



## Example – openPDC on Raspberry Pi 3 Model B, Raspbian Jessie Lite

This implementation summary uses the following software and hardware:

- Grid Protection Alliance openPDC Product Release Latest Stable Version 2.2.70.0
- **Raspberry Pi 3 Model B**
- Raspbian Jessie compatible wireless network USB adapter
- HDMI monitor for use until networking is configured
- Raspbian Jessie Lite 2016-05-27
- Mono 4.4.2.11

### Raspberry Pi Setup

18. Download Raspbian Jessie Lite image from:

[https://downloads.raspberrypi.org/raspbian\\_lite\\_latest](https://downloads.raspberrypi.org/raspbian_lite_latest)

19. Write the image to an SD card.

<https://www.raspberrypi.org/documentation/installation/installing-images/README.md>

20. Boot the Raspberry Pi using the SD card. Run the Raspbian desktop Menu / Preferences / Raspberry Pi Configuration.

```
sudo raspi-config
```

- A. Expand the Filesystem
- B. Change the **pi** user's password from **raspberry** to a new password.
- C. Set the Internationalization Options for Locale and Timezone
- D. Set the Advanced Options to enable SSH and set the Pi's Hostname. This document uses the hostname: *openpdc-pi3*
- E. *Finish* the raspi-config program
- F. Edit the `/etc/default/keyboard` file to set the keyboard configuration parameters. raspi-config does not reliably do this.

```
sudo nano /etc/default/keyboard
# KEYBOARD CONFIGURATION FILE
# Consult the keyboard(5) manual page.
XKBMODEL="pc105"
XKBLayout="us"
XKBVARIANT=""
XKBOPTIONS=""
BACKSPACE="guess"
```

G. Reboot the Pi

21. Configure the Pi's Ethernet to connect to the network. This example uses the built in wireless adapter.

- A. Review the `/etc/network/interfaces` file to find the path to the wifi supplicant configuration file for the wlan# interface. Edit the supplicant configuration file to add the network SSID and PSK pass phrase.

```
cat /etc/network/interfaces
...
allow-hotplug wlan0
iface wlan0 inet manual
    wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf
```



```
...
sudo nano /etc/wpa_supplicant/wpa_supplicant.conf
country=US
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
network={
    ssid="your-wifi-ssid"
    psk="your-wifi-passphrase"
}
```

B. Reboot the Pi

C. Make a note of the Pi's IP address

```
sudo ifconfig -a
```

D. Raspbian wireless setup details can be found at:

<https://www.raspberrypi.org/documentation/configuration/wireless/wireless-cli.md>

22. By default, the Pi can now be accessed by remotely using a terminal running **ssh**

A. Remotely ping test the network connection. You may need to configure your DNS or PC's hosts file to associate the IP address to the new hostname.

```
ping <the Pi's IP address>
```

```
ping openpdc-pi3
```

B. For example, use **ssh** in a **git-bash** session in Windows

```
ssh pi@openpdc-pi3
```

C. Run the standard update commands.

```
sudo apt-get update
```

```
sudo apt-get upgrade
```

23. Optional: Install **git**:

```
# Switch to Home folder
```

```
cd ~
```

```
# Install Git prerequisites - this takes a while
```

```
sudo apt-get install build-essential libssl-dev libcurl4-openssl-dev libexpat1-dev tk-dev gettext -y
```

```
# Get
```

```
wget https://www.kernel.org/pub/software/scm/git/git-2.9.3.tar.gz
```

```
tar xzvf git-2.9.3.tar.gz
```

```
cd git-2.9.3
```

```
# Make Git takes a while, Install Git is quick
```

```
make prefix=/usr/local all
```

```
sudo make prefix=/usr/local install
```

```
# Test Git
```

```
git --version
```

24. Install **unzip**

```
sudo apt-get install unzip
```



## Mono Installation on the Raspberry Pi Zero

25. Build Mono from a release tarball<sup>41</sup> Note: This process takes many hours.

A. Install dependencies. Our earlier git build procedure installed some of these.

```
sudo apt-get install build-essential gettext -y
sudo apt-get install libtool automake autoconf mono-devel -y
```

B. Download the source code tarball file and extract it

```
cd ~
wget http://download.mono-project.com/sources/mono/mono-4.4.2.11.tar.bz2
tar -xvzf mono-4.4.2.11.tar.bz2
cd mono-4.4.2
```

C. Configure the source code

```
./configure --prefix=/usr/local
```

1) Configure Results:

Engine:

```
Host:          armv7l-unknown-linux-gnueabi
Target:        armv7l-unknown-linux-gnueabi
GC:           sgen and Included Boehm GC with typed GC and parallel mark
TLS:          __thread
SIGALTSTACK:  yes
Engine:       Building and using the JIT
oprofile:     no
BigArrays:    no
DTrace:       no
LLVM Back End: no (dynamically loaded: no)
```

Libraries:

```
.NET 4.6:      yes
Xamarin.Android: no
Xamarin.iOS:   no
Xamarin.WatchOS: no
Xamarin.TVOS:  no
Xamarin.Mac:   no
JNI support:   IKVM Native
libgdiplus:    assumed to be installed
zlib:          system zlib
```

D. Run **make** to build and install

```
make
sudo make install
mono --version
Mono JIT compiler version 4.4.2 (Stable 4.4.2.11/f72fe45 Wed Aug 10 21:41:34
EDT 2016)
Copyright (C) 2002-2014 Novell, Inc, Xamarin Inc and Contributors. www.mono-
project.com
TLS:          __thread
SIGSEGV:      normal
Notifications: epoll
Architecture: armel,vfp+hard
Disabled:      none
```

<sup>41</sup> Derived from the procedures described on <http://www.mono-project.com/docs/compiling-mono/linux/>



```
Misc:          softdebug
LLVM:          supported, not enabled.
GC:            sgen
```

## openPDC Server openPDC Software Installation

### 26. Install openPDC

- A. Download and run the installation script file using the Preservation option **-p**

```
cd ~/
mkdir GPA
cd GPA
wget http://www.gridprotectionalliance.org/Products/openPDC/Scripts/install-openPDC.sh
sudo bash install-openPDC.sh -p
```

### 27. Test openPDC

```
sudo mono /opt/openPDC/openPDC.exe -RunAsConsole
```

While the console is running, type **version** to verify the openPDC version, the type **exit** to quit.

### 28. Register openPDC to run automatically

```
sudo bash register-openPDC.sh
```

### 29. Test openPDC control commands. Test them one at a time and wait for each to complete before testing the next command.<sup>42</sup>

```
sudo /opt/openPDC/openPDC stop
sudo /opt/openPDC/openPDC start
sudo /opt/openPDC/openPDC restart
sudo /opt/openPDC/openPDC pause
sudo /opt/openPDC/openPDC resume
```

### 30. Start openPDC on the openPDC Server and use **openPDCConsole** to assure openPDC is running

```
mono /opt/openPDC/openPDCConsole.exe
```

- A. Type **version** to check the openPDC version

```
App Domain: openPDCShell.exe, running on .NET 4.0.30319.42000
Machine Name: openpd-piz
OS Version: Unix 4.4.13.7
Product Name: Raspbian GNU/Linux 8 (jessie) using Mono
Working Memory: 75.8939 MiB
Execution Mode: 32-bit
Processors: 4
Code Base: opt/openPDC/openPDCShell.exe
Build Date: 8/14/2016 8:07:08 AM
Version: 2.2.70.0
```

- B. Type **exit** to quit openPDC

### 31. Enable Root Login for SSH. This is needed to copy files from Windows to the openPDC Server's data folder on the Pi.

- A. Edit the server's **/etc/ssh/sshd\_config** file:

```
sudo nano /etc/ssh/sshd_config
```

---

<sup>42</sup> If these command are executed in quick succession in a script, the abuse may result in openPDC not clearing its process lock file, *openPDC.pid*. In testing, this condition, was fixable by rebooting the Pi device.



## *Deploying the openPDC on POSIX Platforms*

Change: **PermitRootLogin without-password**

To: **PermitRootLogin yes**

B. Restart the ssh service:

```
sudo /etc/init.d/ssh restart
```

C. See: <https://linuxconfig.org/enable-ssh-root-login-on-debian-linux-server> for details.

32. Install **sqlite3** used for running the openPDC *add-user.sh* script

```
sudo apt-get install sqlite3
```