Anyfi Networks | Carrier Wi-Fi System

CONTROLLER

REFERENCE GUIDE

Overview
Installation
Basic Configuration
Radio Policies
Link Level Accounting
System Monitoring

Anyfi Networks

Västergatan 31 B
21121 Malmö
Sweden
info@anyfinetworks.com

COPYRIGHT

NOTICES

RELEASE DATE: 27th of April 2015

DOCUMENT REVISION: R1F v05

RELEASED WITH: CARRIER WI-FI SYSTEM R1F

Contents

Preface

This document details how to install, configure and integrate the Controller component of our Carrier Wi-Fi System.

Intended Audience

This document is intended for system and network administrators. Readers should have specific knowledge in the following areas:

- Networking and data communications
- Internet protocol (IP)

Readers lacking experience with the Vyatta Network OS are encouraged to study its online documentation.

Readers lacking a basic understanding of Software-Defined Wireless Networking (SDWN) concepts are encouraged to study the materials available at www.anyfinetworks.com/resources.

Document Conventions

This guide contains advisory paragraphs and uses the below typographic conventions.

Advisory Paragraphs

This guide uses the following advisory paragraphs:

Warnings alert you to situations that may pose a threat to your system or subscriber's security, as in the following example:

***WARNING** Bridging unauthenticated Wi-Fi traffic to a network interface may pose a security threat to the associated network.*

Cautions alert you to situations that might affect service, as in the following example:

***CAUTION** Restarting a running system will interrupt service.*

Notes provide important information about the structure or functioning of the system, as

in the following example:

NOTE *The Controller is a controller in the Software-Defined Networking (SDN) sense of the word, not in the typical corporate WLAN sense.*

Typographic Conventions

This document uses the following typographic conventions:

Monospace	Examples, command-line output, and representations of configuration nodes. Also commands, keywords, and file names, when mentioned inline.
bold Monospace	Your input: something you type at a command line.
italics	An argument or variable where you supply a value. Also concepts and principles when mentioned inline.
<key>	A key on your keyboard, such as <Enter>. Combinations of keys are joined by plus signs ("+"), as in <Ctrl>+c.
[*arg1* | *arg2*]	Enumerated options for completing a syntax. An example is [enable | disable].
num1–numN	A inclusive range of numbers. An example is 1–65535, which means 1 through 65535, inclusive.
arg1..argN	A range of enumerated values. An example is eth0..eth3, which means eth0, eth1, eth2, or eth3.
arg[*arg…*] *arg*[,*arg…*]	A value that can optionally represent a list of elements (a space-separated list in the first case and a comma-separated list in the second case)

Chapter 1: Functionality Overview

In the Software-Defined Wireless Networking (SDWN) architecture the control plane is centralized in a Controller, while the data plane remains distributed. The role of the Controller is to compute and distribute the forwarding state of all data plane elements. In our system that means determining which client devices should be able to connect to which virtual Wi-Fi networks through which access points.

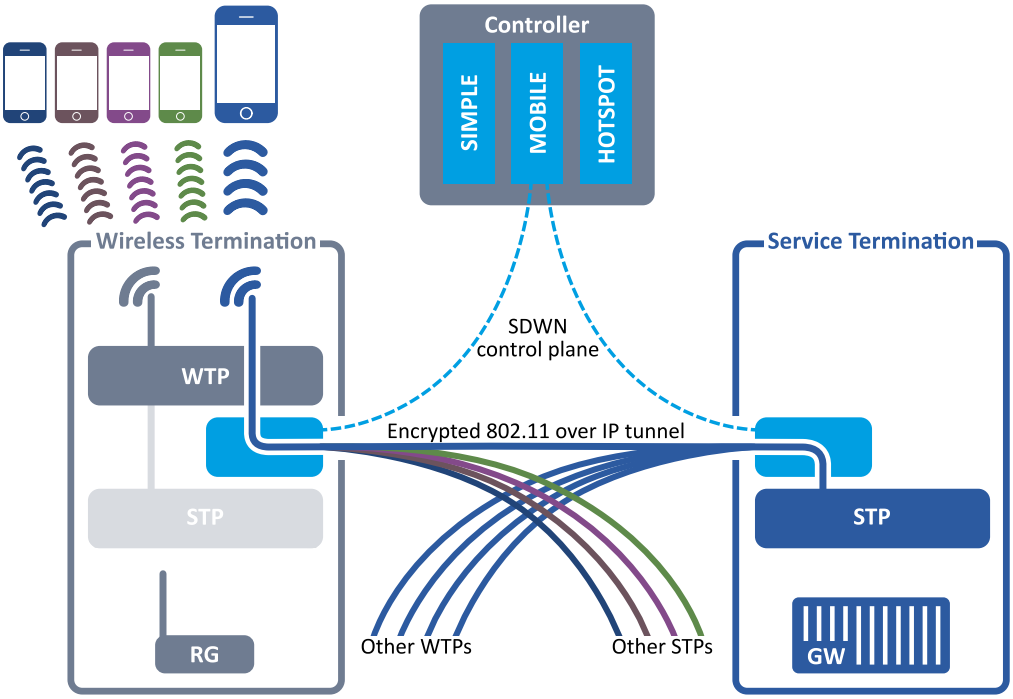

Figure 1: The Controller is at the center of a Software-Defined Wireless Networking (SDWN) architecture.

NOTE *The Controller is a controller in the Software-Defined Networking (SDN) sense of the word, not in the typical corporate WLAN sense. For example the Controller is not involved in IEEE 802.1X authentication and does not have access to end-user credentials or encryption keys.*

Concepts and Principles

The Controller communicates with all data plane elements (access points, residential gateways, Gateways and Optimizers) on a lightweight UDP/IP based SDWN control plane protocol, and acts as an introducer between these elements allowing them to establish direct SDWN data plane tunnels.

NOTE *Firmware updates, channel selection and other device management functionality is out of scope for the SDWN control plane. The Controller can only instruct access points on how to configure unused BSSIDs ("extra SSIDs") and set up the associated Wi-Fi over IP tunnels.*

Anyfi.net Software

The SDWN architecture is designed to work with existing Wi-Fi assets such as residential gateways and public access points. Before such equipment can be integrated with the Controller it must however incorporate our SDWN forwarding data plane implementation for IEEE 802.11. We refer to this embedded software as *Anyfi.net software* and call equipment incorporating it *Anyfi.net enabled*.

NOTE *Anyfi.net software can easily be installed on any Wi-Fi router that runs OpenWrt. Please see http://anyfi.net/openwrt/INSTALL for step-by-step instructions.*

Clients

We refer to mobile client devices as *clients*. The Controller CLI allows for the definition of a `client-group`, essentially a class of *clients* that should be treated in a similar fashion by the Controller and its SDWN Apps.

Radios

In the Software-Defined Wireless Networking (SDWN) architecture all access point Wi-Fi radios are treated as separate entities, referred to as *radios*. The Controller CLI allows for the definition of a `radio-group`, essentially a class of *radios* that should be treated in a similar fashion by the Controller and its SDWN Apps.

Services

Our architecture makes a clear separation between radio access and service definition, by which we essentially mean user experience: the SSID, link-level security and any web authentication/authorization as all aspects of service definition.

You can think of a *service* as a virtual Wi-Fi network, completely separated from other logical networks, potentially with its own authentication, accounting and billing infrastructure. In very technical terms you can also think of it as an IEEE 802.11 Extended Service Set (ESS).

The Controller CLI allows for the definition of a `service-group`, essentially a class of *services* that should be treated in a similar fashion by the Controller and its SDWN Apps.

NOTE *We are working here on a higher level of abstraction than most are used to. For example there can be millions of services under one Controller, each with its own Wireless LAN Controller (WLC), RADIUS AAA server, etc. The Controller's role is managing mobility between these networks.*

SDWN Apps

In Software-Defined Networks (SDN) so-called control programs ultimately make the control plane decisions. The Controller implements a number of such control programs, referred to as *SDWN Apps* or simply *apps*, and the CLI allows the operator to instantiate and apply them to groups of *clients*, *radios* and *services*.

Table 1: *SDWN Apps* built into the Controller.

Name	End-User Experience
`simple`	Seamless and secure remote access to favorite Wi-Fi networks.
`hotspot`	The more traditional hotspot or homespot user experience.

Radio Policies

By defining a `client-group`, a `radio-group` and a `service-group`, and applying an *SDWN App* to them the operator can control which *client* can connect to which *service* through which *radio*. *Radio policies* allow the operator to also define quality requirements on the radio link that must be met before a *client* can connect, making it possible not only to control the user experience but also to ensure the quality of that experience.

Table 2: Radio policy parameters configurable per *client*, *radio* and *service*.

Parameter	End-User Experience
`min-signal-level`	Minimum signal level in dBm
`min-download-capacity`	Minimum attainable downlink bandwidth
`min-uplink-capacity`	Minimum attainable uplink bandwidth
`min-dwell-time`	Time during which these limits must be met
`kick-out`	Force disconnection of thresholds are no longer met

Chapter 2: Installation

We use the Vyatta Network OS as the base operating system for the Controller. This ensures access to advanced IP networking functionality on a secure and trusted platform, and also facilitates installation in many different environments.

This software has been verified on the following versions of the Vyatta Network OS:

- Vyatta Core 6.5
- Vyatta Core 6.6
- Brocade Vyatta 5400 vRouter

The above operating systems can be run on all major hypervisors as well as bare metal x86 hardware. We provide software for both 32-bit and 64-bit architectures, but a 64-bit OS is recommended.

Installing as a Virtual Appliance

A Vyatta Core 6.6R1 64-bit virtual appliance is available for download at http://www.anyfinetworks.com/download/vyatta-anyfi-r1f.ova. Import the virtual appliance into your hypervisor of choice and follow the instructions in the next section to install the Controller software.

We also make example SDWN cores with an integrated Controller available at www.anyfinetworks.com/download.

Installing as a Vyatta Package

It can sometimes be beneficial to install the Controller software onto a running Vyatta system. One example is public clouds like Amazon AWS, where AMIs for the supported base operating systems are readily available. Another example is when the operator wishes to use physical hardware for the installation. Vyatta Network OS can then be installed on the hardware, followed by Controller software.

First configure the Vyatta system to use Anyfi Networks' software repository.

Enter configuration mode	`vyatta@vyatta:~$ configure` `[edit]`
Add Anyfi Networks' package repository	`vyatta@vyatta# edit system package repository anyfi` `[edit system package repository anyfi]`

	vyatta@vyatta# **set url http://packages.anyfinetworks.com/vyatta** [edit system package repository anyfi] vyatta@vyatta# **set components "main contrib non-free"** [edit system package repository anyfi] vyatta@vyatta# **set distribution r1f** [edit system package repository anyfi] vyatta@vyatta# **top** [edit]

Review changes	vyatta@vyatta# **show system package repository**
	+repository anyfi {
	+ components "main contrib non-free"
	+ distribution r1f
	+ url http://packages.anyfinetworks.com/vyatta
	+}
	 repository community {
	 components main
	 distribution stable
	 password ""
	 url http://packages.vyatta.com/vyatta
	 username ""
	 }
	[edit]

Commit, save and exit configuration mode	vyatta@vyatta# **commit** vyatta@vyatta# **save** vyatta@vyatta# **exit**
Download Anyfi Networks' software signing key	vyatta@vyatta$ **wget ** **http://packages.anyfinetworks.com/vyatta/pubkey.gpg -O anyfi.gpg**
Verify key integrity	vyatta@vyatta$ **sha1sum anyfi.gpg** b5a3a3233e3348ef555ba4fae11941f2339bbb88 anyfi.gpg
Install key as trusted	vyatta@vyatta$ **sudo apt-key add anyfi.gpg**
Update the software package database	vyatta@vyatta$ **sudo apt-get update**

Once the repository has been added to the system installation of the Controller software is trivial.

Install the Controller software	vyatta@vyatta:~$ **sudo apt-get install -y ** **vyatta-anyfi-controller anyfi-controller**

This will install two packages: `vyatta-anyfi-controller` containing the Vyatta CLI integration for the Controller and `anyfi-controller` containing the Controller software itself.

***NOTE** The Controller software is freely available as part of the Community Edition of our Carrier Wi-Fi System. Community Edition is unsupported and restricted to a maximum of 100 radios and services, but can be used for both commercial and non-*

commercial purposes. Contact sales@anyfinetworks.com regarding other licensing options.

Upgrading from a Previous Release

To upgrade from a previous release you only need to change distribution within the repository.

Enter configuration mode	`vyatta@vyatta:~$ configure` `[edit]`
Change distribution	`vyatta@vyatta# set system package repository anyfi distribution r1f` `[edit]`

Review changes	`vyatta@vyatta# show system package repository anyfi`
	`  components "main contrib non-free"`
	`>distribution r1f`
	`  password ""`
	`  url http://packages.anyfinetworks.com/vyatta`
	`  username ""`
	`  [edit]`

Commit, save and exit configuration mode	`vyatta@vyatta# commit`
	`vyatta@vyatta# save`
	`vyatta@vyatta# exit`

Then follow the steps below to upgrade the Controller software to the current release.

Upgrading to a Newer Version

The Controller software can be upgraded with the commands below.

Update the index of configured repositories	`vyatta@vyatta:~$ sudo apt-get update`
Upgrade the Controller software	`vyatta@vyatta:~$ sudo apt-get upgrade`

Restart the Controller	`vyatta@vyatta:~$ restart anyfi controller`
	`Stopping anyfi controller: anyfi-controller.`
	`Starting anyfi controller: anyfi-controller.`

This will install new versions of the packages containing the Controller software, if such are available in Anyfi Networks' package repository. Note that the upgraded software will remain compatible with the interfaces described in this guide.

***CAUTION** Restarting the Controller will briefly disrupt service for remotely connected SDWN clients. Locally connected primary users (e.g. owners of residential gateways) will remain unaffected.*

Installing a License

Under the Community Edition license the Controller is limited to a maximum of 100 *radios* and *services*. Use the commands below to install a commercial license.

Enter configuration mode

```
vyatta@vyatta:~$ configure
[edit]
```

Load a license key

```
vyatta@vyatta# set service anyfi controller license key \
    X2z2NuHLDnf3Axs9fIFmRLHzu0MIKGT7Jj51j3kpUHuX6kzmAOkvQfzrOpbn9UHqTA
    PuZhTdszqatRspOqTdLBAA3C5QR0POgeXFjXo784pNV+T457gJHPI45f+eoN5sfAB5
    /WJV6/T78Up2uAt0RkU9Cbt859yiqgF21/PdU/o=
[edit]
```

Commit the change

```
vyatta@vyatta# commit

License OK.

  Serial:         AA1
  Licensee:       Dummy Inc.
  Max # radios:   1
  Max # services: 1
  Expiry date:    2020-01-01
```

Commit, save and exit configuration mode

```
vyatta@vyatta# commit
vyatta@vyatta# save
vyatta@vyatta# exit
```

NOTE *Upgrading a license will not affect the operation of the Controller or cause service disruption for connected clients.*

The license is now installed. Use the operational commands below to verify that new limits have been applied to the running Controller software.

Show license limits

```
vyatta@vyatta:~$ show anyfi controller limits
======== LIMITS ========================================================
type                                value
========================================================================
max # of radios                         1
max # of services                       1

Getting close to the limits? Contact sales@anyfinetworks.com in good
    time.
```

CAUTION *Additional radios and services will not be able to register with the Controller after license limits are reached. Contact sales@anyfinetworks.com in good time before you exceeed the limitations of your license.*

Chapter 3: Configuration

In this chapter we show how to configure the Controller for basic as well as more advanced use-cases.

Basic Networking

The Controller acts as a rendezvous point for all other elements in the SDWN architecture and therefore must have a fixed (and usually public) IP address.

You will later need to configure this IP address (or the corresponding domain name) into Gateways, access points and residential gateways, so make note of it.

Enter configuration mode	vyatta@vyatta:~$ **configure** [edit]
Configure basic networking	vyatta@vyatta# **set interfaces ethernet *eth0..ethN* address *x.x.x.x/xx*** [edit] vyatta@vyatta# **set system name-server *x.x.x.x*** [edit] vyatta@vyatta# **set system gateway-address *x.x.x.x*** [edit]
Disable IP forwarding	vyatta@vyatta# **set system ip disable-forwarding** [edit]
Commit, save and exit configuration mode	vyatta@vyatta# **commit** vyatta@vyatta# **save** vyatta@vyatta# **exit**

The Controller should now have basic IP connectivity and name resolution. You can verify this e.g. with the `ping` command.

Basic SIMPLE

The SIMPLE *app* lets fixed-line subscribers connect seamlessly to their own home Wi-Fi networks on the go, whenever close to any radio in the system.

First we define some basic client, radio and service groups.

Enter configuration mode	vyatta@vyatta:~$ **configure** [edit]

Define client, radio and service groups	vyatta@vyatta# **edit service anyfi controller** [edit service anyfi controller] vyatta@vyatta# **set client-group "all"** [edit service anyfi controller] vyatta@vyatta# **set radio-group "all"** [edit service anyfi controller] vyatta@vyatta# **set service-group "all"** [edit service anyfi controller] vyatta@vyatta# **top** [edit]

We have now defined *client*, *radio* and *service* groups containing all *clients*, *radios* and *services* in the system. Now we can create an instance of the `simple` *app* and apply it to all *clients*, *radios* and *services*. This *app* instance will then start to memorize which clients connect to each *service* locally, and will make the same networks available remotely - when the client comes close to any other radio.

Create an instance of the SIMPLE app and apply it to all clients, radios and services	vyatta@vyatta# **edit service anyfi controller app simple "basic"** [edit service anyfi controller app simple basic] vyatta@vyatta# **set clients "all"** [edit service anyfi controller app simple basic] vyatta@vyatta# **set radios "all"** [edit service anyfi controller app simple basic] vyatta@vyatta# **set services "all"** [edit service anyfi controller app simple basic] vyatta@vyatta# **top** [edit]

Finally we are ready to apply our basic SIMPLE configuration.

Review changes	vyatta@vyatta# **show service anyfi controller**
	+ app {
	+ simple basic {
	+ clients all
	+ radios all
	+ services all
	+ }
	+ }
	+ client-group all {
	+ }
	+ radio-group all {
	+ }
	+ service-group all {
	+ }
	[edit]

Commit, save and exit configuration mode	vyatta@vyatta# **commit** vyatta@vyatta# **save** vyatta@vyatta# **exit**

The Controller is now running the SIMPLE *app* and is ready for use. Configure some *Anyfi.net enabled* networking equipment to use your Controller. Verify that the equipment has correctly registered using the operational commands below.

Show connected services

```
vyatta@vyatta:~$ show anyfi controller services
======== SERVICES ================================================
uuid                                  ssid              auth  groups
==================================================================
c2378125-ed09-db7d-7bb5-0d75e33b3a24  SweetHomeWiFi     psk   all
```

Show connected radios

```
vyatta@vyatta:~$ show anyfi controller radios
======== RADIOS ==================================================
hwaddr                             ip  groups
==================================================================
00:25:86:d9:61:b0   72.81.9.230:51805  all
00:25:86:f8:71:d8  119.2.19.221:34114  all
```

As you can see there is one *service* and two *radios* connected to the Controller. All *clients* that connect to the local Wi-Fi network of the *service* (SSID `SweetHomeWiFi`) will also be able to connect automatically when they come within range of any of the two *radios*.

Basic HOTSPOT

The HOTSPOT *app* provides a more traditional hotspot or homespot user experience by essentially presenting all *services* to all *clients* through all *radios*. This means that we have to restrict the *services* we let it operate on, or access points will quickly run out of BSSIDs ("extra SSIDs").

We also need to implement the service that the HOTSPOT *app* will distribute. This can be done by e.g. installing a Gateway in the network and configuring it to connect to the Controller. We here assume that a Gateway with IP address `54.92.114.81` has been configured to terminate a virtual Wi-Fi network (an open hotspot with SSID "`Hotspot 1`") that we whish to distribute through all *radios*.

We can now define *client*, *radio* and *service* groups suitable for the HOTSPOT *app*.

Enter configuration mode

```
vyatta@vyatta:~$ configure
[edit]
```

Define client, radio and service groups

```
vyatta@vyatta# edit service anyfi controller
[edit service anyfi controller]
vyatta@vyatta# set client-group "all"
[edit service anyfi controller]
vyatta@vyatta# set radio-group "all"
[edit service anyfi controller]
```

```
vyatta@vyatta# set service-group "hotspot-1"
[edit service anyfi controller]
vyatta@vyatta# set service-group "hotspot-1" ip-filter 54.92.114.81
[edit service anyfi controller]
vyatta@vyatta# top
[edit]
```

We have now defined *client* and *radio* groups containing all *clients* and *radios* in the system, as well as a *service* group containing just one *service*: the example hotspot *service*.

Now we can create an instance of the HOTSPOT *app* and apply it to these groups. This *app* instance will immediately start distributing the example hotspot *service* to all *clients* through all *radios* in the system.

Create an instance of the HOTSPOT app

```
vyatta@vyatta# edit service anyfi controller app hotspot "basic"
[edit service anyfi controller app hotspot basic]
vyatta@vyatta# set clients "all"
[edit service anyfi controller app hotspot basic]
vyatta@vyatta# set radios "all"
[edit service anyfi controller app hotspot basic]
vyatta@vyatta# set services "hotspot-1"
[edit service anyfi controller app hotspot basic]
vyatta@vyatta# top
[edit]
```

Finally we are ready to apply our basic HOTSPOT configuration.

Review changes

```
vyatta@vyatta# show service anyfi controller
+ app {
+     hotspot basic {
+         clients all
+         radios all
+         services hotspot-1
+     }
+ }
+ client-group all {
+ }
+ radio-group all {
+ }
+ service-group hotspot-1 {
+     ip-filter 54.92.114.81
+ }
[edit]
```

Commit, save and exit configuration mode

```
vyatta@vyatta# commit
vyatta@vyatta# save
vyatta@vyatta# exit
```

The Controller is now running an instance of the HOTSPOT *app* and is ready for use. Configure some *Anyfi.net enabled* networking equipment to use your Controller. Verify that the equipment has correctly registered using the operational commands below.

Show connected services

```
vyatta@vyatta:~$ show anyfi controller services
======== CONNECTED SERVICES ======================================
ssid                            uuid                 ip  groups
==================================================================
Hotspot 1       c2378125-ed09-...3a24  54.92.146.46:44970  hotspot-1
======== SERVICES ================================================
uuid                                  ssid            auth  groups
==================================================================
c2378125-ed09-db7d-7bb5-0d75e33b3a24  Hotspot 1       open  hotspot-1
```

Show connected radios

```
vyatta@vyatta:~$ show anyfi controller radios
======== CONNECTED RADIOS ========================================
hwaddr                             ip  groups
==================================================================
00:25:86:d9:61:b0   72.81.9.230:51805  all
00:25:86:f8:71:d8  119.2.19.221:34114  all
```

As you can see the Gateway terminating the hotspot network has registered with the Controller, and there are also two access point *radios*. All *clients* that come within range of any of the two *radios* will be presented with the hotspot *service*, i.e. their devices will detect the "Hotspot 1" SSID and they will be able to connect to this *service* through the *radio*.

Securing the Control Plane

The SDWN control plane connects the Controller to all data plane elements: access points, client premises equipment, Gateways and Optimizers. The Controller however only exerts a very limited influence over connected systems, essentially deciding how to use spare radio resources to provide mobile Wi-Fi services. Neither system nor user data plane security is affected by configuring a data plane element with a Controller.

There are however reasons to protect the confidentiality and integrity of the control plane itself. For example, passive monitoring of the control plane traffic can give an attacker information about the mobility of devices across the network coverage area. To prevent such attacks the SDWN control plane is protected with Datagram Transport Layer Security (DTLS).

Even with DTLS protection it is however possible to mount a man-in-the-middle, letting the attacker inject packets on the control plane, e.g. controlling the allocation of radio resources towards unsanctioned purposes. This can however be prevented through so-called public key pinning.

RSA Key Pair Generation

To enable Controller authentication through public key pinning you must first generate an RSA key pair.

Generate an RSA key pair to be used for DTLS	vyatta@vyatta:~$ **generate anyfi controller rsa-key-pair /config/auth/ctrl-key-pair.pem** Generating 2048 bit RSA key pair to file /config/auth/ctrl-key-pair.pem... Done. [...] === CONTROLLER KEY ========== 9878d127f83b41dee54034e88c627b0ac886c44d2a209f3f4447a41a740cddcb

The *controller key* is a cryptographic hash over the public portion of the RSA key pair. It can be configured into other network elements through their command line interfaces (CLI) or the Anyfi.net TR-069 vendor extension.

Public Key Pinning

To enable authentication through public key pinning simply configure the Controller with the RSA key pair generated above.

Enter configuration mode	vyatta@vyatta:~$ **configure** [edit]
Set the RSA key pair to be used for DTLS	vyatta@vyatta# **set service anyfi controller rsa-key-pair file /config/auth/ctrl-key-pair.pem** [edit]
Commit, save and exit configuration mode	vyatta@vyatta# **commit** vyatta@vyatta# **save** vyatta@vyatta# **exit**

Other network elements will now be able to authenticate the Controller using its *controller key*. This offers protection against a rouge Controller or man-in-the-middle attack: network elements configured with a *controller key* will only take instructions from a Controller with access to the private portion of the corresponding RSA key pair.

The operational command below can be used to display the *controller key* of a running Controller.

Show the controller key	vyatta@vyatta:~$ **show anyfi controller key** 9878d127f83b41dee54034e88c627b0ac886c44d2a209f3f4447a41a740cddcb

Note that configuring an RSA key pair to be used for DTLS will not disconnect any data plane elements or cause any service disruption.

***WARNING** DTLS with public key pinning ensures that data plane elements take instructions only from a trusted Controller. The authentication is however not*

mutual, i.e. the Controller does not authenticate data plane elements. It should NOT be assumed that a data plane element connected through DTLS must have knowledge of the controller key.

Radio Policies

Radio policies allow the operator to also define quality requirements on the radio link that must be met before a *client* can connect. A separate *radio policy* can be defined for each *SDWN App* instance.

Enter configuration mode

```
vyatta@vyatta:~$ configure
[edit]
```

Select a SDWN App instance

```
vyatta@vyatta# edit service anyfi controller app hotspot "basic"
[edit service anyfi controller app hotspot basic]
```

List radio policy options

```
vyatta@vyatta# set radio-policy <tab>
Possible completions:
   kick-out
                Force disconnect if quality thresholds no longer met
   min-downlink-capacity
                Minimum attainable downlink bandwidth
   min-dwell-time
                Minimum time client must remain stationary
   min-signal-level
                Minimum received signal strength
   min-uplink-capacity
                Minimum attainable uplink bandwidth

[edit service anyfi controller app hotspot basic]
```

Define a radio policy

```
vyatta@vyatta# set radio-policy min-downlink-capacity <tab>
Possible completions:
   <xx.xx>      Bandwidth in megabits per second (Mbps)

[edit service anyfi controller app hotspot basic]
vyatta@vyatta# set radio-policy min-downlink-capacity 1.0
[edit service anyfi controller app hotspot basic]
vyatta@vyatta# set radio-policy min-uplink-capacity 0.5
[edit service anyfi controller app hotspot basic]
vyatta@vyatta# set radio-policy min-signal-level <tab>
Possible completions:
   <-95-0>      Signal strength in decibels of 1 mW (dBm)

[edit service anyfi controller app hotspot basic]
vyatta@vyatta# set radio-policy min-signal-level -75
[edit service anyfi controller app hotspot basic]
vyatta@vyatta# set radio-policy min-dwell-time <tab>
Possible completions:
```

	<0-4294967295> Dwell-time in seconds (s) [edit service anyfi controller app hotspot basic] vyatta@vyatta# **set radio-policy min-dwell-time 2** [edit service anyfi controller app hotspot basic] vyatta@vyatta# **set radio-policy kick-out** [edit service anyfi controller app hotspot basic] vyatta@vyatta# **top** [edit]
Review changes	vyatta@vyatta# **show service anyfi controller app hotspot "basic"** clients all radios all services hotspot-1 +radio-policy { + kick-out + min-downlink-capacity 1.0 + min-dwell-time 2 + min-signal-level -75 + min-uplink-capacity 0.5 +} [edit]
Commit, save and exit configuration mode	vyatta@vyatta# **commit** vyatta@vyatta# **save** vyatta@vyatta# **exit**

Radio links been *clients* and access point *radios* will now need to meet the specified requirements for a minimum of 2 seconds before the client is allowed to connect. The *Anyfi.net software* in access points will enforce this policy on the IEEE 802.11 level; *clients* whose radio links do not meet the requirements will not receive responses to select management frames. If the radio link quality or spare capacity of the access point later falls below the required threshold values *Anyfi.net software* will force the disassociation of the *client*.

***NOTE** Changes to the radio policy will take effect immediately without unintended service disruption. In large networks (with hundreds of thousands of radios) it may however take some time to propagate the data plane forwarding state updates to all affected radios.*

Alternate Radio Policies for Special Cases

Radio policies provide low-level control over user experience, but sometimes an operator may want different user experiences in different use-cases.

For example, one common problem in homespot networks is that mobile devices connect to the public network even when a more suitable local Wi-Fi network is available. This can result in a poor user experience, unexpected connectivity issues and additional load on the operator's network. To address this problem the operator can configure an alternate

radio policy for when the *client* has a *preference* for a nearby *service*, i.e. when the mobile device has previously connected to a Wi-Fi network that now happens to be within radio range.

Enter configuration mode	vyatta@vyatta:~$ **configure** [edit]
Select an SDWN App instance	vyatta@vyatta# **edit service anyfi controller app hotspot "basic"** [edit service anyfi controller app hotspot basic]
List alternate radio policy options	vyatta@vyatta# **set radio-policy when client has-preference-for nearby-service <tab>** Possible completions: kick-out Force disconnect if quality falls below thresholds min-downlink-capacity Minimum attainable downlink bandwidth min-dwell-time Minimum time that device must remain stationary min-signal-level Minimum received signal strength min-uplink-capacity Minimum attainable uplink bandwidth suppress Suppress service [edit service anyfi controller app hotspot basic]
Configure an alternate radio policy	vyatta@vyatta# **set radio-policy when client has-preference-for nearby-service suppress** [edit service anyfi controller app hotspot basic] vyatta@vyatta# **top** [edit]
Commit, save and exit configuration mode	vyatta@vyatta# **commit** vyatta@vyatta# **save** vyatta@vyatta# **exit**

Remember how the basic task of the Controller is to determine which *client* can connect to which *service* through which *radio*. For the special case above the alternate *radio policy* applies equally to all *services*: if the *client* has a preference for any *service* that happens to be nearby, then the alternate *radio policy* is applied. But for the SIMPLE *app* there is also the case where the candidate *service* itself happens to be nearby, i.e. the client may just as well connect directly through the ether instead of through an SDWN tunnel. An alternate *radio policy* can be configured for this case as well.

Enter configuration mode	vyatta@vyatta:~$ **configure** [edit]
Select a SIMPLE app instance	vyatta@vyatta# **edit service anyfi controller app simple "basic"** [edit service anyfi controller app simple basic]
List alternate radio policy options	vyatta@vyatta# **set radio-policy when service is-nearby <tab>** Possible completions: kick-out Force disconnect if quality falls below thresholds

```
                min-downlink-capacity
                            Minimum attainable downlink bandwidth
                min-dwell-time
                            Minimum time that device must remain stationary
                min-signal-level
                            Minimum received signal strength
                min-uplink-capacity
                            Minimum attainable uplink bandwidth
                suppress    Suppress service

[edit service anyfi controller app simple basic]
```

Configure an alternate radio policy

```
vyatta@vyatta# set radio-policy when service is-nearby suppress
[edit service anyfi controller app simple basic]
vyatta@vyatta# top
[edit]
```

Commit, save and exit configuration mode

```
vyatta@vyatta# commit
vyatta@vyatta# save
vyatta@vyatta# exit
```

Group Filters

In the previous section we have come across the concept of *client*, *radio* and *service* groups. In this section we will explore this concept in more detail and show how group filters can be leveraged for fine-grained control over *service* distribution and *radio policies*.

Restricting Distribution

Taking the basic HOTSPOT configuration as a starting point we now look at how we can restrict the distribution of the `hotspot-1` *service* to a subset of *radios*.

First let us remind ourselves of the basic HOTSPOT configuration.

Enter configuration mode

```
vyatta@vyatta:~$ configure
[edit]
```

Show Controller configuration

```
vyatta@vyatta# show service anyfi controller
  app {
      hotspot basic {
          clients all
          radios all
          services hotspot-1
      }
  }
  client-group all {
  }
  radio-group all {
```

```
    }
    service-group hotspot-1 {
        ip-filter 54.92.114.81
    }
[edit]
```

Assume that we are a fixed-line operator with *Anyfi.net* *enabled* residential gateways installed in consumer homes, and a network of *Anyfi.net* *enabled* public access points installed in prime locations. We may want to e.g. charge more for data flowing through the former than the latter, and when a wholesale customer does not want to pay these higher fees we may wish to limit distribution of their *service* to homespot *radios* only.

To accomplish this we define two *radio* groups: `premium` and `homespots`.

Define premium and homespot radio groups

```
vyatta@vyatta# edit service anyfi controller
[edit service anyfi controller]
vyatta@vyatta# set radio-group "homespots" ip-filter 72.12.0.0/16
[edit service anyfi controller]
vyatta@vyatta# set radio-group "premium" ip-filter !72.12.0.0/16
[edit service anyfi controller]
vyatta@vyatta# top
[edit]
```

In this example we group *radios* based on IP address; `homespots` will register from the `72.12.0.0/16` IP subnet allocated to residential subscribers. *Radios* that register with the Controller from other IP addresses are considered `premium`.

We have now configured the Controller to tag *radios* as belonging either to the `premium` or the `homespots` group. But the HOTSPOT *app* is still configured to distribute the `hotspot-1` *service* across the *radio* group `all`, which all `radios` are members of since it has no filters.

We now change the configuration so that the `hotspot-1` *service* is distributed only through homespot *radios*.

Change the HOTSPOT app configuration to limit distribution to homespot radios

```
vyatta@vyatta# edit service anyfi controller app hotspot "basic"
[edit service anyfi controller app hotspot basic]
vyatta@vyatta# delete radios "all"
[edit service anyfi controller app hotspot basic]
vyatta@vyatta# set radios "homespots"
[edit service anyfi controller app hotspot basic]
vyatta@vyatta# top
[edit]
```

We are now ready to review and commit our changes.

Review changes

```
vyatta@vyatta# show service anyfi controller
 app {
```

```
      hotspot basic {
          clients all
-         radios all
+         radios homespots
          services hotspot-1
      }
  }
  client-group all {
  }
  radio-group all {
  }
+radio-group homespots {
+    ip-filter 72.12.0.0/16
+}
+radio-group premium {
+    ip-filter !72.12.0.0/16
+}
  service-group hotspot-1 {
      ip-filter 54.92.114.81
  }
[edit]
```

Commit, save and exit configuration mode

```
vyatta@vyatta# commit
vyatta@vyatta# save
vyatta@vyatta# exit
```

The `hotspot-1` *service* will now be distributed only through homespot *radios*. In practical terms that means public access points will no longer broadcast the "`Hotspot 1`" SSID, while residential gateways will continue to do so.

NOTE *Filter changes affecting service distribution will take effect immediately without unintended service disruption. In large networks (with hundreds of thousands of radios) it may however take some time to propagate the data plane forwarding state updates to all affected radios.*

Combining Multiple Filters and Groups

Now we continue the above example by adding one more requirement: we have expanded our business to include broadband services for public venues like restaurants, cafés and shops and we wish to distribute the `hotspot-1` *service* across these *radios* as well. That means that we want to differentiate between three types of *radios*: public access points (`premium`), residential gateways (`homespots`) and public venues.

Assume that we have allocated the `72.13.0.0/16` IP subnet for our business broadband service. We use this to define an additional *radio* group.

Define a radio group for public venues

```
vyatta@vyatta# edit service anyfi controller
[edit service anyfi controller]
vyatta@vyatta# set radio-group "venues" ip-filter 72.13.0.0/16
```

```
[edit service anyfi controller]
vyatta@vyatta# top
[edit]
```

The Controller will now tag all *radios* that register from the `72.13.0.0/16` IP subnet as belonging to the `venues` group. That is a step in the right direction, but remember the `premium` *radio* group?

Show the premium radio group definition

```
vyatta@vyatta# show service anyfi controller radio-group "premium"
 radio-group premium {
     ip-filter !72.12.0.0/16
 }
[edit]
```

As you can see the Controller will also tag public venue radios as belonging to the `premium` radio group. In some cases this overlap between radio groups can be beneficial, but in this case we want to define three distinct groups (`premium`, `homespots` and `venues`), where each *radio* is only a member of one group.

We therefore adjust the definition of the `premium` *radio* group to exclude *radios* registering from the business broadband IP subnet.

Adjust the definition of the premium radio group

```
vyatta@vyatta# edit service anyfi controller radio-group "premium"
[edit service anyfi controller radio-group premium]
vyatta@vyatta# set ip-filter !72.13.0.0/16
[edit service anyfi controller radio-group premium]
vyatta@vyatta# top
[edit]
```

Show the definition of the premium radio group

```
vyatta@vyatta# show service anyfi controller radio-group "premium"
 ip-filter !72.12.0.0/16
+ip-filter !72.13.0.0/16
[edit]
```

With this configuration the Controller will only tag *radios* as belonging to the `premium` *radio* group if they register from an IP address that is not in the `72.12.0.0/16` IP subnet allocated to residential broadband customers AND not in the `72.13.0.0/16` IP subnet allocated to business broadband customers. As you can see an implicit AND is assumed between all filter rules within a group.

We are now ready to expand the distribution of the `hotspot-1` *service* to *radios* in public venues.

Add the venues radio group to the HOTSPOT app

```
vyatta@vyatta# edit service anyfi controller app hotspot "basic"
[edit service anyfi controller app hotspot basic]
vyatta@vyatta# add radios "venues"
[edit service anyfi controller app hotspot basic]
vyatta@vyatta# top
```

	[edit]

Review changes	vyatta@vyatta# **show service anyfi controller**
	app {
	hotspot basic {
	clients all
	radios homespots
	+ radios venues
	services hotspot-1
	}
	}
	client-group all {
	}
	radio-group all {
	}
	radio-group homespots {
	ip-filter 72.12.0.0/16
	}
	radio-group premium {
	ip-filter !72.12.0.0/16
	+ ip-filter !72.13.0.0/16
	}
	+radio-group venues {
	+ ip-filter 72.13.0.0/16
	+}
	service-group hotspot-1 {
	ip-filter 54.92.114.81
	}
	[edit]

Commit, save and exit configuration mode	vyatta@vyatta# **commit** vyatta@vyatta# **save** vyatta@vyatta# **exit**

The `hotspot-1` *service* will now be distributed through both homespot and venue *radios*, but not through public access points in premium locations. As you can see multiple *client*, *radio* and *service* groups can be included in one *app* instance. In practice this is often used to encode a logical OR between filter rules.

***NOTE** The Controller will assume an implicit logical AND between all filter rules within a (client, radio or service) group. Multiple groups can be included into a single app instance, effectively encoding a logical OR between filter rules.*

Combining Groups with Radio Policies

We showed above how to use group filters for fine-grained control over *service* distribution. In this section we show how to apply the same concept to *radio policy* control.

Remember that our current configuration distributes the `hotspot-1` *service* through the *radio* groups `homespots` and `venues`. Now assume that we want to define different *radio policies* for the two groups. Since each *app* instance can have only one *radio policy* we need to split the basic *app* in two: one that distributes `hotspot-1` through `homespots` and one that distributes `hotspot-1` through `venues`.

Delete the basic HOTSPOT app instance

```
vyatta@vyatta# delete service anyfi controller app hotspot "basic"
[edit]
```

Define a HOTSPOT app instance for homespots

```
vyatta@vyatta# edit service anyfi controller app hotspot "homespots-1"
[edit service anyfi controller app hotspot homespots-1]
vyatta@vyatta# set clients "all"
[edit service anyfi controller app hotspot homespots-1]
vyatta@vyatta# set radios "homespots"
[edit service anyfi controller app hotspot homespots-1]
vyatta@vyatta# set services "hotspot-1"
[edit service anyfi controller app hotspot homespots-1]
vyatta@vyatta# set radio-policy min-signal-level -80
[edit service anyfi controller app hotspot homespots-1]
vyatta@vyatta# top
[edit]
```

Define a HOTSPOT app instance for venues

```
vyatta@vyatta# edit service anyfi controller app hotspot "venues-1"
[edit service anyfi controller app hotspot venues-1]
vyatta@vyatta# set clients "all"
[edit service anyfi controller app hotspot venues-1]
vyatta@vyatta# set radios "homespots"
[edit service anyfi controller app hotspot venues-1]
vyatta@vyatta# set services "hotspot-1"
[edit service anyfi controller app hotspot venues-1]
vyatta@vyatta# set radio-policy min-signal-level -70
[edit service anyfi controller app hotspot venues-1]
vyatta@vyatta# set radio-policy kick-out
[edit service anyfi controller app hotspot venues-1]
vyatta@vyatta# top
[edit]
```

We are now ready to review and commit our changes.

Review changes

```
vyatta@vyatta# show service anyfi controller
 app {
-    hotspot basic {
-        clients all
-        radios homespots
-        radios venues
-        services hotspot-1
-    }
+    hotspot homespots-1 {
+        clients all
+        radios homespots
```

```
+        services hotspot-1
+        radio-policy {
+            min-signal-level -80
+        }
+    }
+    hotspot venues-1 {
+        clients all
+        radios venues
+        services hotspot-1
+        radio-policy {
+            kick-out
+            min-signal-level -75
+        }
+    }
 }
 client-group all {
 }
 radio-group all {
 }
 radio-group homespots {
     ip-filter 72.12.0.0/16
 }
 radio-group premium {
     ip-filter !72.12.0.0/16
     ip-filter !72.13.0.0/16
 }
 radio-group venues {
     ip-filter 72.13.0.0/16
 }
 service-group hotspot-1 {
     ip-filter 54.92.114.81
 }
[edit]
```

Commit, save and exit configuration mode

```
vyatta@vyatta# commit
vyatta@vyatta# save
vyatta@vyatta# exit
```

The Controller will now send out slightly different *radio policies* to the residential gateways installed in subscriber homes and the business-grade CPE installed in public venues. Other than that the system will continue to function as before.

NOTE *SDWN App instantiation and other configuration changes will take immediate effect without causing unintended service disruption. In large networks (with hundreds of thousands of radios) it may however take some time to propagate the data plane forwarding state updates to all affected radios.*

Other Filter Types

In previous sections we have used *radio* group `ip-filter` to control *service* distribution and *radio policy*. But there are also other types of filters.

Radios can also be grouped based on their Wi-Fi radio MAC address or Organizationally Unique Identifier (OUI).

Show radio group filter types

```
vyatta@vyatta# set service anyfi controller radio-group r <tab>
Possible completions:
   description  Radio group description
+  ip-filter    Filter on IP address or subnet
+  mac-filter   Filter on MAC address or OUI

[edit]
```

Services can be grouped based on their IP address or Universally Unique Identifier (UUID).

Show service group filter types

```
vyatta@vyatta# set service anyfi controller service-group s <tab>
Possible completions:
   description  Service group description
+  ip-filter    Filter on IP address or subnet
+  uuid-filter  Filter on Universally Unique Identifier (UUID)

[edit]
```

Service group filters can be very useful for adjusting *radio policies*. For example, a SIM authenticated *service* will be used by *clients* with an alternative 3G or 4G data connection. It may therefore be beneficial to set higher quality thresholds for such *services* (regardless of *radio*), to prevent *clients* from staying connected to a low quality Wi-Fi signal instead of falling back to 3G or 4G.

Clients can be grouped based on their MAC address or Organizationally Unique Identifier (OUI).

Show client group filter types

```
vyatta@vyatta# set service anyfi controller client-group c <tab>
Possible completions:
   description  Client group description
+  mac-filter   Filter on MAC address or OUI

[edit]
```

While it is less common to use *client* group filters to control *radio policies* or *service* distribution it is fully supported by the Controller.

Chapter 4: Integration

In a Software-Defined Wireless Networking (SDWN) architecture most of the integration requirements, e.g. authentication, accounting and billing, fall onto the data plane elements. The Controller can therefore be operated as a freestanding control plane element. There are however good reasons to integrate towards network monitoring systems and structured data analysis tools such as Splunk.

SYSLOG for Link Level Accounting

The Controller collects radio link level accounting information and makes this available on a SYSLOG interface. This information gives an overview of the functioning of the entire network, and can be processed into near real-time dashboards and graphical reports with a structured data analysis tool such as Splunk.

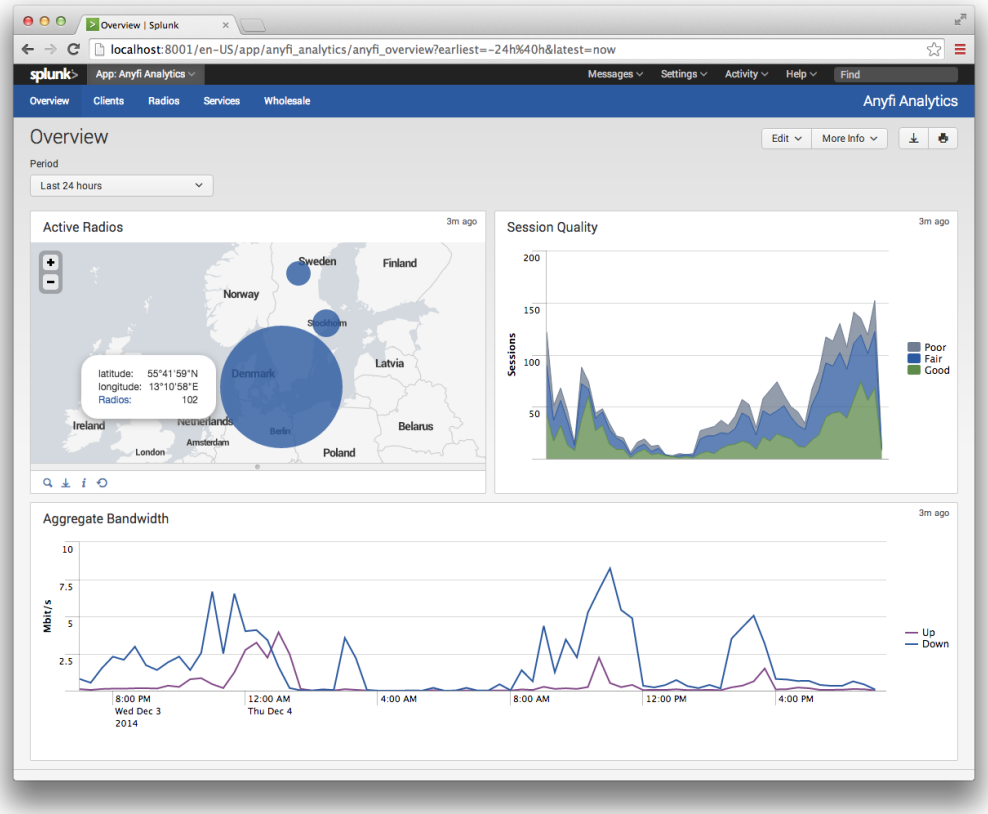

Figure 2: A structured data analysis tool such as Splunk can be used to process the SYSLOG output of the Controller into dashboards and graphical reports.

Integrating the Controller towards an external system running Splunk can be accomplished with a few simple commands.

Enter configuration mode	vyatta@vyatta:~$ **configure** [edit]
Direct the SYSLOG output of the Controller to an external machine running Splunk	vyatta@vyatta# **edit system syslog** [edit system syslog] vyatta@vyatta# **set host *x.x.x.x* facility all level info** [edit system syslog] vyatta@vyatta# **top** [edit]
Commit, save and exit configuration mode	vyatta@vyatta# **commit** vyatta@vyatta# **save** vyatta@vyatta# **exit**

SNMP for System Monitoring

The Vyatta Network OS supports system monitoring through the Simple Networking Monitoring Protocol (SNMP). Configure the system for remote monitoring with the following commands.

Enter configuration mode	vyatta@vyatta:~$ **configure** [edit]
Configure SNMP v2c read-only with traps	vyatta@vyatta# **edit service snmp** [edit service snmp] vyatta@vyatta# **set listen-address *x.x.x.x*** [edit service snmp] vyatta@vyatta# **set trap-source *x.x.x.x*** [edit service snmp] vyatta@vyatta# **set community *name* authorization ro** [edit service snmp] vyatta@vyatta# **set community *name* client *x.x.x.x*** [edit service snmp] vyatta@vyatta# **set trap-target *x.x.x.x* community *name*** [edit service snmp] vyatta@vyatta# **set location *<location>*** [edit service snmp] vyatta@vyatta# **set contact *<contact>*** [edit service snmp] vyatta@vyatta# **top** [edit]
Commit, save and exit configuration mode	vyatta@vyatta# **commit** vyatta@vyatta# **save** vyatta@vyatta# **exit**