GRID PROTECTION ALLIANCE

*** WORK IN PROGRESS - DRAFT ***

This document only contains the first 12 pages. The remainder is still work in progress. Future drafts and the final document will supersede this draft.

The final version of this document is expected to be completed by October, 2016.

Deploying the openPDC on POSIX Platforms

Grid Protection Alliance, Inc.
www.gridprotectionalliance.org
Last Updated: July 20, 2016

Derived from the **openPDC POSIX Deployment Instructions** April 8, 2015 Document

Contents

Introduction 4
  Documentation Scope 4
  Editor's Note 4
Raspberry Pi Setup 5
Installation Prerequisite: Mono 6
  Installing Mono 6
  Verifying Mono Installation 7
  Testing Mono 7
Installing the openPDC 8
  Troubleshooting Installation 9
Running the openPDC 10
  Execution as Terminal Application 10
  Execution as Daemon 10
    Controlling the Daemon 11
  Running the Remote Console 11
  Testing the Default Configuration 13
    IEEE C37.118 Output Stream 13
    Gateway Exchange Protocol 13
Configuring the openPDC 14
  Manually Updating Configuration using SQL 14
  Updating Configuration Using the openPDC Manager 14
    Installing the openPDC Manager 14
  Using the Default SQLite Configuration 16
    Copy SQLite Configuration for Local Update 16
    Use SQLite Configuration via File Share 17
  Using Other Databases 18
  Enabling the openHistorian 19
Completing Security Configuration 20
  Managing User Accounts 21
    Adding New Users 21
    Authenticating Cross Platform Users 21
    Disabling Pass-Through Authentication 22
  Updating Transport Layer Security 23

Manually Creating a Certificate 23
Using an Existing Certificate 24
Changing the TLS Version 24
Running the openPDC with Elevated Rights 25
Securing the Configuration Database 25
Running on a Raspberry Pi 27

Introduction

With the release of version 2.1 of the openPDC, GPA has implemented features that allow the openPDC to easily be used in deployments running a POSIX[1] compatible operating system such as Linux and Apple's Mac OS X. GPA has tested several common distributions and hardware combinations that include:

- Debian on x86 style hardware (64-bit)
- Ubuntu on HyperV (64-bit)
- Ubuntu on SEL-3354 hardware (32-bit)
- Raspbian on Raspberry Pi hardware (32-bit)
- Mac OS X on MacBook Air hardware (32-bit)

This document describes the process to install, configure and operate the openPDC on a POSIX compatible operating system. The directions that follow focus on Linux deployments of the openPDC; however, where necessary, there are exceptions noted that may be needed for Mac OS X deployments.

Documentation Scope

This document is derived from the Grid Protection Alliance document: **Deploying the openPDC on POSIX Platforms**, April 8, 2015

The examples in this documentation are intended to be repeatable proof of concepts for evaluation purposes. Production deployments will necessarily have more detailed requirements than this document covers.

Unless otherwise noted, the examples in this document were tested using the following software and hardware:

- ➢ openPDC Stable Release 2.2.70.0
- ➢ Raspberry Pi 3 Model B
- ➢ Raspbian Jessie full desktop 2016-05-27

Editor's Note

This document was work in progress over a period of time. During this time, various software versions changed. Priority was given to updating and testing the current software versions at the time the examples were written. Lower priority was given to updating the screen shot images and those may display earlier than final versions.

[1] POSIX, Portable Operating System Interface, is a family of IEEE standards to assure compatibility among operating systems. Most distributions of Linux, Mac OS X and iOS as well as Android are POSIX compatible.

Raspberry Pi Setup

1. Download Raspbian Jessie Full Desktop image from:

   https://downloads.raspberrypi.org/raspbian_latest

2. Write the image to an SD card.

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

3. Boot the Raspberry Pi using the SD card. Change the default password and hostname using either the Raspbian desktop Menu / Preferences / Raspberry Pi Configuration or a Terminal. Reboot after changing the hostname.

   A. Open a Terminal and change the **pi** user's password from *raspberry* to a new password.

   ```
   passwd --help
   ```

   B. Optionally, change the Pi's Hostname by editing the */etc/hostname* and */etc/hosts* files

   ```
   sudo nano /etc/hostname
   sudo nano /etc/hosts
   sudo reboot
   ```

4. Configure the Pi's Ethernet to connect to the LAN and Internet.

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

   ```
   sudo ifconfig -a
   ```

5. 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.

   For example, ping the hostname *openpdc-pi3*

   ```
   ping openpdc-pi3
   ```

   B. For example, to 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 dist-upgrade
   ```

6. If you want to remotely access the Raspbian Full Desktop, install the **x11vnc** remote control application.

   A. Install and run **x11vnc** using following commands:

   ```
   sudo apt-get install x11vnc
   x11vnc -display :0
   ```

   B. For more information and instructions see: http://www.karlrunge.com/x11vnc

Installation Prerequisite: Mono

The openPDC is a .NET application and as such it requires a Common Language Runtime (CLR) engine to execute. For use in POSIX compatible environments, the openPDC uses Mono as its runtime engine. Mono is a cross platform implementation of the .NET framework that is widely available on various hardware and software architectures. For the openPDC version 2.1, Mono version 3.12 or later is required[2].

To install Mono, follow the installation guide using Xamarin packages for your distribution:

http://www.mono-project.com/download/

Installing Mono

The following installation procedure is derived from the procedure described in Install Mono on Linux - Debian, Ubuntu, and derivatives using the following reference:

http://www.mono-project.com/docs/getting-started/install/linux/

```
# Mono Installation
sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys
3FA7E0328081BFF6A14DA29AA6A19B38D3D831EF

echo "deb http://download.mono-project.com/repo/debian wheezy main" | sudo tee
/etc/apt/sources.list.d/mono-xamarin.list

sudo apt-get update
sudo apt-get upgrade

# mod_mono sources
echo "deb http://download.mono-project.com/repo/debian wheezy-apache24-compat main" |
sudo tee -a /etc/apt/sources.list.d/mono-xamarin.list

# libgdiplus sources
echo "deb http://download.mono-project.com/repo/debian wheezy-libjpeg62-compat main" |
sudo tee -a /etc/apt/sources.list.d/mono-xamarin.list

# do the update and upgrade again
sudo apt-get update
sudo apt-get upgrade

# Install Mono – this takes a while
sudo apt-get install mono-devel
sudo apt-get install mono-complete
sudo apt-get install referenceassemblies-pcl
sudo apt-get install ca-certificates-mono
```

[2] It is possible that an older version of Mono may work, but the openPDC version 2.1 incorporates many features of .NET 4.5 that are only available in later versions of Mono. The openPDC has not been fully tested with earlier versions of Mono; therefore, use of Mono version 3.12 or later, e.g., version 3.12.1 that includes the FREAK security fix, is strongly recommended.

Verifying Mono Installation

Once Mono is installed, validate that the system is running the proper version of Mono using the ***mono*** application with the version parameter, ***-V***, from a terminal session. The reported version should be "version 3.12" (or later) as shown below.

```
Mono JIT compiler version 4.4.1 (Nightly 4.4.1.0/4747417 Wed Jun 22 11:38:12 UTC 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
    Misc:          softdebug
    LLVM:          supported, not enabled.
    GC:            sgen
```

If after installation of the latest version of Mono the ***mono*** application reports an older version[3], modify the system path so that version 3.12 of Mono is found first.

If there are no Mono version 3.12 packages available for your POSIX distribution, then Mono must be compiled manually. In this case, follow these instructions:

http://www.mono-project.com/docs/compiling-mono/

Testing Mono

The following commands are derived from the examples from

http://www.mono-project.com/docs/getting-started/mono-basics/

This tests both Mono and HTTPS.

```
# Download a test C# source file
wget
https://raw.github.com/mono/mono/master/mcs/class/Mono.Security/Test/tools/tlstest/tlstes
t.cs

# Compile the C# source file.  This might return warning messages
mcs tlstest.cs /r:System.dll /r:Mono.Security.dll

# Run the executable
mono tlstest.exe https://www.nuget.org

# The last command should not return any errors.
```

[3] It is possible to have multiple versions of Mono simultaneously installed on a system.

Installing the openPDC

Due to the vast number of POSIX operating systems and hardware combinations, there are currently no precompiled deployment packages for the openPDC[4]. To begin the process of installing the openPDC, download the following script file[5]:

http://www.gridprotectionalliance.org/Products/openPDC/Scripts/install-openPDC.sh

Mac OSX Note:

The ***wget*** application must be available before running this script. To install wget, install Brew from http://brew.sh and then run:

```
brew install wget
```

Note that X11/XQuartz is not required for openPDC.

For a default installation and configuration, the script can be executed right away to get things started. However, to customize the installation or complete the security configuration, review the following installation options.

Below are the available script parameters to allow customization of the installation:

-n: Use nightly builds
-p: Preserve source code
-d: Install destination (defaults to /opt/openPDC/)
-u: Uninstall the openPDC

To fully enable security for this installation, as detailed later, preserve the local source code by using the ***-p*** parameter. To install the openPDC using the latest source code instead of the published stable release, use the ***-n*** parameter to install the latest nightly build.

The script should be run using the ***bash***[6] shell with root access, for example, to install the openPDC and preserve the source code after installation run the following:

```
sudo bash install-openPDC.sh -p
```

[4] As time allows, GPA will add deployment packages for common POSIX distributions, e.g., Debian and Ubuntu on Linux and Mac OS X. Third-parties are welcomed to assist with maintaining up-to-date deployment packages for the openPDC on various distributions.
[5] During execution the script will require internet access to download files to be installed. Since other scripts and source code files will be downloaded as part of the installation, it is best to run the script from its own folder.
[6] The ***bash*** shell command is required over the normal ***sh*** command to be able to properly test for Mac OS X platforms. Most all scripts to be executed, as referenced in this document, will require bash for this reason.

Troubleshooting Installation

If there are issues with the script during the installation process, validate that the Mono build system, XBuild, is working properly. To do this, execute the ***xbuild*** application with a version parameter, ***/version***, from a terminal session and verify that the application reports version 12.0 of the XBuild Engine, similar to the following:

```
$ xbuild /version
XBuild Engine Version 12.0
Mono, Version 3.4.1.0
Copyright (C) 2005-2013 Various Mono authors
```

If the version of XBuild is not the same as reported above or the application fails to run, verify the path to the ***xbuild*** command is correct.

If the installation script needs to be run again, clear out the temporary files in the installation folder (i.e., the folder where script was run) and delete the destination folder (e.g., /opt/openPDC) before the next attempt.

Running the openPDC

The installation script will automatically deploy the openPDC with a default configuration, as such the application can be run immediately. The openPDC can be started as a terminal application or registered as a daemon that will run in the background and startup automatically.

Execution as Terminal Application

To test the openPDC or validate its current configuration, it can simply be run as a terminal application. To run the openPDC as a terminal application, execute the ***openPDC.exe*** application using the Mono runtime and the ***RunAsConsole*** parameter:

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

This will start the openPDC as a console style application with feedback reported directly to the current session. When running in this mode, input can be accepted just by typing. Note that keyboard input is being logged even if any user input is visually interrupted by a new status update from the openPDC. As an example to start issuing commands, just type ***Help*** and press enter for a list of possible commands.

To exit the application and stop the openPDC, just type the command ***Exit*** and press enter.

Execution as Daemon

To run the openPDC automatically at startup, the application must be deployed to run as daemon in the background. To configure the daemon to run automatically startup, execute the registration script[7]:

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

If the openPDC needs to be unregistered later, use the ***-u*** parameter with this script to unregister the service. Note that uninstalling the openPDC will also automatically unregister the service.

When running the openPDC as a background service, input and output from the application must be handled by another application, such as the openPDC Remote Console. See the *Running the Remote Console* section for more details.

[7] Note that needed scripts are downloaded as part of the install process. In order to run properly both the ***openPDC.sh*** and ***register-openPDC.sh*** scripts are required to be in the same folder during execution; for Mac OSX, this also includes the ***openPDC.plist*** file.

Controlling the Daemon

To manually control the openPDC service running as a daemon, use the following commands:

Start Service

```
sudo /opt/openPDC/openPDC start
```

Restart Service

```
sudo /opt/openPDC/openPDC restart
```

Pause Service

```
sudo /opt/openPDC/openPDC pause
```

Resume Service

```
sudo /opt/openPDC/openPDC resume
```

Stop Service

```
sudo /opt/openPDC/openPDC stop
```

Running the Remote Console

To view the current activity and issue commands to the openPDC while it is running as a daemon, run the openPDC Remote Console.

Run Locally

The remote console is already preconfigured to properly connect to the locally running openPDC when executed from the same machine. To start the remote console, execute the following command from a terminal session[8]:

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

When you want to quit the openPDCConsole, type the command: ***exit***[9]

Run Remotely

The openPDC can be accessed from a remote machine by running the remote console application on another machine, including from a Windows installation[10]. In order for the remote console application to properly connect to the machine running the openPDC, the machine's IP address or DNS name will be required. There are two options for specifying the openPDC server to connect to: one way is by using a command line parameter for the openPDCConsole application, the other is setting the configuration so that the openPDCConsole always connects to the desired remote machine.

[8] The remote console application does not require being run as the root user, i.e., no ***sudo*** required.

[9] Typing the ***exit*** command cleanly closes and disconnects the console. Aborting using [Ctrl+C] may lead to misbehavior where typed characters are no longer echoed to the terminal.

[10] Run the Windows installer for the openPDC to get the openPDC Remote Console running in a Windows environment; download from here: https://www.gridprotectionalliance.org/Products/openPDC/Releases/. The installed components can be reduced to only the manager and tools during the installation process by deselecting the openPDC Service option, see *Installing the openPDC Manager* section for more details.

Specifying the openPDC Server on the Command Line

The openPDCConsole application will accept the openPDC server name using the ***-server*** command line parameter, for example:

```
mono /opt/openPDC/openPDCConsole.exe -server="openPDC:8500;interface=0.0.0.0"
```

Important Note: The **-server="…"** connection string parameter on the command line should not have any spaces in it.

Specifying the openPDC Server in the Configuration

To configure the console application running on a remote machine to always connect to a specific openPDC service instance, make sure the remote console application is not running, then edit[11] the ***openPDCConsole.exe.config*** file and find the following settings that need to be updated:

```
<?xml version="1.0"?>
<configuration>
  <categorizedSettings>
    <remotingClient>
      <add name="ConnectionString" value="server=openPDC:8500;interface=0.0.0.0" />
      <add name="IntegratedSecurity" value="False" />
    </remotingClient>
</configuration>
```

The ***openPDC*** text, as shown in the ***ConnectionString*** or in the ***-server*** command line parameter example above, must be replaced with the DNS name or IP that is running the openPDC service. Do not forget to include "`;interface=0.0.0.0`" in the changes as this forces the connection to be IPv4[12]. If the openPDC Remote Console application is being used on a Windows machine to connect to the openPDC service running in a POSIX environment, also set ***IntegratedSecurity*** to ***False***[13]. Integrated security allows an application to login as the currently authenticated user without reentering credentials, but this is only available on Windows.

Once these settings are changed and the configuration file is saved, the console application can be run on the remote machine and connected to the openPDC service.

Exit Console

The remote console application can be closed by issuing an ***Exit*** command and pressing enter. Note that unlike when running the openPDC as a terminal application, issuing the ***Exit*** command from a remote console session will not terminate the remotely running openPDC application, it will only close the remote console session.

[11] The configuration files are simple text based XML files and can be edited with most any kind of text editing tool. There is also an included UI based configuration editing tool, ***ConfigurationEditor.exe***, which can be run to make changes to the configuration files that is available on both Windows and POSIX platforms.
[12] The default configuration is setup as IPv4 to support as many possible distributions as possible. If the system running the openPDC supports IPv6, the server and client connections can be configured to use IPv6 by specifying "`;interface=::0`" in the relevant configuration settings and connection strings. The ***interface*** setting is used to specify the IP of the network interface controller (NIC) to use for the connection – an IP of zero means that the default NIC should be used for the connection; the format of the interface IP setting determines the IP stack version, i.e., IPv4 or IPv6, to use for the connection.
[13] If the command line option is being used from a Windows machine for specifying the server name, this setting must be changed in the config file first. The ***IntegratedSecurity*** setting is ignored in POSIX environments.