

Secure Policy-Based Configuration Management (PBCONF) User's Guide

Secure Policy-Based Configuration Management (PBCONF) User's Guide

Software Installation Manual

Rev 3, April 6, 2018

DISCLAIMER OF WARRANTIES AND LIMITATION OF LIABILITIES

THIS REPORT WAS PREPARED AS AN ACCOUNT OF WORK SPONSORED BY AN AGENCY OF THE UNITED STATES GOVERNMENT. NEITHER THE UNITED STATES GOVERNMENT NOR ANY AGENCY THEREOF, NOR ANY OF THEIR EMPLOYEES, MAKES ANY WARRANTY, EXPRESS OR IMPLIED, OR ASSUMES ANY LEGAL LIABILITY OR RESPONSIBILITY FOR THE ACCURACY, COMPLETENESS, OR USEFULNESS OF ANY INFORMATION, APPARATUS, PRODUCT, OR PROCESS DISCLOSED, OR REPRESENTS THAT ITS USE WOULD NOT INFRINGE PRIVATELY OWNED RIGHTS. REFERENCE HEREIN TO ANY SPECIFIC COMMERCIAL PRODUCT, PROCESS, OR SERVICE BY TRADE NAME, TRADEMARK, MANUFACTURER, OR OTHERWISE DOES NOT NECESSARILY CONSTITUTE OR IMPLY ITS ENDORSEMENT, RECOMMENDATION, OR FAVORING BY THE UNITED STATES GOVERNMENT OR ANY AGENCY THEREOF. THE VIEWS AND OPINIONS OF AUTHORS EXPRESSED HEREIN DO NOT NECESSARILY STATE OR REFLECT THOSE OF THE UNITED STATES GOVERNMENT OR ANY AGENCY THEREOF.

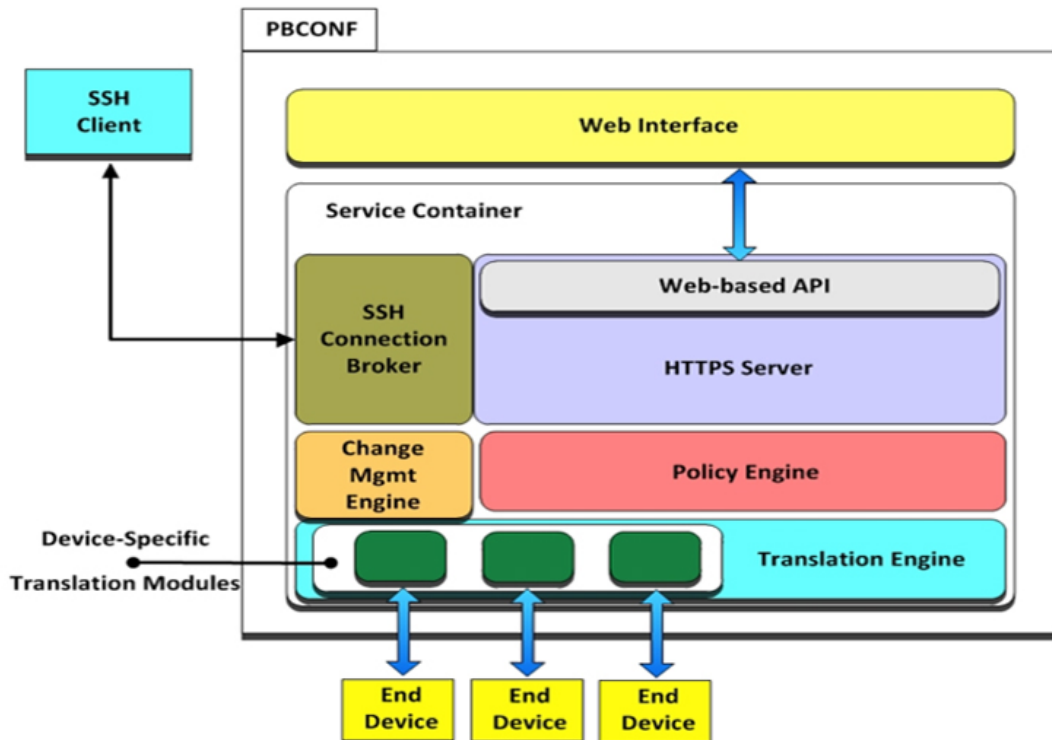
SOFTWARE DESCRIPTION

The Secure Policy-Based Configuration Framework (PBCONF) is an interoperable, open source framework for secure remote configuration of modern and legacy devices.

Description

The PBCONF software is currently in a proof-of-concept stage of development. Its purpose is to support the secure configuration and remote access of modern and legacy devices from a variety of vendors. The open source framework combines a policy engine with a translation engine to address the interoperability challenges of various remote access control methods and provides utilities with a single organization-wide view of the security configuration of their power delivery devices. The system leverages distributed architecture concepts to enable both centralized and peer-based configuration of the devices to support scalability and resiliency.

The PBCONF system is composed of several functional components or subsystems that are contained within a PBCONF node. The PBCONF node comprises a web interface and a service container. The service container is made up of subcomponents: the SSH connection broker, an HTTPS server that hosts a web-based API, a policy engine, a change management engine, and a translation engine that hosts the device-specific vendor modules.



Benefits and Value

Benefits and values provided to utilities by the PBCONF software include:

- Offers a cost-effective solution that supports scalability and resiliency

- Allows for consistent global policy application across vendor products
- Enables efficient inquiry and security/compliance checks against current policies

Platform Requirements

The initial target OS environment for PBCONF is Linux Ubuntu 16.04 LTS (Long Term Support version). The future expectation is that the PBCONF system will be made available to platforms using OS environments other than Linux.

CONTENTS

1 OVERVIEW OF THE DEVELOPMENT ENVIRONMENT	1-1
Assumptions	1-1
Operating System (OS).....	1-1
Go Development Language	1-1
Java.....	1-1
Git Account.....	1-1
Text Editor	1-1
Notation	1-2
Superuser.....	1-2
2 SETTING UP THE ENVIRONMENT	2-1
System Requirements	2-1
Step 1 – Install Dependencies	2-1
Step 2 – Set up the Local Git Environment	2-2
Step 3 – Set up the Build Workspace.....	2-2
Step 4 - Download and Install Go	2-3
Step 5 - Install the Protocol Buffers.....	2-4
Step 6 – Download and Install PBCONF Code and Drivers	2-6
3 BUILDING PBCONF	3-1
Step 1 – ENV Update	3-1
Step 2 – Ontology Build	3-1
Step 3 - Tests	3-2
Step 4 - Static Analysis.....	3-2
Step 5 - Create the Installer Script.....	3-4
Step 6 – Run the Installer Script	3-4
4 INSTALLING PBCONF AND PROVISIONING NODES	4-1
Step 1 – Install PBCONF	4-1
Step 2 – Provision the Node	4-1
Node Name.....	4-1
Initial Admin User, Password and Email.....	4-1
API Listener Address	4-2
SSH Broker Listener Address	4-2
Enable Web Interface	4-2
Web Interface Listener Port	4-2
Step 3 – Update the PBCONF Configuration File.....	4-3
5 RUNNING PBCONF	5-1
Step 1 - Run PBCONF.....	5-1
Step 2 – Set up the IP address of the host machine	5-2
Step 3 - Access the web-based user interface	5-2

Step 4 – Exchange security certificates	5-4
Step 5 – Linking PBCONF nodes.....	5-5
6 LIST OF ABBREVIATIONS	6-1
A CREATING ADDITIONAL NODES WITH THE PBCONF INSTALLER.....	2
Assumptions and Prerequisites.....	2
Step 1 – Install Dependencies	2
Step 2 – Setup Git	2
Step 3 – Transfer and Install.....	3

1

OVERVIEW OF THE DEVELOPMENT ENVIRONMENT

Assumptions

This guide provides step-by-step instructions for installing PBCONF. Basic familiarity with the command line is required. The user is assumed to be familiar with attributes and functions of an operating system such as user privileges, folder structure, copying files, browsing a website, and assigning a fixed IP address to the local machine.

Operating System (OS)

The initial target OS environment for PBCONF is Ubuntu version 16.04 LTS (Long Term Support). The instructions provided in this guide will focus on an installation procedure for the desktop version (rather than the server version). The future expectation is that the PBCONF system will be made available to platforms using OS environments other than those that are based on Linux.

Go Development Language

The PBCONF system is primarily written utilizing the free and open source programming language Go (<https://golang.org>). It will be necessary to download and install Go to complete the PBCONF installation. Instructions for establishing a Go programming environment on the Ubuntu OS are provided in this User's Guide.

Java

The Java tool chain and runtime environments were used in the initial development and testing of the PBCONF ontology engine. New installations of PBCONF should be built with the most recent version of Java Development Kit (JDK) and Java Runtime Environment (JRE).

Git Account

The code bases for several of PBCONF's dependencies are maintained in Git¹ repositories. Prior to starting the build and installation process, the user must have valid credentials to the appropriate Git repositories that host the code bases. Typically, these credentials include a user name and email address. The procedures in this guide assume that a user account has been established.

Text Editor

PBCONF installation will require the use of a text editor to make environment-specific changes. Gedit is a text editor that is supplied with the Ubuntu desktop distribution and is suitable for this purpose. If using a different text editor, the user should ensure that it provides access to super user privileges (sudo). Where this user's guide indicates <your editor of choice>, the user should

¹ <https://en.wikipedia.org/wiki/Git>

insert the command to run their editor (for example, gedit), being sure to **not** include the < and > characters.

Notation

Throughout this installation guide, any user-provided information will be denoted within command line instructions between the < and > characters. These characters are **not** to be included in the typed command. For example, in the following command line sample:

```
Prompt> git config --global user.email <you@example.com>
```

The text you@example.com would be replaced with the email address of the person or persons that would receive emails related to their use of the git repository.

Superuser

Individuals performing the installation should have root privileges as these privileges are required when executing installation steps that require superuser² (root) access.

² <https://www.linux.com/learn/linux-101-introduction-sudo>

2

SETTING UP THE ENVIRONMENT

System Requirements

PBCONF has the following minimum host platform requirements driven by the Go programming language. If installing PBCONF on a virtual machine (VM) hosted by Windows or MacOS then the architecture requirements will be met. The following requirements are listed for advanced users running Linux on dedicated hardware. If needed, more detailed requirements are available at <https://github.com/golang/go/wiki/MinimumRequirements>.

- Operating System
 - o Linux Kernel 2.6.23 or later with glibc – Note: this requirement is met through the installation of Ubuntu 16.04 LTS, which is recommended in this guide.
- Architecture (for advanced users running on dedicated hardware)
 - o AMD64 (development and test environment)
 - o x386
 - o ARM
 - o S390x
 - o Ppc64le

PBCONF has been tested with the Go Language version 1.8.3. Subsequent toolchain/language versions should be functionally compatible with PBCONF in accordance with the Go Compatibility Pledge³.

Step 1 – Install Dependencies

The first step toward supporting the PBCONF build process is to install the following software on the Ubuntu 16.04 LTS host. Instructions are provided below.

SSH
SQLITE3
GIT
UNZIP
ANT
JDK
Maven

³ <https://golang.org/doc/go1compat>

All of the packages listed above, upon which PBCONF is dependent, can be downloaded and installed from a single Ubuntu shell command line:

```
Prompt> sudo apt-get update && sudo apt-get install -y build-essential ssh sqlite3 git unzip ant default-jdk maven
```

Step 2 – Set up the Local Git Environment

If not already configured, Git will require a minimum amount of configuring, consisting of the user identity. The Git account credentials are used in the following Ubuntu shell command line, which establishes the Git environment:

Note: the reference to global in the instructions below requires 2 dashes (--global).

Note: the operations below do not result in any user feedback when they are successful.

```
Prompt> git config --global user.email <you@example.com>  
Prompt> git config --global user.name <Your Username>
```

Step 3 – Set up the Build Workspace

Next, the user will need to establish a workspace on the local machine. The workspace is the primary structure used to instantiate and build the code base for the PBCONF software system. The first step in building the workspace is the selection of a primary host directory under which the workspace will be instantiated. This User's Guide references the installer's home directory (\$HOME) as the suggested location to establish this workspace. If a different path is desired for the workspace, the \$HOME environment variable should be changed to match that directory, or the user should select a different environment variable (pointing to the desired path) in place of the \$HOME environment variable used throughout this guide.

To initialize the workspace, the following commands should be entered from an Ubuntu command shell prompt: (Note: the following operations do not result in any user feedback when they are successful.)

```
Prompt> mkdir -p $HOME/PBCONF_Workspace/go/pkg  
Prompt> mkdir -p $HOME/PBCONF_Workspace/go/bin  
Prompt> mkdir -p $HOME/Downloads  
Prompt> mkdir -p $HOME/temp
```

Step 4 - Download and Install Go

The following command should initiate the download of the tar⁴ file needed for the Go installation. The expected file size is 85.86 MB. If the command should fail (due to either a connection failure or incorrect file size), try the alternative command underneath it.

```
Prompt> wget -O $HOME/Downloads/go1.8.3.linux-amd64.tar.gz  
https://storage.googleapis.com/golang/go1.8.3.linux-amd64.tar.gz
```

Alternative command in case the command above fails to download the tar file:

```
Prompt> wget -O $HOME/Downloads/go1.8.3.linux-amd64.tar.gz  
https://golang.org/dl/go1.8.3.linux-amd64.tar.gz
```

After the file has been downloaded, the following command should be executed to place the Go language files in the “*/usr/local*” directory structure:

```
Prompt> sudo tar -C /usr/local -xzf $HOME/Downloads/go1.8.3.linux-  
amd64.tar.gz
```

Note: the operation above does not result in any user feedback when successful.

The Go binary distributions assume that the files will be installed in the “*/usr/local/go*” directory by default. After extracting the Go files, the user must ensure that the “*/usr/local/go*” binary directory is added to the operating system’s \$PATH environment variable so that the Go files can be found during the build process. To do this, open Ubuntu’s “. profile” file (note that there is a “.” before the word profile) with a text editor as follows:

```
Prompt> <your editor of choice> ~/.profile
```

⁴ <https://www.lifewire.com/tar-file-2622386>

Add the following lines to the end of the .profile file:

Note that these should be the last two lines of text in the file, regardless of other text

```
export PATH=${PATH}:/usr/local/go/bin
export GOPATH=$HOME/PBCONF_Workspace/go
```

Save and close .profile.

To activate the changes to the .profile file, execute the following command from the terminal command line:

```
Prompt> source ~/.profile
```

To verify the installation of Go, execute the following command:

```
Prompt> go version
```

An example of the desired response is shown below:

```
go version go1.8.3 linux/amd64
```

If needed, additional information for installing Go can be found on the Go language website⁵.

Information specific to the installation of Go in an Ubuntu environment can be found at the following websites:

<https://github.com/golang/go/wiki/Ubuntu>
<https://golang.org/doc/install>

Step 5 - Install the Protocol Buffers

Protocol buffers are used to encode structured data for interchange and storage between programs. Google uses protocol buffers for most of its internal protocols and file formats.

⁵ <https://golang.org>

Following are instructions for retrieving and installing the necessary Google protocol buffers for Linux.

From the command line, enter the following to retrieve the zipped protocol buffers:

```
Prompt> wget -O $HOME/Downloads/protoc-3.0.0-linux-x86_64.zip  
https://github.com/google/protobuf/releases/download/v3.0.0/protoc-  
3.0.0-linux-x86_64.zip
```

Alternatively, a web browser can be used to retrieve the file. To do so,

1. Navigate to: <https://github.com/google/protobuf/releases/tag/v3.0.0>
2. Find and download the file: [protoc-3.0.0-linux-x86_64.zip](#)

Assuming that the zip file now resides in the Downloads directory, move the file and extract the contents into the \$HOME/temp directory as follows:

```
Prompt> mv $HOME/Downloads/protoc-3.0.0-linux-x86_64.zip $HOME/temp  
Prompt> cd $HOME/temp  
Prompt> unzip protoc-3.0.0-linux-x86_64.zip
```

Following this operation, copy the extracted protoc file in to the “/usr/local/bin” directory:

```
Prompt> cd bin  
Prompt> sudo cp protoc /usr/local/bin/protoc
```

To verify the overall extraction process, execute the following command from a terminal shell:

```
Prompt> sudo which protoc
```

Verify that the response is “/usr/local/bin/protoc”.

To continue verification, take ownership and repeat the query by typing the following two commands:

(Note: <your user id> refers the username that is logged into the Linux machine. This is the same username that appears before the @ symbol in the prompt.)

```
Prompt> sudo chown <your user id> /usr/local/bin/protoc  
Prompt> which protoc
```

Again, verify that the resulting directory structure is “*/usr/local/bin/protoc*”.

Step 6 – Download and Install PBCONF Code and Drivers

To install the PBCONF code, it must be retrieved from the code repository, hosted on Github. Install PBCONF System software, communication modules, and the two device-specific modules, “sel421” and “linux” using the following commands: (Note that some delay may occur before any feedback is provided.)

```
Prompt> go get -v github.com/iti/pbconf/cmd/pbconf  
Prompt> go get -v github.com/iti/pbconf/cmd/pbbroker  
Prompt> go get -v github.com/iti/pbconf/cmd/drivers/linux  
Prompt> go get -v github.com/iti/pbconf/cmd/drivers/sel421
```


3

BUILDING PBCONF

Building the Go Language portions of PBCONF is a five-step process. Each of the steps can be performed using the Linux command line, and assumes that the installer has previously configured the build environment per the previous chapters in this guide.

Step 1 – ENV Update

Modify the path of JAVA_HOME and ANT_HOME to fit your environment. First, locate and then open “env” using your editor of choice with the following shell commands:

```
Prompt> cd $HOME/PBCONF_Workspace/go/src/github.com/iti/pbconf  
Prompt <your editor of choice> ontology/env
```

The file will include the following lines:

```
JAVA_HOME=${JAVA_HOME:-${/usr/libexec/java_home}}  
ANT_HOME=${ANT_HOME:-${ant_dir}}
```

Modify the two lines to reflect the changes shown below and include the third line as shown:

```
JAVA_HOME=/usr  
ANT_HOME=/usr/bin/ant  
protoc_dir=/usr/local/bin
```

Step 2 – Ontology Build

Generate the Protocol Buffers Inter-Process Communications Code (IPC). This is code written into the Google Protocol Buffers that are translated into both the Go Language and Java. This step must be performed from the directory as shown in the following instructions.

```
Prompt> cd $HOME/PBCONF_Workspace/go/src/github.com/iti/pbconf
Prompt> make -C ontology protoc
Prompt> make -C ontology java
Prompt> make -C ontology test
```

Step 3 - Tests

PBCONF includes unit tests at various stages of development. While it is not required to run these tests, it is advised to do so. These tests should be performed from the top level of the installation source tree as follows.

Note that tests will require a few seconds to complete and will produce many lines of feedback. Briefly scan the lines of feedback. Tests that were completed successfully end with messages such as “PASS” or “OK”.

```
Prompt> cd $HOME/PBCONF_Workspace/go/src
Prompt> go test -v github.com/iti/pbconf/lib/pbdevice
Prompt> go test -v github.com/iti/pbconf/lib/pbnode
Prompt> go test -v github.com/iti/pbconf/lib/pbchange
Prompt> go test -v github.com/iti/pbconf/lib/pbpolicy
Prompt> go test -v github.com/iti/pbconf/lib/pbreports
Prompt> go test -v github.com/iti/pbconf/lib/pbontology
```

Step 4 - Static Analysis

Static analysis ensures that the code is syntactically correct and follows the best practices set forth by the respective language developers. It also ensures that deviations from standard practices are noted as intentional. Commands to execute the tests are provided as follows:

Note: the operations below do not provide any user feedback when they are successful.

```
Prompt> cd $HOME/PBCONF_Workspace/go/src

Prompt> go tool vet -printf=false github.com/iti/pbconf/lib/pbapi

Prompt> go tool vet -printf=false github.com/iti/pbconf/lib/pbauth

Prompt> go tool vet -printf=false
github.com/iti/pbconf/cmd/pbbroker

Prompt> go tool vet -printf=false
github.com/iti/pbconf/lib/pbchange

Prompt> go tool vet -printf=false github.com/iti/pbconf/cmd/pbconf

Prompt> go tool vet -printf=false
github.com/iti/pbconf/lib/pbdatabase

Prompt> go tool vet -printf=false
github.com/iti/pbconf/lib/pbdevice

Prompt> go tool vet -printf=false
github.com/iti/pbconf/lib/pbglobal

Prompt> go tool vet -printf=false
github.com/iti/pbconf/lib/pbinternode

Prompt> go tool vet -printf=false
github.com/iti/pbconf/lib/pblogger

Prompt> go tool vet -printf=false github.com/iti/pbconf/lib/pbnode

Prompt> go tool vet -printf=false -unreachable=false
github.com/iti/pbconf/lib/pbpolicy

Prompt> go tool vet -printf=false -unreachable=false
github.com/iti/pbconf/lib/pbreports

Prompt> go tool vet -printf=false -unreachable=false
github.com/iti/pbconf/lib/pbtranslate

Prompt> go tool vet -printf=false
github.com/iti/pbconf/lib/pbtransport

Prompt> go tool vet -printf=false
github.com/iti/pbconf/cmd/drivers/sel421

Prompt> go tool vet -printf=false
github.com/iti/pbconf/cmd/drivers/linux
```

Step 5 - Create the Installer Script

The PBCONF source includes scripts to generate an installer. The purpose of an installer is to facilitate the creation of PBCONF nodes on additional Linux machines without having to build PBCONF each time.

The first step in creating the installer is to instantiate the encryption support package. Note that the command below may result in a delay before the prompt returns. There is no feedback when the command is successful.

```
Prompt> cd $HOME/PBCONF_Workspace/go/src/github.com/iti/pbconf  
Prompt> go get -u golang.org/x/crypto/bcrypt
```

Modify the code used to generate password hashes to use the official Google bcrypt code. This operation can be performed using the following command. Note that there is no feedback when the command is successful.

```
Prompt> sed -i  
"s|github.com/btcsuite/golangcrypto/bcrypt|golang.org/x/crypto/bcrypt|" support/passgen.go
```

Step 6 – Run the Installer Script

At this point, a file named [build-installer.sh](#) has been created. This script file includes Go commands and other command line instructions that, when executed, will build a portable installer.

Run the installer script as shown in the two steps below.

```
Prompt> cd $HOME/PBCONF_Workspace  
/go/src/github.com/iti/pbconf/support/installer  
  
Prompt> bash ./build-installer.sh
```

A successful operation requires several seconds to complete and generates many lines of feedback. The script will generate a file named [pbconf-installer.sh](#) in the same directory. This file is the installer, which can be copied to any computer of the same type as the build platform (Linux amd64, for example) and executed to install PBCONF. Instructions for doing so are included in Appendix A.

4

INSTALLING PBCONF AND PROVISIONING NODES

The following instructions will guide the user through installation and provisioning of two PBCONF nodes that will later be connected in a hierarchy. The sample installation expects that two virtual machines (VMs) are instantiated and utilize the Ubuntu 16.04 LTS operating system. See Appendix A for instructions on provisioning additional VMs for PBCONF using the installer.

The example in this chapter describes how to create the following two nodes:

Table 4-1 Two nodes to be setup during this exercise

Node Name	Function	IP Address
my-pbconf-m1	Master node	192.168.0.13
my-pbconf-s1	Slave node	192.168.0.19

Step 1 – Install PBCONF

From the command line, execute the following installation command:

```
Prompt> sudo bash ./pbconf-installer.sh
```

If successful, the user will be prompted for input to provision a new node. Instructions follow.

Step 2 – Provision the Node

Setting up a PBCONF node requires the user to establish a small number of basic parameters such as the node identification, login credentials, and communication ports. These inputs are described below.

Node Name

Node Name will be used by other system components when referencing the specified node. To follow the examples provided in this user's guide, the first node to be provisioned is named "my-pbconf-m1", as shown in the example to follow.

Initial Admin User, Password and Email

Admin user, password and email are used to establish the user's credentials for logging into the node. The default username is "root", the default password is "changeme", and

the default user email address is root@localhost. Any of these can be changed as desired or they can be accepted by pressing ENTER at each of the three prompts.

API Listener Address

The user can interface with the PBCONF system through three different interfaces, one of which is through an API. The API Listener Address field lets the user select a port number to be appended to the node's IP address, through which the API will listen. The default value of 8080 can be changed as desired or it can be accepted by pressing ENTER at the prompt.

SSH Broker Listener Address

The SSH Broker acts as a bridge between the user and the device for interactive access, brokering between secure (PBCONF) and the end device. PBCONF currently supports SSH for the secure connection side, and for the end device, it supports SSH, Telnet and Serial. The default port number for the SSH Broker Listener is 8022. The default value can be changed as desired or it can be accepted by pressing ENTER at the prompt.

Enable Web Interface

The general user interface for the PBCONF system is a web interface. This option is enabled by default. Pressing ENTER at the prompt accepts this default configuration.

Web Interface Listener Port

By default, the web interface listener port number is 443. The default value can be changed as desired or it can be accepted by pressing ENTER at the prompt.

A sample of the prompts and user responses are represented below. Values in parentheses are part of the prompt and are the default values to be used if the user does not enter an alternative value. The user selects to use the default value by pressing the Enter key instead of specifying a value. For this working exercise, the default values are used except for the Node Name, which is changed to "my-pbconf-m1".

```
Node Name (Ubuntu): my-pbconf-m1
Initial Admin User (root): root
Initial Admin User Password (changeme): changeme
Initial Admin User email (root@localhost): <your email address>
API Listener address (:8080): 8080
SSH Broker Listener Address (:8022): 8022
Enable Web Interface (yes): yes
Web Interface Listener Port: (:443): 443
```

After defining the initial node parameters, the following prompts for IP addresses will appear. Enter the information as prescribed below. Note that the IP address 192.168.0.13 is being assigned to my-pbconf-m1 in this setup example. When installing the second node, named “my-pbconf-s1”, on a separate virtual machine, repeat as shown below, but assign the IP address 192.168.0.19 to my-pbconf-s1.

```
Any other SAN to attach to the certificate? Press enter when
done:127.0.0.1

Any other SAN to attach to the certificate? Press enter when
done:192.168.0.13

Any other SAN to attach to the certificate? Press enter when done:
<Press Enter>
```

Step 3 – Update the PBCONF Configuration File

As part of the installation process, the file “*pbconf.conf*” is created and placed in the “*/etc/pbconf*” directory. This file must be updated to deactivate the “*/tmp*” text from the file.

This can be completed from the command line as follows:

```
Prompt> sudo sed -i "/\//tmp/s/^/# /" /etc/pbconf/pbconf.conf
```

Note that there is no response if the command is executed successfully.

Alternatively, this operation can be completed by manually commenting out each line that contains the “*/tmp*” text.

The section of code that is affected is located under the Service Manager heading. Two lines need to be commented out by adding the comment symbol “#”. When completed, the affected section should appear as shown below.

```
[service-manager]
# location of the policy engine (undefined if internal)
# policy-engine=//tmp/pb/pe
# location of the change management engine (undefined if internal)
# change-management-engine=/tmp/pb/cme
# location of the ssh broker engine (undefined if internal)
sshbroker = //usr/bin/pbbroker
# location of the translation engine (undefined if internal)
# translation-engine = //tmp/pb/te
```


5

RUNNING PBCONF

At this step, all of the information required to bring a PBCONF node online has been provided. This chapter provides instructions for running PBCONF and interacting with it using the web user interface. Additionally, a second node will be brought online and a hierarchy will be established between the two nodes.

Step 1 - Run PBCONF

From the command line, run PBCONF and provide it the path of the configuration file. This will require super-user privileges (sudo).

```
Prompt> sudo pbconf -c /etc/pbconf/pbconf.conf
```

An example of a successful result is shown below. A few warnings and errors will exist and can be ignored. Note that the last line, “Registering namespace endpoints” indicates a successful operation. (Note that the command prompt is not available while PBCONF is running. To end PBCONF and restore the command prompt, press CTRL+C). With PBCONF running, it can now be accessed through a web browser.

```
2018-02-23T09:50:24.443-08:00 Main (INFO): Initializing System
2018-02-23T09:50:24.443-08:00 Main (INFO): loading Config file
2018-02-23T09:50:24.443-08:00 Main (INFO): Opening Database Connection
2018-02-23T09:50:24.443-08:00 Main (INFO): Loading Change Management Engine
2018-02-23T09:50:24.443-08:00 Main (INFO): Loading API manager
2018-02-23T09:50:24.443-08:00 Main (INFO): Loading Policy Engine
2018-02-23T09:50:24.444-08:00 Main (INFO): Using Internal pol engine
2018-02-23T09:50:24.444-08:00 Main (INFO): Loading Secure Connection Broker
2018-02-23T09:50:24.444-08:00 Main (INFO): Loading Translation Engine
2018-02-23T09:50:24.445-08:00 Main (INFO): Loading translation Modules
2018-02-23T09:50:24.47-08:00 HTTP API (DEBUG): startAPIhandler()
2018-02-23T09:50:24.472-08:00 HTTP UI (DEBUG): startwebUIhandler()
2018-02-23T09:50:24.475-08:00 Main (WARNING): Force Verify is on, but no
signature defined for /etc/pbconf/drivers/linux. Not loading
2018-02-23T09:50:24.476-08:00 Main (INFO): Starting API Handler
2018-02-23T09:50:24.506-08:00 Device API (DEBUG): Registering device
endpoints
2018-02-23T09:50:24.521-08:00 Node API (INFO): Registering node endpoints
2018-02-23T09:50:24.522-08:00 Policy API (INFO): Registering policy endpoints
2018-02-23T09:50:24.526-08:00 Report API (INFO): Registering reports
endpoints
2018-02-23T09:50:24.578-08:00 Main (INFO): Loading Translation Modules
2018-02-23T09:50:24.58-08:00 Database access (INFO): Report timer value = 7s
2018-02-23T09:50:24.581-08:00 Database access (DEBUG): GetNodeConfigElement
Error: sql: no rows in result set
2018-02-23T09:50:24.581-08:00 Database access (DEBUG): Could not recover a
timeout to check on child nodes, using default of 3
```

```
2018-02-23T09:50:24.581-08:00 Heartbeat (DEBUG): timeout value for 21s
2018-02-23T09:50:24.611-08:00 Database access (INFO): Report timer value = 5s
2018-02-23T09:50:24.611-08:00 Database access (DEBUG): GetNodeConfigElement
Error: sql: no rows in result set
2018-02-23T09:50:24.612-08:00 Database access (DEBUG): Could not recover a
timeout to check on child nodes, using default of 3
2018-02-23T09:50:24.741-08:00 CME:Main (DEBUG): Called with limit >1<
2018-02-23T09:50:24.758-08:00 CME:Main (DEBUG): Called with limit >1<
2018-02-23T09:50:24.81-08:00 CME:Main (DEBUG): Called with limit >1<
2018-02-23T09:50:24.843-08:00 Namespace API (DEBUG): Registering namespace
endpoints
```

Step 2 – Set up the IP address of the host machine

It will be necessary to assign a static IP address to the host Linux machine. The IP address will be the same that was assigned to the PBCONF node during setup (refer to Table 4-1). The IP configuration can be made through Ubuntu, using the Networking settings, or through the command line.

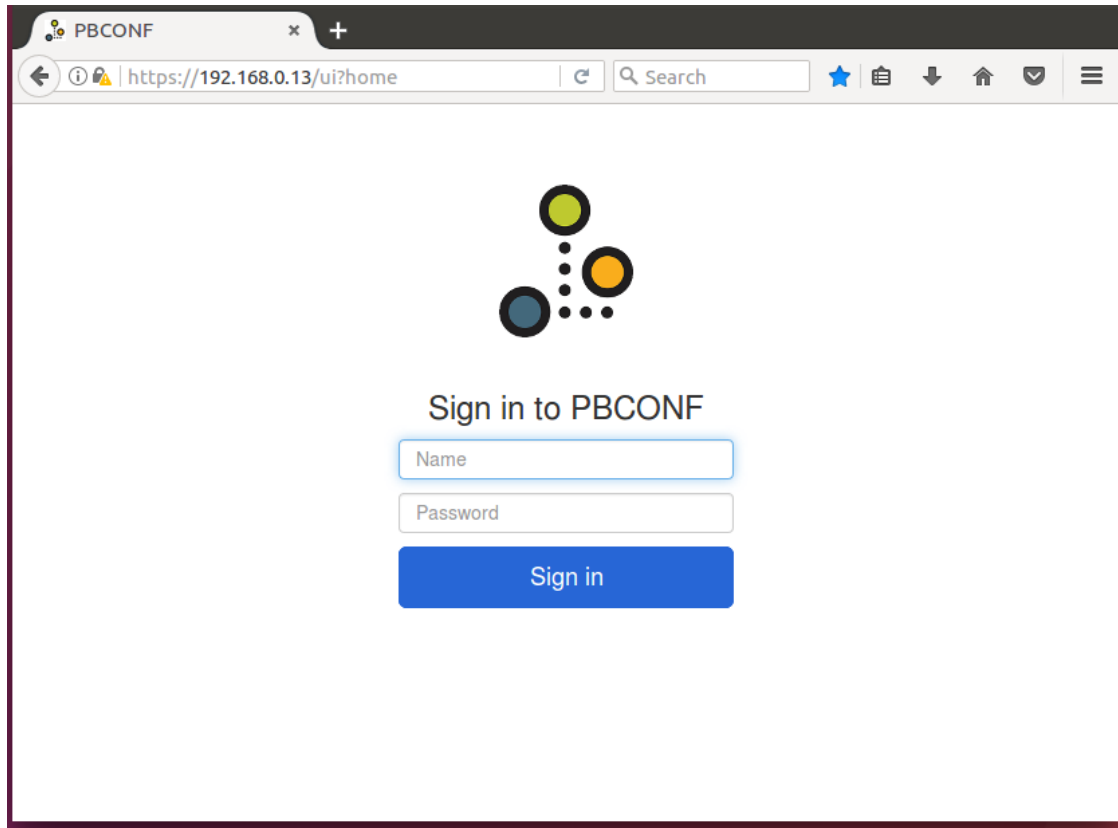
Step 3 - Access the web-based user interface

If the web interface was enabled during the setup of the node (it is enabled by default), open a browser window and navigate to

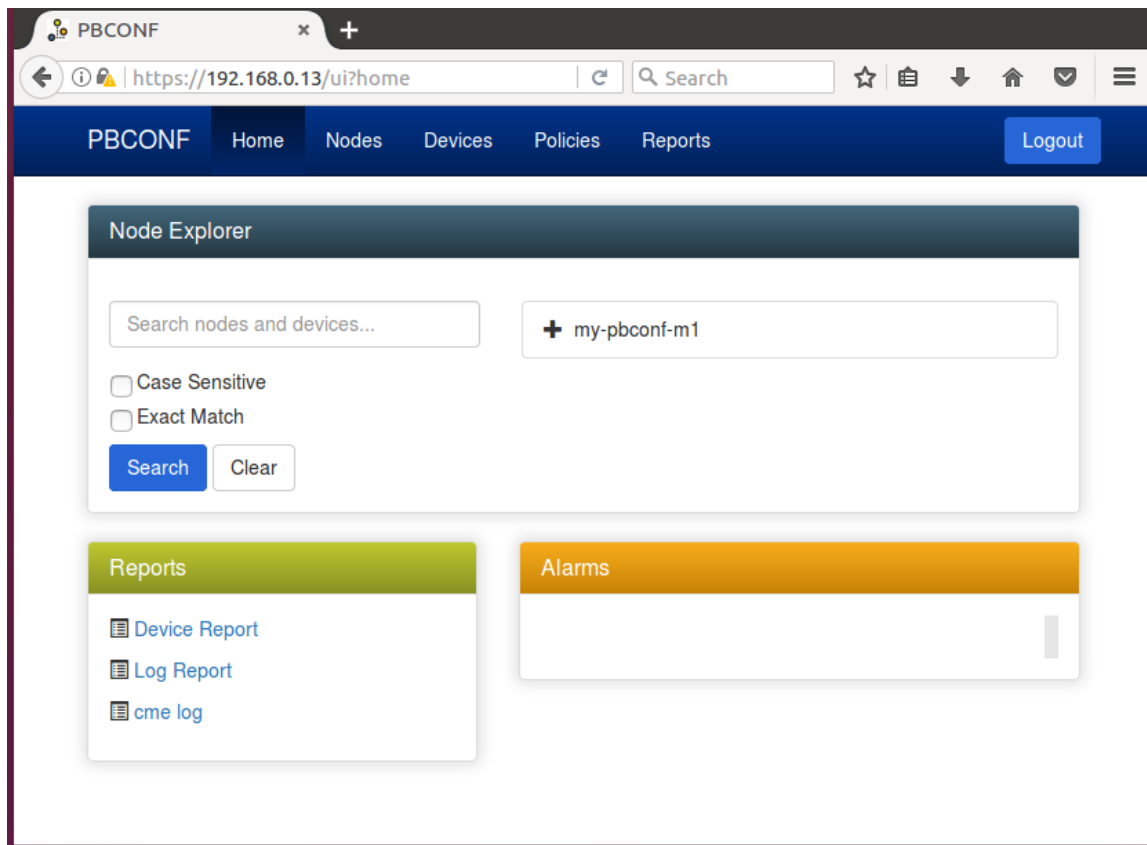
<https://192.168.0.13/ui> (substituting the IP address assigned to the PBCONF node if different than this example).

If a port other than default port for https (443) was assigned during node setup, include that in the address (for example, <https://192.168.0.13:xxxx/ui>, where xxxx is the chosen port). If using the default port, the port number can be omitted.

If the browser connects successfully, it will warn that the connection is not secure and/or that the security certificate for the site is invalid. Each browser handles this condition differently. The user should find the option which allows an exception (or exemption) for that site. Accept the exception according to the browser-specific instructions (not listed here). When this one-time operation is complete, the browser will connect and will present a login screen as shown below.



Login using the PBCONF initial admin user name and password. The default is “root” with password “changeme”. If the login is successful, the web application will present the home page of the PBCONF application as shown below.



The master node is now configured and running. Repeat the installation procedure on another virtual machine that will host the slave node.

Continuing to follow this example, the slave node, my-pbconf-s1, is assigned the IP address 192.168.0.19 and its host machine is also configured with the same static IP address.

Step 4 – Exchange security certificates

This step assumes that two nodes are running. Before the two nodes will communicate with each other, the public certificates of the two nodes need to be exchanged and put into each other's "trustedcerts" folder. The file transfer can be accomplished through a variety of means including flash drive, email, or SSH between the two host machines. The specific method is left to the user.

Below is shown a verification of the slave node's certificate (from its trustedcerts folder) copied into the trustedcerts folder of the master node and the filename changed from pbconf.pem to my-pbconf-s1.pem. Note: the filename of the certificate does not matter to PBCONF; the nodes will trust all certificates that are placed in the trustedcerts folder. The filename is changed only to identify it for the benefit of troubleshooting.

```
root@ubuntu: /etc/pbconf/trustedcerts
root@ubuntu:~# cd /etc/pbconf/trustedcerts
root@ubuntu:/etc/pbconf/trustedcerts# ls
my-pbconf-s1.pem  pbconf.pem
root@ubuntu:/etc/pbconf/trustedcerts#
```

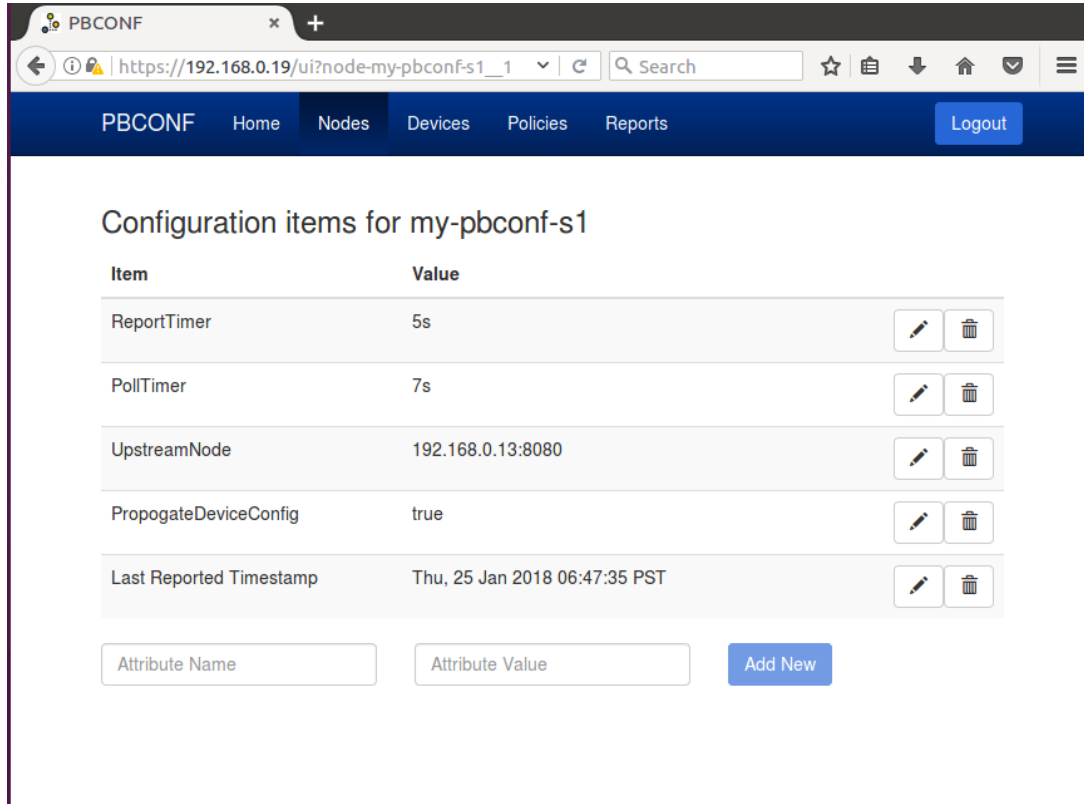
Next copy the master node's certificate and place it into the slave node's trustedcerts folder. Below is verification of this transfer, with the master certificate filename changed to my-pbconf-m1.pem.

```
root@ubuntu: /etc/pbconf/trustedcerts
root@ubuntu:~# cd /etc/pbconf/trustedcerts
root@ubuntu:/etc/pbconf/trustedcerts# ls
my-pbconf-m1.pem  pbconf.pem
root@ubuntu:/etc/pbconf/trustedcerts#
```











Step 5 – Linking PBCONF nodes

Once the public certificates are copied and placed in the other node's trustedcerts folder, the nodes can be connected to each other through the web UI. From the slave node, click on the Nodes tab and then click on the slave node in the resulting list. The response will be a page with

the configuration items for this node. Find the configuration item called UpstreamNode and click the pencil icon on that line to edit the field. As shown below, enter the master node's IP address followed by the API port (8080 by default) of the master node. For example: "192.168.0.13:8080". Click on the checkmark to accept this change. This will connect the slave node to the master node at the specified address.

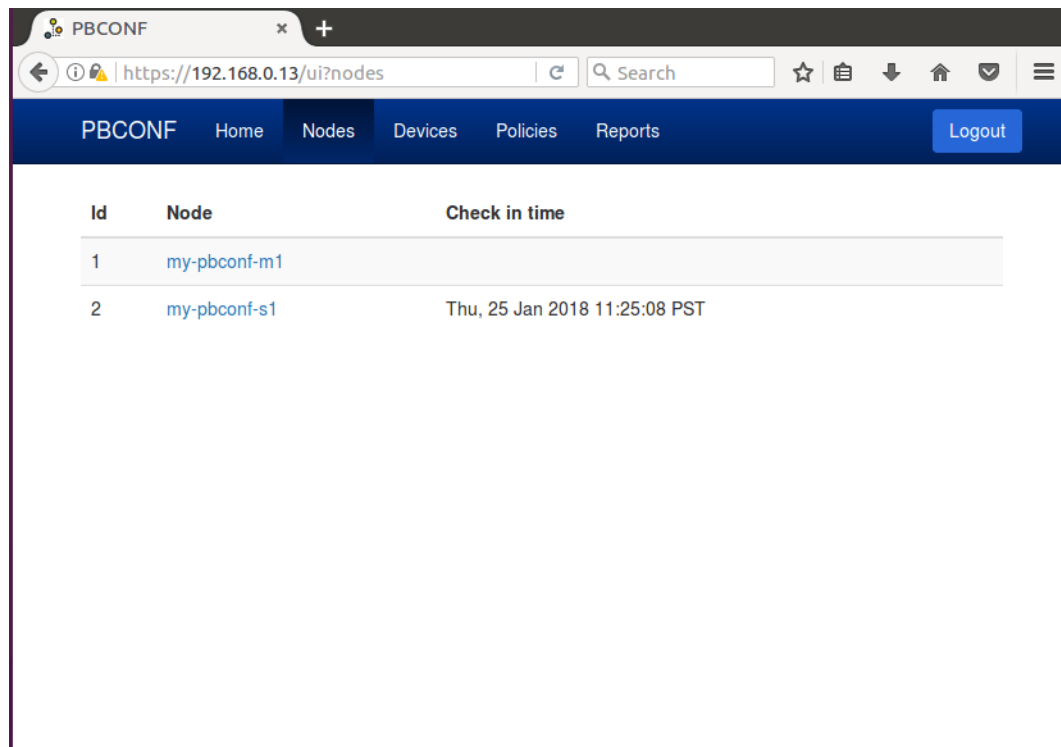


The screenshot shows the PBCONF web interface in a browser. The address bar displays `https://192.168.0.19/ui?node-my-pbconf-s1_1`. The navigation bar includes links for Home, Nodes, Devices, Policies, Reports, and a Logout button. The main content area is titled "Configuration items for my-pbconf-s1" and contains a table with the following data:

Item	Value	
ReportTimer	5s	 
PollTimer	7s	 
UpstreamNode	192.168.0.13:8080	 
PropagateDeviceConfig	true	 
Last Reported Timestamp	Thu, 25 Jan 2018 06:47:35 PST	 

Below the table, there is a form with two input fields: "Attribute Name" and "Attribute Value", followed by an "Add New" button.

To check that the connection is successful, go to the web UI on the master node and look in the nodes tab. The slave node should appear in the list of nodes along with the last time this node checked in. An example is shown below.



The screenshot shows a web browser window with the PBCONF application. The address bar shows the URL `https://192.168.0.13/ui?nodes`. The navigation bar includes links for Home, Nodes, Devices, Policies, and Reports, along with a Logout button. The main content area displays a table with the following data:

Id	Node	Check in time
1	my-pbconf-m1	
2	my-pbconf-s1	Thu, 25 Jan 2018 11:25:08 PST

At this point, a simple hierarchy of PBCONF nodes has been established. A similar procedure is required for building the tree with additional nodes.

6

LIST OF ABBREVIATIONS

IP	- Internet Protocol
IPC	- Inter-Process Communications
JDK	- Java Development Kit
JRE	- Java Runtime Environment
LTS	- Long Term Support
OS	- Operating System
SSH	- Secure SHell
VM	- Virtual Machine

A

CREATING ADDITIONAL NODES WITH THE PBCONF INSTALLER

Chapters 2 and 3 of this guide instruct the user on building PBCONF for the particular environment. The procedure includes instructions for a distributable installer package so that PBCONF can be installed on additional machines of the same type and platform without having to rebuild each time. This appendix describes the prerequisites and the instructions for copying and using the installer package to set up additional PBCONF nodes on other host machines.

Assumptions and Prerequisites

- The target machine is of the same type as the build platform (e.g. Linux, amd64).
- The operating system of the target machine is Ubuntu 16.04 LTS, which is the same as for the machine on which the installer is built.

Step 1 – Install Dependencies

PBCONF relies on services provided by software packages SSH, SQLite3, Git, Unzip, and Java Development Kit. These can be installed in one step from the command line as follows:

```
Prompt> sudo apt-get update && sudo apt-get install -y build-essential ssh sqlite3 git unzip default-jdk
```

If successful, this operation will take several seconds to complete and will generate many lines of feedback.

Step 2 – Setup Git

Set up the Git environment if not already done on the host machine.

```
Prompt> git config --global user.email <you@example.com>  
Prompt> git config --global user.name <Your Name>
```

Note: the reference to global in the instructions below requires 2 dashes (--global).

Note: the operations below do not result in any user feedback when they are successful.

Step 3 – Transfer and Install

After building PBCONF on the original machine, the installer package is created in folder:

[*\\$HOME/PBCONF_Workspace/go/src/github.com/iti/pbconf/support/installer*](#)

The filename is: [*pbconf-installer.sh*](#)

Copy this file and transfer it to the Downloads folder on the new machine.

After the file has been copied, set the current directory to Downloads in preparation to execute the file.

```
Prompt> cd Downloads
```

At this step, the host machine is ready for installation of PBCONF. Continue to Chapter 4 and then to Chapter 5 of this document to provision the node, run PBCONF, and connect this node to other nodes in the hierarchy.