao.

Python Installation Check:

This is a quick check. Open the "Python" section of the NPWS and see if the Python information is displayed. If you don't see any information, be sure you have installed Python on your machine. See above "Requirements".

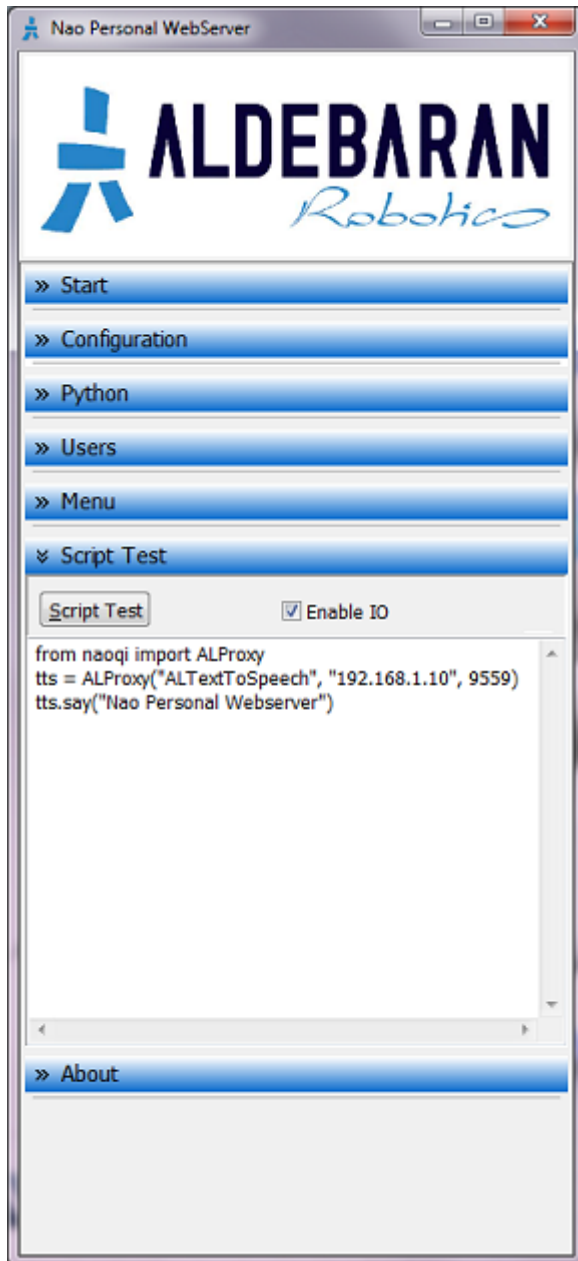


Execute a test script on the Nao:

Open the "Script Test" section of the NPWS and notice the small test script. Change the IP-Address and Port in the script to match it with your Nao settings. Press on the "Script Test" button to execute the script.

If you check the "Enable IO" checkbox, any Python messages will be displayed in the "Python" section where the Python information is displayed.

If your Nao speaks "Nao Personal Webserver", the NPWS is successfully installed. If you don't get any response or an error message is displayed, be sure you have installed the Aldebaran Python SDK. See above "Requirements". You can change this test script if you like. The NPWS will remember this script if you close the program.



Set the user login information:

Before we can start the server, the user login information must match the settings of the Nao. In this way the server can handle more Nao's or handle one Nao if you switch for example the network where your Nao is connecting to. You can login as another user instead of changing the IP all the time.

The user login information is stored as a XML file on the NPWS program folder. (users.xml) This file can be edited by an XML editor or by a special dialog in the NPWS program. Change one of the users to match it with your Nao settings. (Right-Mouse Click for popup window

or press on the name in the tree to edit directly)



```
<?xml version="1.0" encoding="UTF-8"?>
- <users>
  - <user>
    <username>Robert</username>
    <password>123</password>
    <ip>192.168.1.9</ip>
    <port>9559</port>
  </user>
  - <user>
    <username>Guest</username>
    <password>pass</password>
    <ip>10.192.168.1</ip>
    <port>9559</port>
  </user>
  - <user>
    <username>Nao</username>
    <password>123</password>
    <ip>192.168.1.10</ip>
    <port>9559</port>
  </user>
</users>
```

3) Start the server:

The Start button:

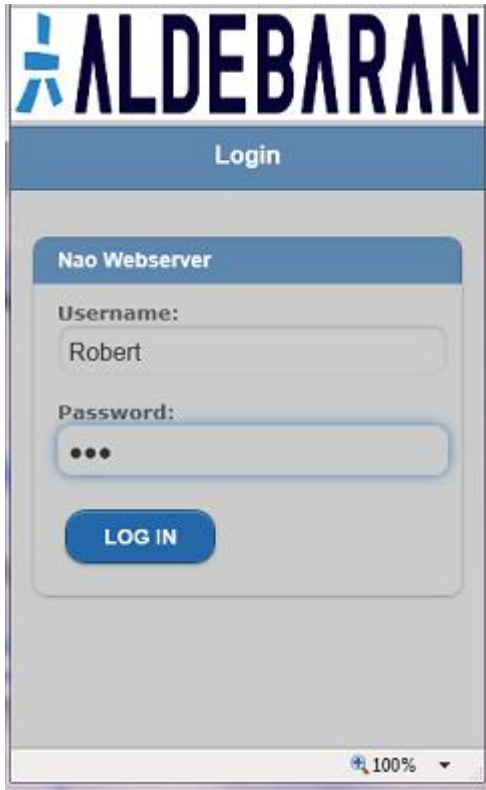
Start the server by pressing on the red start button in the "Start" section. The button will change to green. Look at the information in the "Configuration" section. Here you can see the sever IP and Port. Any browser on the same network can connect to the server by using this IP and Port. If you use a port other than 80, you have to add it to the URL in the browser. For example for the screenshot under it will be "http://192.168.1.3:8080/". If you press on the globe button in the "Start" section the default web browser is started and a connection with the server is made using "localhost" as IP.



Using the web browser client menu :

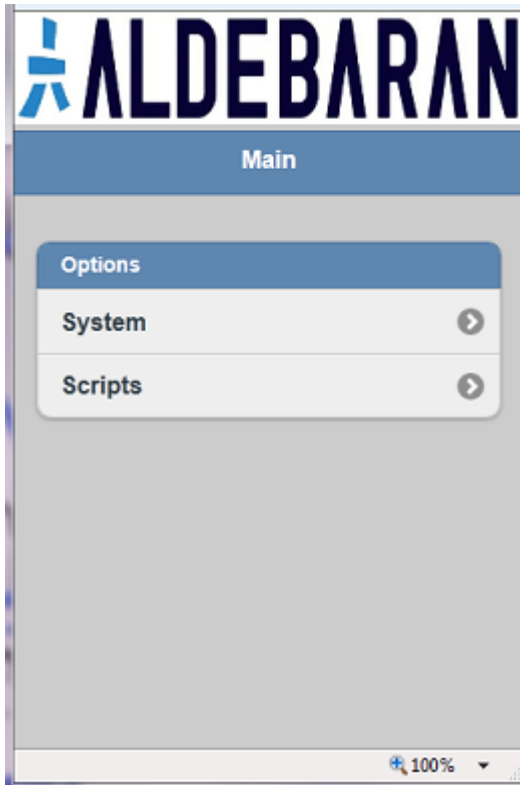
Login Screen:

This is the start screen of the NPWS web browser client. Fill in a username and password using the information added by "Set the user login information". Press the Login button to open the client main menu.



Client Main Menu:

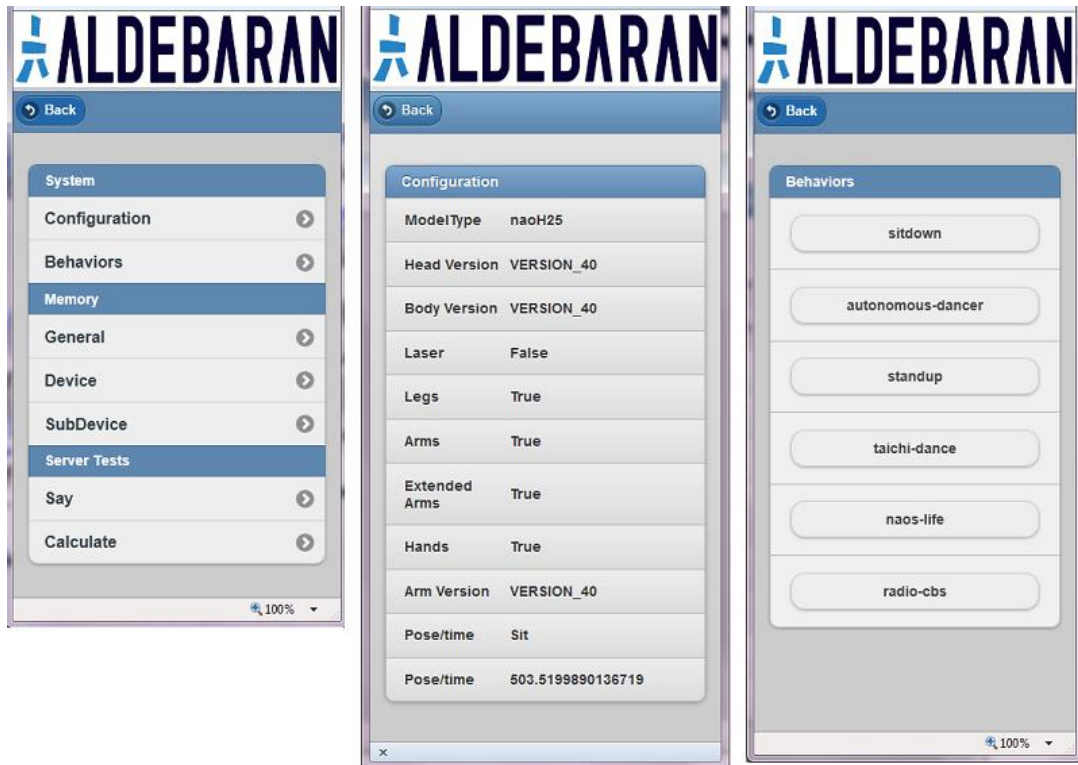
This is the current main menu. Press on the "System" button to see if we can ask Nao's system information, start behaviors and execute internal server Python test scripts.



Client System Menu:

This is the current system menu. Press on the "Configuration" button to see if we can ask Nao's system configuration. Try to start behaviors and try execute internal server Python test scripts.

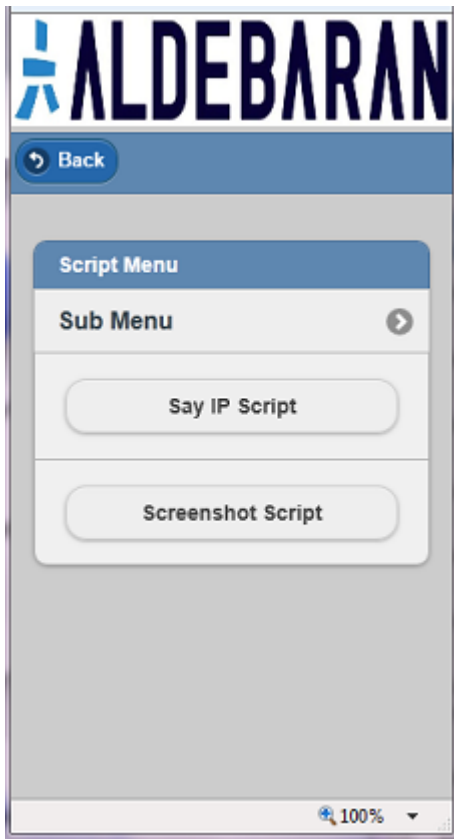
Warning ! The memory menu is still in development. The memory overview has been split into three parts (General, Device and SubDevice) but they are still too large. The General overview is also not OK yet. Asking these overviews can result into a timeout. I had some good results using the Chrome Browser. This browser will ask you to if the browser should wait for result. Other browsers can crash the server. These overviews need to be split later into alphabetic views. (A,B,C,...)



The Server Test are two internal server scripts compiled into the server. The "Say" button will show a simple "speak" menu. Fill in a word or sentence and Nao should say this. The calculate is an idea still in development. You can fill in a calculation like "2+2*4". If executed, Nao will say the calculation and the answer. (still all very basic..)

Client Script Menu:

Go back to the main menu and press the "Scripts" button. This is the core menu of the NPWS. With this Script menu the users are able to make their own submenu's and execute self made Python scripts on the server.



Press on the "Say IP Script". The system will start the "say.py" script located in the scripts folder.

```
from naoqi import ALProxy
tts = ALProxy("ALTextToSpeech", str(npws.ip), npws.port)
tts.say("message is "+str(npws.ip))
npws.html("<li>executed : say.py<li>")
```

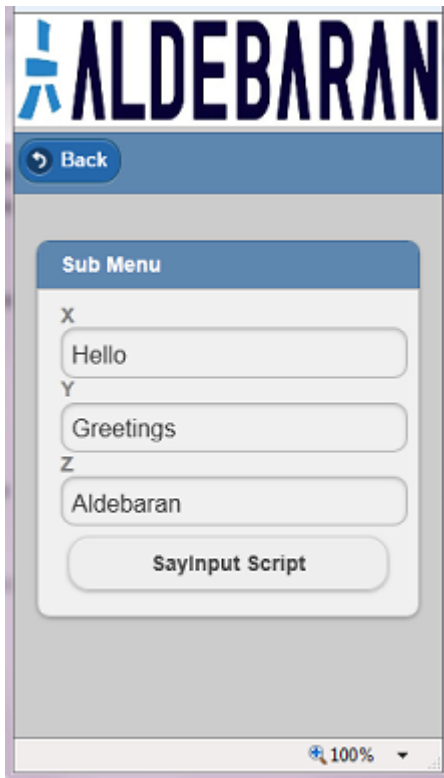
Inside the script you can use the npws class created by the server. There are four fields published.

- 1) npws.ip : this is the IP of the login user Nao IP.
- 2) npws.port : this is the Port of the login user Port.
- 3) npws.html : this can be filled by the user and will be used as a return page in the client webbrowser after the script has been executed. It supports basic html syntax.
- 4) npws.input : See under, when using input parameters in the menu.

Client Script Menu with parameters:

It is also possible to create a menu with Input fields. These field parameters will be transfered to the script where the user can use them

inside the script. Press on the "Sub Menu" button to open an example menu with three input fields.



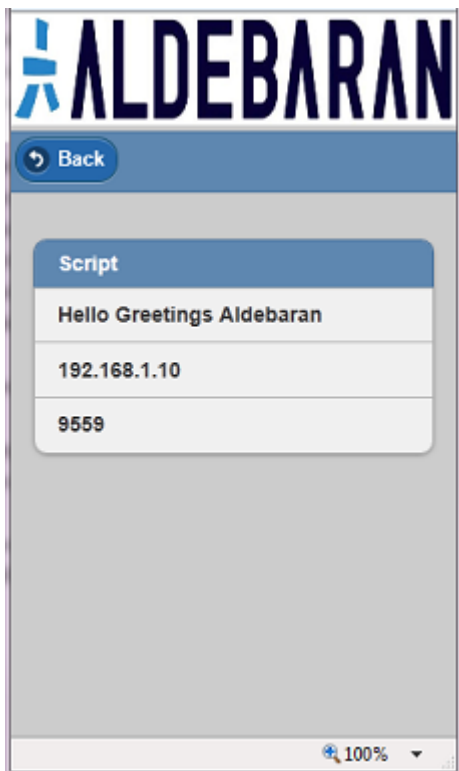
Press on the "SayInput Script". The system will start the "sayinput.py" script located in the scripts folder.

Inside the script you can use the npws class created by the server. You can use the ".input" field array of the class to read the values added in the client input fields. The fields in the array are in the same order as the input fields of the menu.

```
from naoqi import ALProxy<br>  
  
tts = ALProxy("ALTextToSpeech", str(npws.ip), npws.port)  
  
#fill the npws.html with info. This will be returned to the client  
after the script has been executed.  
npws.html='<li>  
  
#go through the npws.input list and take the value in the inputfield  
var.  
#let Nao say these inputfields  
for inputfield in npws.input:  
    tts.say(inputfield)  
    npws.html=npws.html+' '+inputfield  
  
npws.html=npws.html+'</li>'
```

```
#return also the Nao IP and Port to the client.  
npws.html=npws.html+'<li>'+npws.ip+'</li>'+<li>'+str(npws.port)+'</li>'+<br>
```

The script above will return the following page by the npws.html field. As you can see the html "li" tag will create a row in the page.



4) Create your own Client Sub Menu's and execute scripts on the server:

The client menu is stored as a XML file on the NPWS program folder. (menu.xml) This file can be edited by an XML editor or by a special dialog in the NPWS program.

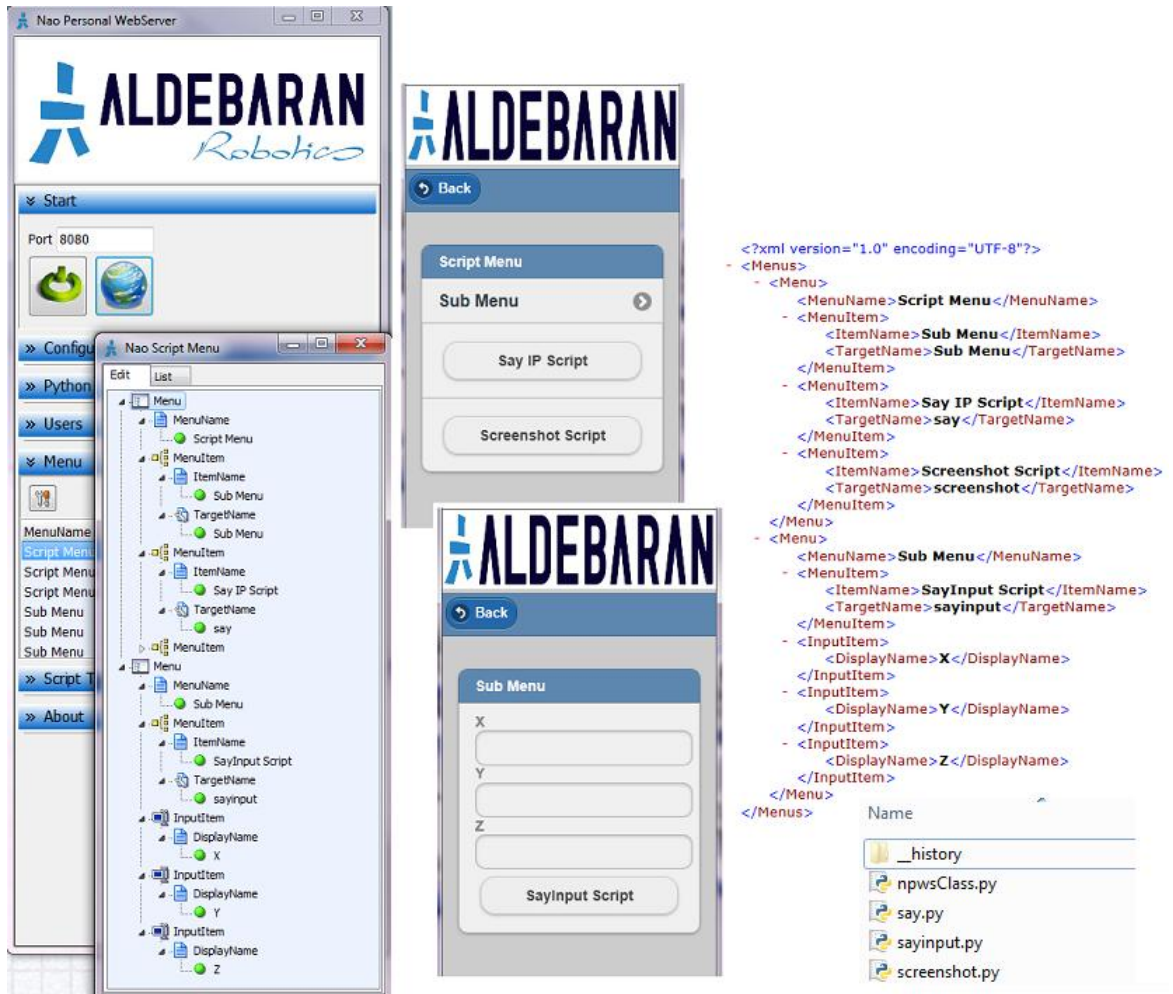
Note : It will take some time to add all the edit possibilities in the NPWS menu editor. Like moving menu's and items up and down (re-order) is not supported yet.

Left: Server side - NPWS program with menu edit dialog. (add and edit menu's)

Middle: Client side (web browser) - User Menu.

Right: NPWS Menu XML.

Right Bottom: Folder with Python scripts that are executed by the menu.



There are six different tags for the menu XML:

Menu: New Submenu.

MenuName: Name of the menu displayed as header text.

MenuItem - ItemName: Button with ItemName text.

MenuItem - TargetName: Python script filename OR Menu - MenuName (submenu).

InputItem: Editbox for script parameter.

DisplayName: Headertext for editbox.

How does it work..:

A "Menu" tag will create a new submenu with a header text of the "MenuName" tag.

A "MenuItem" tag creates a button with a text of the "ItemName" tag.

The "TargetName" tag is used to define the Python Script that will be executed **BUT** if the "TargetName" tag has the same name as another Menu "MenuName", the system will create a button targeting this menu and **NOT** using this "TargetName" tag to execute a python script. In this way the user can create more submenu's.

So, the server will figure out if the "TargetName" is used as "MenuName" and will create a button targeting this other menu and if no "MenuName" is found the server will create a button that will execute a python script with the name "TargetName".

In the example above you see two submenu's. "Script Menu" and "Sub Menu".

In the "Script Menu" you see two "Menu Items".

The first item has a "TargetName" "Sub Menu" and because the second menu "MenuName" is "Sub Menu" the server will create a button targeting this second menu.

The first menu has also a "MenuItem" with "ItemName" "Say IP Script" and because there is no other menu with this "MenuName", the server will create a button that executes a Python Script with the "TargetName" "Say".