MicroClimate™ Management System [MCMS]
Installation Guide

Juniper Networks, Inc.
1133 Innovation Way
Sunnyvale, CA 94089

Phone: 888.JUNIPER (888.586.4737)
Web: https://support.juniper.net/support/requesting-support/

WARNING !

Refer to Install Guide before Installation

Warranty Notice: Device Attenuation Required

Do not connect OLT directly to ONUs without proper attenuation. PON transceivers will be **permanently damaged** unless connected with minimum 16dB attenuation (20dB recommended)
Damage from optical overload will void warranty.

Combination of attenuator and splitters can provide required attenuation -- see example:

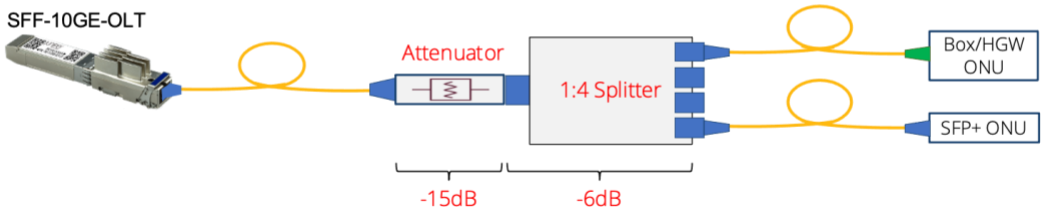

Table of Contents

Introduction

The MicroClimate™ Management System (MCMS) is the management solution for Tibit PON networks. The MCMS architecture is shown in Figure 1 and consists of the MCMS PON Manager graphical user interface, MCMS Netconf Server, and MCMS PON Controller. Together these components provide a complete network management solution for provisioning and monitoring MicroPlug™ OLT devices, as well as the subtended MicroPlug™ ONU devices and third-party ONUs compliant with the 10G-EPON and XGS-PON standards.

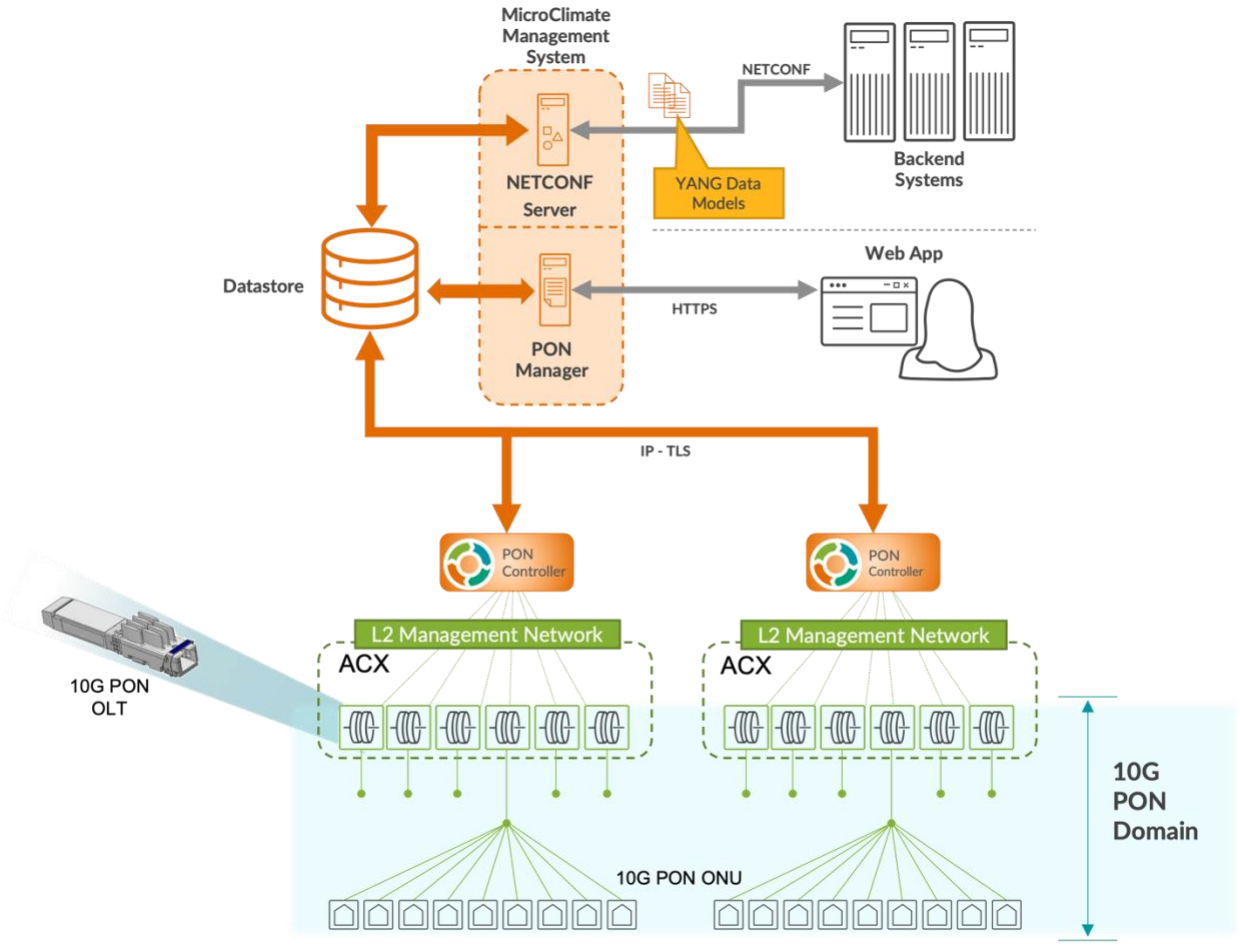

Figure 1: MicroClimate™ Management System Architecture

MCMS PON Manager

The MCMS PON Manager is a single-page web application (Web App) and an accompanying REST API that provides a graphical user interface for managing the PON Network. The Web App is built on the Angular web application framework, which provides an HTML and JavaScript front-end user interface. The REST API accompanies the Web App for the purposes of providing access to MongoDB for managing MCMS PON Manager users and the PON Network.

The MCMS PON Manager has the following features:

- Alarm management.
- Dashboard view with a summary of PON network conditions.
- Device monitoring and statistics.
- Device provisioning and management.
- Logging for diagnostics and troubleshooting.
- MCMS PON Controller database management.
- MCMS PON Manager user management.
- Polyglot graphical OMCI (and future 10G EPON OAM) service configuration tool.
- Service configuration, including VLANs, Service Level Agreements (SLAs), 802.1X Authentication, and DHCP Relay.

Note that the REST API should only be used with the MCMS PON Manager. At this time, the REST API is not a published interface and will change in future releases. Future versions for MCMS PON Manager may make the REST API available directly to customer applications.

MCMS Netconf Server

The MCMS Netconf Server provides a standard Netconf interface and customer facing API for managing the PON network. The Netconf Server is built on the Netopeer2 and Sysrepo open-source architecture and interfaces with MongoDB. The Netconf solution supports standard Broadband forum (BBF) TR-383 and TR-385 YANG models for configuring subscriber services for the PON network. In addition to standard YANG models, MCMS YANG models provide a complete Netconf management solution for PON Controllers, OLTs, and ONU devices.

MCMS PON Controller

The MCMS PON Controller is a stateless management controller and device driver application for configuring and monitoring the end points in a Tibit MicroPlug™ OLT PON network. MongoDB serves as the northbound application programming interface for the PON Controller. The PON Controller applies configuration to OLT and ONU devices from documents stored in

MongoDB. The MCMS PON Controller uses IEEE 1904.2 packets (L2) to communicate with the OLT and ONU devices. The PON Controller also collects state information, statistics, and logs from devices and reports the information to higher layer applications through MongoDB. The PON Manager and Netconf interfaces manage the PON Controller through MongoDB.

MongoDB Datastore

The Mongo database (MongoDB) provides the datastore for the MicroClimate™ Management System. The MongoDB datastore contains all the configuration, state, statistics, alarms, and logging data for the devices in the PON network. Northbound interfaces, such as the MCMS PON Manager, MCMS Netconf Server, and customer applications interface with MongoDB to provision and retrieve monitoring information for devices in the PON network. MongoDB serves as the interface between the PON Manager and Netconf and the PON Controller.

Provisioning data generally flows "downstream" through the management network. The PON Manager and Netconf interfaces write device configuration to MongoDB. The PON Controller reads the configuration data from MongoDB and programs the OLT and ONU devices accordingly.

Monitoring data, including device state, statistics, alarms, and logging, is collected and flows "upstream" through the management network. The PON Controller periodically collects state information from devices in the PON network and writes the monitoring data to MongoDB. The PON Manager reads the monitoring data from MongoDB for display in the Web App.

Server Sizing

Lab Testing

It is recommended to run all the MCMS PON Management applications on a single VM (or single server).

- Management Applications
  - Pon Controller
  - Pon Manager
  - MongoDB
  - Netconf server

- Server Hardware Requirements
  - 2 vCPUs
  - 8 GB RAM
  - 20 GB disk space

- PON Size
  - 4 OLTs
  - 32 ONUs

Production Deployments

It is recommended to run each of the MCMS PON Management applications on separate VMs (or servers). The following table provides guidelines for deployments.

Note: The PON Controller sizing is for the PON Controller process only. It does not include the base OS (Ubuntu 18.04). The sizing for PON Manager, Netconf and MongoDB below does include the base OS (Ubuntu 18.04).

Application	Subscriber Size	
	3072 Subscriber **48 OLTs and 3072 ONUs (max. on 1 switch)**	**100K Subscribers**
PON Controller[1]	1 Instance: 1 vCPU 1 GB RAM 1 GB disk space	32 x Pon Controller Instances: 32 vCPU 32 GB RAM 32 GB disk space
PON Manager	2 vCPU 8 GB RAM 10 GB disk space	8 vCPU 32 GB RAM 50 GB disk space
Netconf[2]	1 Instance: 2 vCPU 8 GB RAM 10 GB disk space	3 x Netconf Instances: 6 vCPU 24 GB RAM 30 GB disk space
MongoDB	2 vCPU 8 GB RAM 10 GB disk space	16 vCPU 64 GB RAM 100 GB disk space

[1] A single PON Controller instance supports 3072 Subscribers (a single switch with 48 OLTs x 64 ONUs/OLT) and requires <=5 ms latency (round trip) between the PON Controller and OLTs. For 100K subscribers, 32 instances are required.

[2] Size of typical Netconf server entity: <=15 switches populated with OLTs. A single Netconf Server entity supports up to 45K subscribers (3K x 15). For 100K subscribers, 3 instances are required.

MongoDB Installation

MongoDB serves the purpose of being the datastore for the MicroClimate™ Management System. This includes all PON-specific information as well as user models. This section describes how to install the MongoDB software on a server running 18.04.

Installation

To install the mongoDB Community Edition,

1. Follow this link to the official mongoDB website: https://docs.mongodb.com/manual/tutorial/install-mongodb-on-ubuntu/
2. Using the table of contents at the top of the page, select the option; "Uninstall MongoDB Community Edition", and follow the instructions to uninstall mongoDB if installed.
3. Using the same table of contents at the top of the page, select the option; "Install MongoDB Community Edition using .deb Packages", and follow the instructions to install the latest version of mongoDB.
    - Ensure that you are using the "Ubuntu 18.04 (Bionic)" options.

MongoDB Scripts

Alternatively, these scripts are included with the MCMS PON Manager and can automate installing, starting, and uninstalling MongoDB.

Installation steps:

From a Linux shell, unpack the R2.1.2-PonManager.zip file and change to the new unpacked directory.

```
unzip UnifiedPON-R2.1.2-MicroClimate-PonManager.zip
cd UnifiedPON-R2.1.2-MicroClimate-PonManager
```

Note: If unzip is not already installed on the ubuntu system, run the following command to install:

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

mongodb_install.sh

This script will uninstall the current version of MongoDB and then install and start the latest version of MongoDB server Community Edition.

- Supported Operating Systems:
    a. Ubuntu 18.04
- Dependencies:
    a. systemctl
- Run the command:

```
sudo bash ./mongodb_install.sh
```

mongodb_start.sh

This script will start a forked instance of a MongoDB server

- Supported Operating Systems:
  a. Ubuntu 18.04
- Dependencies:
  a. systemctl
- Run the command
  sudo bash ./mongodb_start.sh

Enabling MongoDB Authentication and TLS/SSL

This section describes how to enable a replica set in MongoDB using keyfile authentication and TLS/SSL between replica set members. With keyfile authentication, each mongod instance in the replica set uses the contents of the keyfile as the shared password for authenticating other members in the deployment. Only mongod instances with the correct keyfile can join the replica set.

Note: With Pon Controller R2.0.4, only a single member replica set is supported. When the Pon Controller is upgraded to R2.1.0 or greater, additional mongod instances can be added to the existing replica set.

In this document, the single mongod instance is referenced as:

Replica Set Member	Hostname
Member 0	mongodb0.domain.net

Important: In the following steps for configuration of the MongoDB server, please use the FQDN for your MongoDB server in place of *mongodb0.domain.net*. If you do not have a DNS server in your test environment, it is recommended to use the server hostname in place of *mongodb0.domain.net*.

Mongo Keyfile Authentication and User Authentication

Step 1: Create a keyfile

The following operation uses `openssl` to generate a complex pseudo-random 1024-character string to use as a shared password.

```
$ openssl rand -base64 756 > mongodb-keyfile
$ sudo chown mongodb:mongodb mongodb-keyfile
$ sudo chmod 400 mongodb-keyfile
$ sudo cp -p mongodb-keyfile /etc/ssl/
```

Note: “mongodb-keyfile” is an example filename. You can select any filename of your choice. Be sure to remember this filename for the following steps.

Step 2: Start mongodb0 with access control enabled

For each member of the replica set, update the mongod configuration file (/etc/mongod.conf) with the following parameters:

```
$ sudo vi /etc/mongod.conf

   security:
     keyFile: /etc/ssl/mongodb-keyfile

   replication:
     replSetName: rs0

   net:
      bindIp: localhost,mongodb0.domain.net
```

Restart mongod process:

```
$ sudo systemctl restart mongod.service
```

Step 3: Connect to mongodb0 instance over the localhost interface

Connect a mongo shell mongodb0 instance over the localhost interface. You must run the mongo shell on the same physical machine as the mongod instance.

The localhost interface is only available since no users have been created for the deployment. The localhost interface closes after the creation of the first user.

```
$ mongo
MongoDB shell version v4.0.27
connecting to:
mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("c71844ea-05b4-4586-944b-
61d211f6a875") }
MongoDB server version: 4.0.27
>
```

Step 4: Initiate the replica set

From the mongo shell, run the rs.initiate() method using the example configuration settings:

```
rs.initiate()
```

Step 5: Create the user administrator

```
admin = db.getSiblingDB("admin")
admin.createUser(
  {
    user: "admin",
    pwd: "changeme",
    roles: [ { role: "root", db: "admin" } ]
  }
)
```

Note: The password *"changeme"* is an example password. Please create a unique password for the user.

Step 6: Authenticate as the user administrator

Connect a new mongo shell to the primary replica set member:

```
$ mongo -u admin -p
```

Note: If you do not specify the password to the -p command-line option, the mongo shell prompts for the password.

Step 7: Create users for Tibit Pon Controller, Pon Manager and Netconf

Reconnect to mongo instance and create users in “tibit_users” database.

```
use tibit_users
db.createUser(
  {
    user: "pdmPonController",
    pwd:  "pdmPass",
    roles: [ { role: "readWrite", db: "tibit_pon_controller" } ]
  }
)
db.createUser(
  {
    user: "pdmNetconf",
    pwd:  "pdmPass",
    roles: [ { role: "readWrite", db: "tibit_pon_controller" } ,
             { role: "readWrite", db: "netconf_db" } ]
  }
)
db.createUser(
  {
    user: "pdmPonManager",
    pwd:  "pdmPass",
    roles: [ { role: "readWrite", db: "tibit_pon_controller" },
             { role: "readWrite", db: "tibit_users" } ]
  }
)
db.createUser(
  {
    user: "pdmUserAdmin",
    pwd:  "pdmPass",
    roles: [ { role: "readWrite", db: "tibit_users" } ]
  }
)
```

Note: The password "pdmPass" is an default password. This password can be changed or left “as is”.

Step 8: Verify user access into Mongo

Ensure the new username/passwords successfully access mongo.

```
$ mongo --authenticationDatabase=tibit_users -u pdmPonManager -p
$ mongo --authenticationDatabase=tibit_users -u pdmUserAdmin -p
$ mongo --authenticationDatabase=tibit_users -u pdmPonController -p
$ mongo --authenticationDatabase=tibit_users -u pdmNetconf -p
```

Mongo TLS/SSL

This section describes how to create a private certificate authority using OpenSSL.

Step 1: Create a private root key for the self-signed certificate

```
$ openssl genrsa -des3 -out rootCA.key 2048
Generating RSA private key, 2048 bit long modulus (2 primes)
.....................................................................
...........+++++.....................................................+++++ e is 65537
(0x010001)
Enter pass phrase for rootCA.key:
Verifying - Enter pass phrase for rootCA.key:
```

Note: If you see an error with "`Can't load /home/user/.rnd into RNG`", run the following command and then redo Step 1. (Replace "/home/user" with the home directory of the current user)

```
$ sudo openssl rand -out /home/user/.rnd -hex 256
```

Step 2: Self-sign the root certificate

```
$ openssl req -x509 -new -nodes -key rootCA.key -sha256 -days 1024 -out rootCA.pem
Enter pass phrase for rootCA.key:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:
State or Province Name (full name) [Some-State]:
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:
Email Address []:
```

Optional: A second option for self-signing the root certificate is using the following command using the -subj option:

```
$ openssl req -x509 -new -nodes -key rootCA.key -sha256 -days 1024 -out rootCA.pem
-subj "/C=AU/ST=State/L=City/O=Organization/OU=Department/CN="
```

Copy the root certificate to the /etc/ssl/ directory:

```
$ sudo cp rootCA.pem /etc/ssl
```

Step 3: Create a certificate for the MongoDB server

```
$ openssl genrsa -out mongodb0.key 2048
Generating RSA private key, 2048 bit long modulus (2 primes)
.....................+++++
.............+++++
e is 65537 (0x010001)
```

Step 4: Create a config file (.cnf) with SAN extensions for use with OpenSSL

Create a config file for the mongodb server using your favorite editor and ensure it contains the following lines:

Note: Input the IP address of mongo server. This will allow connecting to the MongoDB server with either the DNS name or the IP address. If you do not have a DNS server in your test environment, it is recommended to use the server hostname in place of *mongodb0.domain.net* below.

```
$ vi mongodb0.cnf
       subjectAltName = @alt_names

       [alt_names]
       DNS.1 = mongodb0.domain.net
       IP.1 = 10.0.0.1
```

Step 5: Generate the Certificate Signing Request (CSR)

Important: If you do not have a DNS server in your test environment, it is recommended to use the server hostname in place of *mongodb0.domain.net* for the “Common Name” below.

```
$ openssl req -new -key mongodb0.key -out mongodb0.csr
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:
State or Province Name (full name) [Some-State]:
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []: mongodb0.domain.net
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

Note: A second option for generating the Certificate Signing Request (CSR) is using the following command using the -subj option:

```
$ openssl req -new -key mongodb0.key -sha256 -days 1024 -out mongodb0.csr -subj
"/C=AU/ST=State/L=City/O=Organization/OU=Department/CN=mongodb0.domain.net"
```

Step 6: Self-sign the CSR

Self-sign the CSR for the mongodb server:

```
$ openssl x509 -req -in mongodb0.csr -CA rootCA.pem -CAkey rootCA.key -CAcreateserial
-out mongodb0.crt -days 500 -sha256 -extfile mongodb0.cnf
Signature ok
subject=C = AU, ST = Some-State, O = Organization, OU = Department, CN =
mongodb0.domain.net
Getting CA Private Key
Enter pass phrase for rootCA.key:
```

Note: This creates a signed certificate called mongodb0.crt which is valid for 500 days.

Step 7: Create the .pem file for the mongodb server

```
$ cat mongodb0.key mongodb0.crt > mongodb0.pem
$ sudo cp mongodb0.pem /etc/ssl/mongodb0.pem
```

Step 8: Enable SSL on the MongoDB Server

For Update the mongod configuration file (/etc/mongod.conf) with the “ssl:” settings below:

```
$ sudo vi /etc/mongod.conf

security:
  keyFile: /etc/ssl/mongodb-keyfile

replication:
  replSetName: rs0

net:
  bindIp: localhost,mongodb0.domain.net
  ssl:
     mode: requireSSL
     PEMKeyFile: /etc/ssl/mongodb0.pem
```

Restart mongod process:

```
$ sudo systemctl restart mongod.service
```

Step 9: Verify connectivity to MongoDB server using the PEM file

```
$ mongo --ssl --sslCAFile=/etc/ssl/rootCA.pem --host mongod0.domain.net
--authenticationDatabase=tibit_users -u pdmPonManager -p
```

For more information on configuring MongoDB, please reference:

https://docs.mongodb.com/v4.0/replication/
https://docs.mongodb.com/v4.0/core/authentication/

https://docs.mongodb.com/v4.0/tutorial/configure-ssl/
https://docs.mongodb.com/v4.0/tutorial/deploy-replica-set-with-keyfile-access-control/

PON Manager Installation

This section describes how to install, configure, and run the MCMS PON Manager software on a system running Ubuntu 18.04.

Requirements and Dependencies

Supported Browsers

Browser	Version
Firefox	85.0.0
Chrome	88.0.0

Supported Operating Systems

Operating System	Version
Ubuntu	18.04

Required Packages

The following Ubuntu 18.04 apt packages are installed by the Debian package.

Package	Version
Python	3.6.7
Python3-dev	3.6.7
Python3-venv	3.6.7
MongoDB	4.0.0
Pip3	9.0.0
Apache2	2.4.0
Systemd	2.3.7
Libapache2-mod-wsgi-py	4.5.17
openssl	1.1.1

Package	Version
ufw	0.36

Python Package Dependencies (pip3)

The following Python 3 packages are installed with pip3 by the Debian package within a MCMS PON Manager specific Python virtual environment.

Package	Version
blinker	1.4.0
dataclasses	0.6
Django	2.2.14
django-cors-headers	3.4.0
django-rest-framework	0.1.0
django-rest-swagger	2.2.0
djangorestframework	3.11.0
djongo	1.2.30
netifaces	0.10.9
openapi-codec	1.3.2
optional-django	0.1.0
Pillow	7.1.1
pymongo	3.11.2
pyserial	3.4
python-resize-image	1.1.19
python-memcached	1.59
pytz	2019.1
simplejson	3.16.0
sqlparse	0.2.4

Package Contents

The MCMS PON Manager software is provided as a .zip file. The contents of the package are described in the table below.

File/Directory	Description
tibit-ponmgr_R2.1.2_all.deb	Debian Package based MCMS PON Manager application
legacy_uninstall.py	Uninstallation script for MCMS PON Manager versions R1.2.0 and below
LICENSES.txt	Lists the licensing information for third party software used
mongodb_install.sh	Installs mongoDB
mongodb_start.sh	Starts forked mongod instance
mongodb_uninstall.sh	Removes mongodb application, logs, configuration, and databases
README.txt	Quick startup guide for installing and configuring PON Manager debian package
bulk_configure/	Directory containing bulk configuration utility
bulk_configure/bulk_configure.py	Bulk configuration Python 3 script
bulk_configure/olt_bulk_config.csv	File where OLT device info to be pre-provisioned is configured
bulk_configure/onu_bulk_config.csv	File where ONU device info to be pre-provisioned is configured
bulk_configure/README.txt	Guide for utilizing the bulk configuration utility
version.txt	Version information

Installation Methods

This section describes the steps to install and uninstall the MCMS PON Manager software using the provided installation scripts.

Note: Read all steps including the notes section prior to installing the software**.**

Debian Package

The tibit-ponmgr_R2.1.2_all.deb package handles installation of all dependencies except for Python.

Prerequisites:

- Ensure that you have an active internet connection.
- Install/upgrade existing python3 package to be equal to or greater than version 3.6.7.
- Install/upgrade existing pip3 package to be equal to or greater than version 9.0.0.
- Ensure that you have your MongoDB Server running that will manage user authentication and serve as your default database.
- Ensure you have the latest ubuntu updates

Installation steps:

1. From a Linux shell, unpack the .zip file and change to the new unpacked directory.

```
unzip UnifiedPON-R2.1.2-MicroClimate-PonManager.zip
cd UnifiedPON-R2.1.2-MicroClimate-PonManager
```

Note: If unzip is not already installed on the ubuntu system, run the following command to install:

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

2. Use apt-get to install the package.

```
sudo apt-get install ./tibit-ponmgr_R2.1.2_all.deb
```

Configuration steps:

Ensure that you have root/sudo access to modify files.

1. Navigate to the directory: '/etc/tibit/ponmgr/'
   - Within this directory, there are several PON Manager configuration files that will need to be edited with a text editor of your choice.
   - Copy the root certificate (rootCA.pem file from MongoDB Installation step 2) to the system running Pon Manager into the /etc/ssl/ directory and update the following configuration files to match the settings from MongoDB Installation section.

```
$ sudo vi /etc/tibit/ponmgr/user_database.json
```

```
{
    "host": "mongodb0.domain.net",
    "name": "tibit_users",
    "port": "27017",
    "auth_enable": true,
    "auth_db": "tibit_users",
    "username": "pdmUserAdmin",
    "password": "pdmPass",   << Use password from MongoDB Installation (if changed)
    "tls_enable": true,
    "ca_cert_path": "/etc/ssl/rootCA.pem",
    "dns_srv": false,
    "db_uri": "",
    "replica_set_enable": false,
    "replica_set_name": "rs0",
    "replica_set_hosts": []

}
```

```
$ sudo vi /etc/tibit/ponmgr/databases.json
{
  "Default": {
     "auth_db": "tibit_users",
     "auth_enable": true,
     "ca_cert_path": "/etc/ssl/rootCA.pem",
     "db_uri": "",
     "dns_srv": false,
     "host": "mongodb0.domain.net",
     "name": "tibit_pon_controller",
     "password": "pdmPass",  << Use password from MongoDB Installation (if changed)
     "port": "27017",
     "replica_set_enable": false,
     "replica_set_hosts": [
        "localhost",
        "127.0.0.1"
     ],
     "replica_set_name": "rs0",
     "tls_enable": true,
     "username": "pdmPonManager"
  }
}
```

Restart Pon Manager process (apache2):

```
$ sudo systemctl restart apache2.service
```

- user_database.json
    - Stores configuration for database to be used for user authentication

Key	Description
host	Hostname/IP Address of the MongoDB server hosting your user database.
name	Name of your user database. Default user database name is 'tibit_users'. *(If it doesn't exist, it will be created)*
port	MongoDB port number.
auth_enable	Boolean value determining if the MongoDB server at *host:port* is using authentication.
auth_db	Name of your MongoDB authentication database. *(Used when auth_enabled = true)*
username	The username of the MongoDB user to authenticate with. *(Used when auth_enabled = true)*
password	The password of the specified MongoDB user. *(Used when auth_enabled = true)*
tls_enable	Boolean value specifying whether the MongoDB server at *host:port* is using encryption.
ca_cert_path	The local path to the encryption certificate.
dns_srv	Boolean value specifying whether the MongoDB server is using a DNS seed list.
db_uri	Raw MongoDB connection URI. All Other fields are ignored if this is used.
replica_set_enable	Boolean value specifying if the MongoDB server is running as a replica set.
replica_set_name	The name of the MongoDB replica set. *(Used when replica_set_enabled = true)*
replica_set_hosts	List of hosts to be used as the MongoDB replica set. *(Used when replica_set_enabled = true)*

- databases.json
    - Stores list of all available PON databases in PON Manager

Key	Description
host	Hostname/IP Address of your mongoDB server hosting your PON Controller database
name	Name of your PON Controller database. Default PON Controller database name is 'tibit_pon_controller'. *(If it doesn't exist, it will be created)*
port	MongoDB server port number.
auth_enable	Boolean value determining if the MongoDB server at *host:port* is using authentication.
auth_db	Name of your MongoDB authentication database. *(Used when auth_enabled = true)*
username	The username of the MongoDB user to authenticate with. *(Used when auth_enabled = true)*
password	The password of the specified MongoDB user. *(Used when auth_enabled = true)*
tls_enable	Boolean value specifying whether the MongoDB server at *host:port* is using encryption.
ca_cert_path	The local path to the encryption certificate.
dns_srv	Boolean value specifying whether the MongoDB server is using a DNS seed list.
db_uri	Raw MongoDB connection URI. All Other fields are ignored if this is used.
replica_set_enable	Boolean value specifying if the MongoDB server is running as a replica set.
replica_set_name	The name of the MongoDB replica set. *(Used when replica_set_enabled = true)*
replica_set_hosts	List of hosts to be used as the MongoDB replica set. *(Used when replica_set_enabled = true)*

- recovery_email_configuration.json
    - Configures SMTP email server for sending user password recovery emails

Key	Description
host	Host to use for sending email
port	Port to use for the SMTP server defined above *Type: String*
use_tls	Use TLS connection *'true' or 'false'* *Type: String*
user	Username for the SMTP server
password	Password for user

Enabling HTTPS:

This section describes the steps to configure and enable HTTPS traffic to and from the MCMS PON Manager.

Note: If enabling HTTPS after initial installation, the Apache2 service will need to be restarted.

Prerequisites:

- You will need a valid certificate and private key

To enable HTTPS, these configuration files will need to be modified (located in /etc/tibit/ponmgr/):

1. tibitdev-api.conf
    a. Within this configuration file, there is a section titled; 'HTTPS Options'. Uncomment this section.

```
# SSLEngine On
# SSLProtocol all -SSLv3 -TLSv1 -TLSv1.1
# SSLCipherSuite HIGH:!aNULL:!MD5
# SSLCertificateFile [path to cert here]
# SSLCertificateKeyFile [path to key here]
# <FilesMatch '\.(cgi|shtml|phtml|php)$'>
# SSLOptions +StdEnvVars
# </FilesMatch>
# Directory /usr/lib/cgi-bin>
# SSLOptions +StdEnvVars
# </Directory>
# BrowserMatch 'MSIE [2-6]' nokeepalive / ssl-unclean-shutdown
downgrade-1.0 force-response-1.0
```

On lines 4 and 5 above, there are the placeholders: '[path to cert here]' and '[path to key here]'. Replace these placeholders, including removing the brackets, with the path to a valid certificate and private key, respectively.

2. tibitdev-web.conf
    a. Within this configuration file, there is a section titled; ‘HTTPS Options’. Commant the section out

```
ProxyPass /api http://127.0.0.1:8013/
```

    b. Within this configuration file, there is a section titled; ‘HTTPS Options’. Uncomment this section.

```
# RewriteRule ^(.*)$ https://%{HTTP_HOST}$1 [R-301,L]
```

    c. Within this configuration file, there is a section titled; ‘HTTPS virtualhost’. Uncomment this section.

```
#<VirtualHost *:443>
# ServerName tibitdev-web
# ServerAdmin webmaster@localhost
# DocumentRoot /var/www/html/tibit-ponmgr
# SSLProxyEngine on
# SSLProxyCheckPeerName off
# ProxyPass /api https://127.0.0.1:8013/
# <Directory /var/www/html/tibit-ponmgr>
#  Options -Indexes
#  Require all granted
#  AllowOverride all
#  RewriteEngine on
#  RewriteCond ${REQUEST_FILENAME} -f
#  RewriteRule \.json$ - [NS,F]
# </Directory>
# ErrorDocument 404 /index.html
# SSLEngine On
# SSLProtocol all -SSLv3 -TLSv1 -TLSv1.1
# SSLCipherSuite HIGH:!aNULL:!MD5
# SSLCertificateFile [path to cert here]
# SSLCertificateKeyFile [path to key here]
# <FilesMatch '\.(cgi|shtml|phtml|php)$'>
#         SSLOptions +StdEnvVars
#  </FilesMatch>
#  Directory /usr/lib/cgi-bin>

#         SSLOptions +StdEnvVars
#  </Directory>
#  BrowserMatch 'MSIE [2-6]' nokeepalive ssl-unclean-shutdown
downgrade-1.0 force-response-1.0
#</VirtualHost>
```

On lines 20 and 21 above, there are the placeholders '[path to cert here]' and '[path to key here]'. Replace these placeholders, including removing the brackets, with the path to a valid certificate and private key, respectively

The HTTPS configuration defined above will configure the standard HTTP port 80 to redirect all traffic to HTTPS port 443. Optionally, all traffic on the standard HTTP port 80 can be disabled. While both configurations are secure, disabling the standard HTTP port 80 results in traffic only being accepted on HTTPS port 443. To do so, edit the file; /etc/apache2/ports.conf

Note: This requires configuration of a non MCMS PON Manager configuration file. Any changes made to this file will not be undone by installation or uninstallation of MCMS PON Manager. Changes made to this file may affect other applications hosted on the Apache server, if any.

3. Restart Apache

```
sudo service apache2 reload
```

Validation Steps after Installation:

In a browser, navigate to “http://*IPofServerGoesHere*”

- Ex. “http://10.1.10.255”

Note: If an untrusted/self-signed certificate is used with HTTPS, you may have to navigate in your browser to the API directly to accept the API certificate.

Installation Notes:

- If you run into issues after updating the configuration files and restarting apache, try clearing your browser’s cache.
- It is recommended that 127.0.0.1 be used for the MongoDB server if you have it running locally
- Web pages are served from the default port 80. If using HTTPS, web pages are served from port 443.
- Location of install: /opt/tibit/ponmgr, /etc/tibit/ponmgr, /var/www/html (symbolic links)
- Location of error log: /var/log/apache2/error.log

Uninstall:

To Remove (keep config files):

```
sudo apt-get remove tibit-ponmgr
```

This will remove the PON Manager, but will leave configuration files behind

To Remove (remove config files):

```
sudo apt-get remove tibit-ponmgr –purge
```

This will remove the PON Manager, and will also remove configuration files

Initialize Mongo Database

Prior to installing the PON Controller, it is required to initialize the database through the PON Manager Web Interface. This will preload firmware, images and default device configurations. This section describes how to create the initial login account in PON Manager and initialize the mongo Database. The initial account will have administrative privileges and supports the ability to create new accounts for access into the PON Manager.

- Connect to PON Manager using any of the IP addresses configured on the ubuntu system
- Create an Account. Ensure the password meets the Password Requirements listed

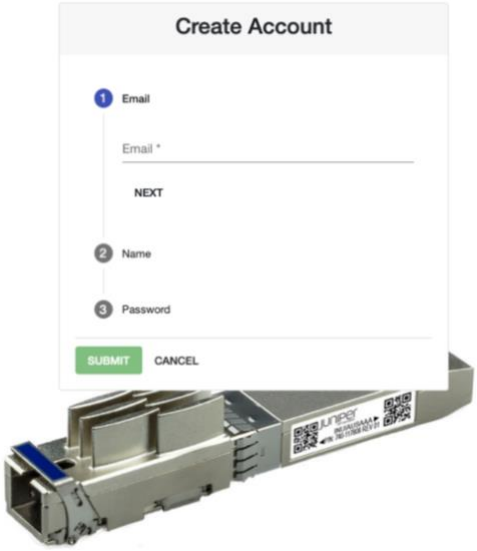

Figure 2: Create Initial Account

- Initialize the database:

  Click **Global Config** → **Databases** → Select database **Default** → **Start**

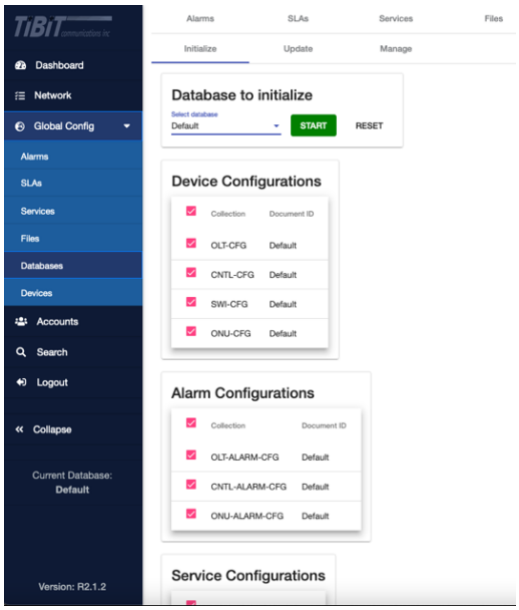

Figure 3: Initialize Database

PON Controller Installation

This section describes how to install, configure, and run the MCMS PON Controller software on a system running Ubuntu 18.04.

> **Note**: Before proceeding, ensure that the MongoDB is active and the database is initialized as described in the previous section, PON Manager Installation.

Package Contents

The MCMS PON Controller software is provided as a .zip file. The content of the package is described in the table below.

File/Directory	Description
tibit-poncntl_R2.0.4_amd64.deb	Debian Package based MCMS PON Controller application
INSTALL.txt	Quick startup guide for installing and configuring PON Controller debian package
version.txt	Version information

Installation Methods

This section describes the steps to install and uninstall the MCMS PON Controller software using the provided installation scripts.

Debian Package

The tibit-poncntl_R2.0.4_amd64.deb package handles the complete installation of PON Controller software.

Prerequisites:

- Ensure that you have an active internet connection.
- Ensure that you have your MongoDB Server running that will manage user authentication and serve as your default database.
- Ensure you have the latest ubuntu updates

Installation steps:

1. From a Linux shell, unpack the .zip file and change to the new unpacked directory.

```
unzip UnifiedPON-R2.0.4-MicroClimate-PonController.zip
cd UnifiedPON-R2.0.4-MicroClimate-PonController
```

Note: If unzip is not already installed on the ubuntu system, run the following command to install:

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

2. Use apt-get to install the package.

```
sudo apt-get install ./tibit-poncntl_R2.0.4_amd64.deb
```

Configuration steps:

Ensure that you have root/sudo access to modify files.

1. Navigate to the directory: '/etc/tibit/poncntl/'

- Within this directory, there is one PON Controller configuration file that will need to be edited with a text editor of your choice.
- Copy the root certificate (rootCA.pem file from MongDB Installation step 2) to the system running Pon Controller into the /etc/ssl directory and update Pon Controller configuration to match the settings from MongoDB Installation section:

```
$ sudo vi /etc/tibit/poncntl/PonCntlInit.json
{
    "CNTL": {
        "Auth": false,
        "CFG Version": "R2.2.0",
        "DHCPv4": true,
        "UMT interface": "tibitvirt"
    },
    "DEBUG": {},
    "JSON": {
        "databaseDir": "/opt/tibit/poncntl/database/",
        "defaultDir": "/opt/tibit/poncntl/database/"
    },
    "Local Copy": {
        "CNTL-STATE": false,
        "OLT-STATE": false,
        "ONU-STATE": false
    },
    "Logging": {
        "Directory": "/var/log/tibit",
        "FileCount": 3,
        "FileSize": 1024000
    },
    "MongoDB": {
        "auth_db": "tibit_users",
        "auth_enable": true,
        "ca_cert_path": "/etc/tibit/rootCA.pem",
        "host": "10.0.0.1",
        "name": "tibit_pon_controller",
        "password": "pdmPass",  << Use password from MongoDB Installation (if changed)
        "port": "27017",
        "tls_enable": true,
        "username": "pdmPonController",
        "dns_srv": false,
```

```
    },
    "databaseType": "MongoDB",
    "interface": "eno1.4090"
}
```

Restart Pon Controller process (tibit-poncntl):

```
$ sudo systemctl restart tibit-poncntl.service
```

	Field	Values	Description
	interface	“interface-string”	Selects the interface for the PON Controller to search for OLTs. Example: “enp0s8.4090”
	databaseType	“MongoDB”	Selects the type of Database to be used
CNTL	Auth	true or false	Used to enable or disable the Controllers Authentication Engine
	CFG Version	“R2.0.4”	The PON Controller version for the format of this file
	DHCPv4	true or false	Used to enable or disable the Controller’s DHCPv4 Engine
	DHCPv6	true or false	Used to enable or disable the Controller’s DHCPv6 Engine
	UMT Interface	“string”	Prefix to be used for the virtual interface created between the Authentication engine and Tibit Authenticator. For example, “tibitvirt” will result in the following virtual interface pair (“tibitvirteap”, “tibitvirtumt”)
DEBUG			Not used at this time
Local Copy			In MongoDB mode, a local copy of the state file is written to the JSON database. This allows another process to get status information without going to the Mongo database
	CNTL-STATE	true or false	Write the Controller State file to the JSON database as well as MongoDB
	OLT-STATE	true or false	Write the OLT State files to the JSON database as well as MongoDB
	ONU-STATE	true or false	Write the ONU State files to the JSON database as well as MongoDB
Logging			
	Directory	“directory-path”	Directory path where log files are to be stored
	FileCount	integer	The maximum number of log files to store for each module, before overwriting the oldest. (Modules include the PON Controller, TAPI, UMT Relay and the Authentication and DHCP Engines)
	FileSize	integer	The maximum size in bytes of each individual log file
MongoDB			Fields for MongoDB mode
	auth_db	“string”	Name of the MongoDB database that will be used for user authentication and authorization.

	Field	Values	Description
	auth_enable	true or false	Set to true if MongoDB access control is enabled. Set to false if MongoDB access control is not enabled.
	ca_cert_path	"/directory/filename"	Directory path to the file (in PEM format) containing the Root CA certificate that was used to sign the server certificate.
	host	"ip-address"	IP address used to access the Mongo database. Example:"127.0.0.1". If dns_srv is set to true, then this field must contain the FQDN used to lookup DNS records for the Mongo database.
	name	"database-name"	Name of the database.
	port	"network-port"	Network port number to access the Mongo database. Example:"27017". If dns_srv is set to true, then this field is ignored.
	tls_enable	true or false	Set to true to enable TLS when connecting to MongoDB. Set to false to disable TLS when connecting to MongoDB.
	username	"string"	The username to use to connect to the MongoDB database.
	password	"string"	The password to use to connect to the MongoDB database.

Validation Steps after Installation:

1. Verify the Pon Controller process is running. From a Linux shell, enter:
   ```
   systemctl status tibit-poncntl
   ```

2. View the Pon Controller log. From a Linux shell, enter:
   ```
   journalctl -f -u tibit-poncntl
   ```

Netconf Installation

This section describes how to install, configure, and run the MCMS Netconf Server software on a system running Ubuntu 18.04.

Requirements and Dependencies

Supported Operating Systems

Operating System	Version

Ubuntu, 64-bit	18.04

Software and Firmware Dependencies

Package	Version
MCMS PON Controller	R2.0.4 or greater
OLT Firmware	R2.1.2 or greater

MongoDB Dependencies

Database	Version
MongoDB	4.0.0 or greater

Required APT Packages

The following Ubuntu 18.04 apt packages are automatically installed by the Netconf .deb package.

Package	Version
libc6	2.27
libcurl3-gnutls	7.16.2
libgcc1	1:3.0
libpcre3	8.39
libpython3.6	3.6.5
libssl1.1	1.1.1
libstdc++6	5.2
zlib1g	1:1.1.4

Package Contents

The MCMS Netconf Server software is provided as a compressed .zip file. The content of the package is described in the table below.

File/Directory	Description
docker/	Directory containing scripts for building and running the MCMS Netconf Server in a docker container.
examples/	Directory containing example scripts for configuring and managing the OLT PON network using NETCONF/YANG.

File/Directory	Description
examples/bbf	Directory containing examples for using Broadband Forum TR-383 and TR385 YANG models for configuring OLT and ONU devices.
examples/nacm	Directory containing examples for configuring the Network Configuration Access Control Model (NACM) for the MCMS Netconf Server.
examples/sdn	Directory containing examples for using the MCMS Netconf Server with an SDN controller supporting a RESTCONF interface.
INSTALL.txt	Instructions for installing and quick start guide for running the MCMS Netconf Server.
LICENSE.txt	Text file that lists the licensing information for third party software used by the MCMS Netconf Server.
tibit-netconf.deb	Debian installer package containing the MCMS Netconf Server application binaries, default configuration, and supporting files.
yang/	Directory containing the Tibit YANG modules and other YANG modules supported by the server.

Installation

This section describes the steps to install MCMS Netconf Server software using the Debian (.deb) package. The Linux ‘apt’ utility is used to manage the .deb package, which contains the server binaries, configuration, and supporting files. The table below describes where the files are installed on the target system.

Directory	Description
/etc/tibit/netconf/	Configuration files for the MCMS Netconf Server.
/opt/tibit/netconf/	MCMS Netconf Server binaries, examples, YANG modules, and other supporting files.
/var/log/tibit/	Log files generated by the Netconf server.

Prerequisites

- Ubuntu Linux 18.04 should be running with the latest updates.
- An active internet connection is required to install Ubuntu Linux apt packages.
- MongoDB must be configured with a Replica Set as described in Section MongoDB Installation.
- Root level access is required to install the software.

Installation Steps

1. From a Linux shell, unpack the .zip file and change to the new unpacked directory.

```
unzip UnifiedPON-R2.1.1-MicroClimate-Netconf.zip
cd UnifiedPON-R2.1.1-MicroClimate-Netconf
```

Note: If unzip is not already installed on the ubuntu system, run the following command to install:

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

2. Use 'apt-get' to install the Netconf .deb package.

```
sudo apt-get install ./tibit-netconf_R2.1.1_amd64.deb
```

3. Configure the MCMS Netconf Server's MongoDB connection information. Edit the file '/etc/tibit/netconf/NetconfInit.json' and modify the IP address, port number, and database name as required.

```
cat /etc/tibit/netconf/NetconfInit.json

    {
        "Logging": {
            "Filename" : "/var/log/tibit/netconf.log",
            "FileCount" : 3,
            "FileSize" : 1024000,
            "Netconf" : {
                "Console" : "INFO",
                "File" : "INFO",
                "Syslog" : "INFO"
            }
        },
        "MongoDB": {
            "auth_db": "tibit_users",
            "auth_enable": true,
            "ca_cert_path": "/etc/ssl/rootCA.pem",
            "host": "mongodb0.domain.net",
            "name": "tibit_pon_controller",
            "netconf_db": "netconf_db",
            "password": "pdmPass",  << Use password from MongoDB Installation
            "port": "27017",
            "tls_enable": true,
            "username": "pdmNetconf",
            "dns_srv": false,
            "db_uri": "",
            "replica_set_enable": false,
            "replica_set_name": "rs0",
            "replica_set_hosts": []
        }
```

```
        "NETCONF": {
            "System Name": ""
        },
        "Netopeer2": {
            "Log Level": 1,
            "Sysrepo Timeout": 55
        }
    }
```

4. Restart the MCMS Netconf Server using Linux systemd scripts.

```
sudo systemctl restart tibit-netconf
```

Validation Steps

From a shell, run the "/opt/tibit/netconf/bin/netconf-users' command to enable users access to Netconf:

```
$ sudo /opt/tibit/netconf/bin/netconf-users -a -u <username>
```

Next, run the '/opt/tibit/netconf/examples/nc_get_tibitnc_version.sh' script and verify the MCMS Netconf Server version is reported as in the output below.

The script prompts for a password as shown below. When prompted, enter the password for the current user (that was just added in the previous step).

```
$ cd /opt/tibit/netconf/examples
$ ./nc_get_tibitnc_version.sh
Interactive SSH Authentication
Type your password:
Password:
DATA
<data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <netconf-state xmlns="urn:com:tibitcom:ns:yang:netconf">
    <version>
      <build-date>2021-07-01T18:56:21-07:00</build-date>
      <build-sha>e073079</build-sha>
      <version>R2.1.1</version>
    </version>
    <database>
      <ip-address>127.0.0.1</ip-address>
      <port>27017</port>
      <name>tibit_pon_controller</name>
      <status>online</status>
    </database>
  </netconf-state>
</data>
```

Uninstall

The uninstall process deletes the MCMS Netconf Server binaries and configuration files. This script completely removes the /etc/tibit/netconf and /opt/tibit/netconf directories from the file system.

Uninstall Steps

1. Use 'apt-get' to uninstall the Tibit Netconf service.

   Note: This will completely remove the /etc/tibit/netconf and /opt/tibit/netconf directories.

   ```
   sudo apt-get purge tibit-netconf
   ```

2. If the following warning occurs during 'apt-get purge', use 'rm -rf /opt/tibit/netconf' to completely remove the /opt/tibit/netconf directory.

   Note: dpkg: warning: while removing tibit-netconf, directory '/opt/tibit/netconf' not empty so not removed

   ```
   sudo rm -rf /opt/tibit/netconf
   ```