Technical Guide

Allied Telesis™

Bridging

Feature Overview and Configuration Guide

Bridging introduction

Bridging is a data link layer (Layer 2) process that enables multiple physical LAN segments to be interconnected into a single larger network. Bridges forward and flood traffic based on MAC addresses. Bridging can also be used to connect two remote sites to the same broadcast domain.

Bridging can:

- connect two separate networks as if they were a single network.
- connect two or more Layer 2 interfaces together to form a single broadcast domain, or many broadcast domains in the case of the VLAN aware bridge 0.
- forward packets in software, based on destination address in the Layer 2 header. This is similar forwarding logic to Layer 2 switching, which forwards packets in hardware.

There are two main use-cases for bridging:

- extending a broadcast domain across two or more physically separated sites.
- applying security processing to traffic transparently in a Layer 2 network.

What topics are discussed in this guide?

The guide describes the bridging feature and operation. It begins with some basic useful commands to help you create and add interfaces to a simple bridge, change the ageing timer, and verify your bridge configuration. This is followed with some more detailed configuration examples, for example, how to bridge traffic between VLAN and Ethernet interfaces for multiple VLAN IDs.

The guide goes on to describe the more advanced bridge MAC filtering. This is a Layer 2 filter containing a collection of rules applied to a bridge. Each rule matches certain types of Layer 2 traffic, and will either discard it or allow it to continue through the bridge. Finally we provide some examples of advanced bridge MAC filtering. One example blocks or allows frames using filters based on source MAC address and Ethernet protocol type. Another example uses a fine-grained filter on IPv4/6 L3/L4 source and destination addresses. And the final example configures a PPPoE client on the bridge.

AlliedWare Plus™ OPERATING SYSTEM

Special terms

Here's a description for a few special terms used in this guide:

TERM	DESCRIPTION
Virtual Tunnel Interface (VTI)	In order to apply higher-layer functions (like multicasting, routing protocols, filtering etc.) to a VPN tunnel, it is convenient to treat the tunnel as a virtual Layer 3 interface. The virtual IP interface that is overlaid on a VPN tunnel is called a Virtual Tunnel Interface.
Bridge Entity ID	A Layer 3 interface to allow the host to be managed over the bridged network.
L2TPv3 pseudo-wire	L2TPv3 is an IETF standard for the encapsulation of multi-protocol Layer 2 communications traffic over IP networks. A pseudo-wire is an emulated circuit. A pseudo-wire can extend Layer 2 circuits via intermediate packet switched networks, including the Internet. For example, you can connect to two physically separated VLANs such as a remote office and a main office network, via an L2TPv3 Ethernet pseudo-wire. This is achieved by bridging each office VLAN to a virtual Tunnel Interface (VTI) terminating an L2TPv3 Ethernet pseudo-wire.

Products and software version that apply to this guide

This guide applies to Allied Telesis UTM Firewalls and Secure VPN Routers, running version **5.4.5** or later.

To see whether your product supports advanced security features, see the following documents:

- The product's Datasheet
- The product's Command Reference

These documents are available from the above links on our website at alliedtelesis.com.

Most features described in this document are supported from AlliedWare Plus™ software version 5.4.5 or later. The following features became available in later releases:

- From software version 5.4.8-1.1 the port-protected bridge filtering feature is supported, see "Port-protected bridge filtering" on page 24
- From software version 5.4.8 the maximum number of bridge entities that can exist within one physical device was increased from 64 to 255. This version also supports enhanced bridge MAC filtering.
- Software version 5.4.7-0.1 added support for disabling of FDB MAC address learning on bridges—see "Disabling MAC-learning on bridges" on page 12
- In software version 5.4.6-0.1 the maximum number of bridges that could be configured on UTM Firewalls and Secure VPN Routers was increased from 16 to 64.
- From software version 5.4.9-2.1 VLAN aware bridges are supported —see "VLAN aware bridge" on page 9
- From software version 5.5.0-1.4, PPPoE relay is supported—see the PPP Feature Overview and Configuration Guide for information on how to configure this feature.

Contents

Bridging operation

Bridging forwards packets in software, based on the Layer 2 header. This is similar to the forwarding logic in Layer 2 switching, which forwards packets in hardware. Switch ports cannot be bridged. Tunnels, physical Ethernet interfaces, and VLANs **can** be bridged. However, these interfaces can only be members of one bridge at one time. If a packet is bridged then it is not processed by the normal Layer 3 packet forwarding path, such as routing and firewall.

Bridging features

The bridge combines its constituent interfaces into a virtual Layer 2 switch. A range of interface types can be attached to a bridge, they include:

- Ethernet
- Tagged Ethernet
- VLAN
- Tunnel

By default, there are no limitations on the types of Ethernet traffic that a bridge will forward. Tagged or untagged traffic can be forwarded by a bridge. The software will check the validity of the Ethernet frame to be bridged, which includes checking Layer 3 protocol fields. Invalid frames will be dropped, and the ingress port (not the bridge, but the underlying port) discard counter will be incremented.

Bridges implement the same forwarding rules as a switch

- Broadcast and multicast traffic is flooded to all interfaces attached to the bridge.

  The source MAC addresses of packets ingressing each interface are stored in a forwarding table, just as with a switch, so that unicast packets will only be sent to an interface that is known to provide a path to the packet's destination MAC.
- Destination lookup failures (failure to find a packet's destination MAC in the forwarding table) will result in the packet being flooded to all but the ingress interface, just as with a switch.
- MAC addresses will age out of the MAC forwarding table if packets with that particular source MAC address have not been received on the bridge interface for a certain length of time. The length of time (ageing time) can be configured using the **ageing-time** command.

Bridges have forwarding rules for Layer 2 Control plane messages

The IEEE Std 802.1D and IEEE Std 802.1Q define a list of reserved destination MAC addresses that are used for Layer 2 Control Plane (L2CP) messaging. See IEEE reserved MAC addresses.

L2CP messages using these reserved MAC addresses are typically not allowed to cross between bridged interfaces. This prevents unnecessary looping and Ethernet broadcast storms that could otherwise arise if neighboring devices are unable to recognize them.

Many Layer 2 control plane protocols, for example Link Aggregation Control Protocol (LACP), Link Layer Discovery Protocol (LLDP), and Link Operation Administration and Management rely on this behavior. The bridge implementation in AlliedWare Plus follows these IEEE standards and therefore does not transparently forward all L2CP messages.

In AlliedWare plus, the following protocol control frames are not forwarded through the bridge:

- (MAC Control) 802.3 01-80-C2-00-00-01
- (Link Aggregation) 802.3 01-80-C2-00-00-02
- 802.1X PAE address 01-80-C2-00-00-03
- 802.1AB LLDP 01-80-C2-00-00-0E

Additionally, Layer 2 AMF messages are a *special case*, and they are also treated as Layer 2 control plane messages and so are not bridged. All other Ethernet frame types and Layer 2 data plane messages are bridged by default.

However, not all L2CP messages are blocked or prevented from crossing the IEEE 802 bridged domain by default. EPSR, and STP BPDUs are currently bridged.

Bridges can have Layer 3 interfaces

The bridge is treated as a Layer 3 interface into the Layer 2 network to which its constituent interfaces are connected. As such, the bridge can have higher-layer configuration applied to it – i.e. IPv4 and/or IPv6 addresses can be attached to the bridge, the bridge can be a PIM interface, an OSPF interface, a destination interface for static IP routes, etc.

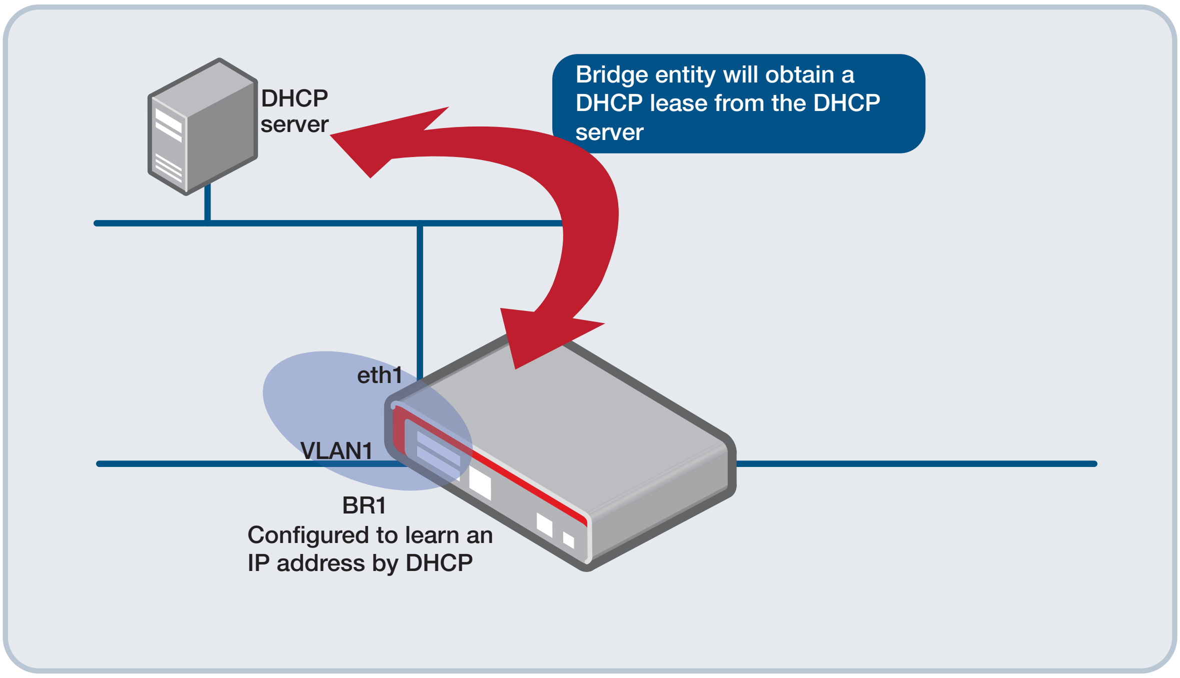
An IP address cannot be allocated to an interface that is attached to a bridge because it is operating as a Layer 2 interface.

A bridge can even be configured to learn an IP address by DHCP

If there is a DHCP server on one of the Ethernet segments attached on one of the bridge’s constituent interfaces, then the bridge can obtain a DHCP lease from that server.

For example, if a host attached to eth1of the device, in subnet A, wishes to connect to a host attached to one of the interfaces of a bridge, in subnet B, then the device will route the packets between eth2 and the bridge entity.

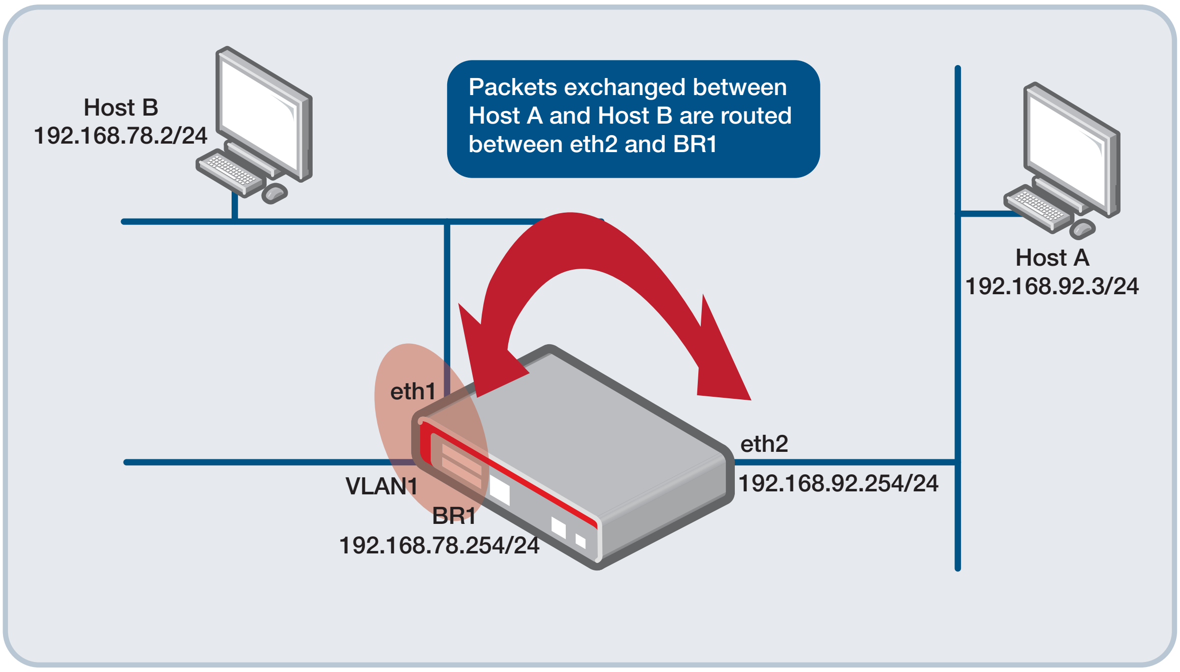
Figure 1: Layer 3 interface into Layer 2 network example


DHCP server
Bridge entity will obtain a DHCP lease from the DHCP server
eth1
VLAN1
BR1
Configured to learn an IP address by DHCP

A bridge can have multiple interfaces on the same device

Multiple separate bridges can exist within the same physical device. However, any given interface can only be attached to one bridge at a time.

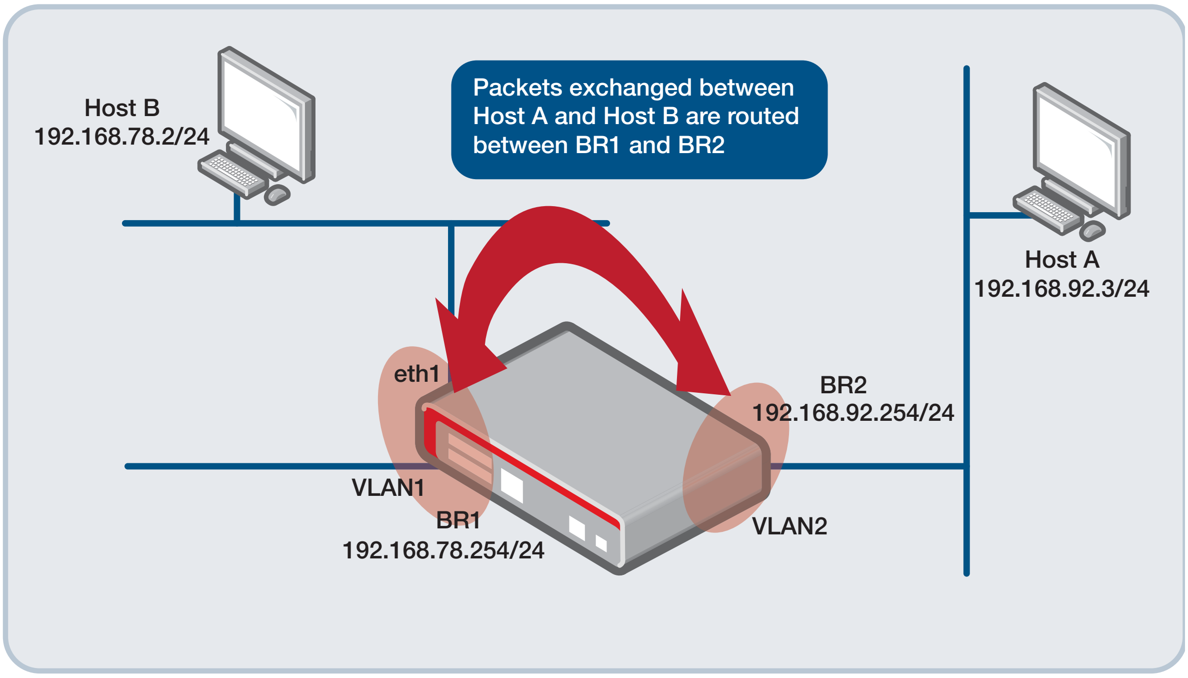
Figure 2: Bridge route between eth1 and eth2 example



It is even possible to route packets between bridge entities

From software version 5.4.8, the maximum number of bridge entities that can exist within one physical device is 255. Prior to software version 5.4.8 the limit was 64.

Figure 3: Bridge route between two bridges example



Bridge entities can have UP and DOWN events. If all the interfaces within a bridge go down, then the bridge itself is deemed to have gone down. If any one of its constituent interfaces comes up, then the bridge is deemed to have an UP event. Triggers can be configured on bridge UP or DOWN events.

Bridging show commands

Bridge show commands provide information such as the:

- content of a bridge's MAC forwarding table— **show bridge macaddr**
- state of the bridge's Layer 3 interface— **show interface *<entity ID>***
- number of octets/packets that have been sent/received by the interfaces attached to the bridge. This displays the interface counters for the specific interfaces that are part of the bridge, for example, **show interface eth1** (if eth1 is part of the bridge).
- number of octets/packets that have been sent/received by the interfaces attached to the bridge. This displays the interface counters for the specific interfaces that are part of the bridge, for example, **show interface eth1** (if eth1 is part of the bridge), or **show interface tunnel1** (if virtual tunnel interface terminating a VPN is part of the bridge).
- counters, which represent the number of octets/packets that have been exchanged between the bridge entity and the rest of the device. This includes management traffic to/from the management IP address configured on the bridge, and data routed between the bridge entity and other Layer 3 interfaces of the device— **show interface <entity ID>**.For more detail on show commands, see "Using the show commands" on page 11.

VLAN aware bridge

From software version 5.4.9-2.1 onwards, bridge 0 is available as a special case VLAN aware bridge. Prior to this, bridge 0 was not a valid bridge, bridge IDs began from 1. VLAN aware bridges have all of the features of standard bridges, including MAC address learning, filtering, and forwarding.

VLAN aware bridges forward traffic between bridge port-members based on VLAN tags. VLAN tags can be modified within a single bridge - you don’t need to configure separate bridge entities for each VLAN ID.

This means a VLAN aware bridge will forward VLAN IDs attached to Ethernet frames crossing the bridge between port-members. A VLAN aware bridge will send a frame belonging to a given VLAN ID only to bridge member ports configured to accept it. See "Example 5: Encapsulated bridge interface" on page 22 for more detail.

Simple bridge configuration

Here are the basic configuration steps and commands to create a bridge:

Step 1. **Create the bridge**	

`awplus#`
`configure terminal`
	Enter Configuration mode.

`awplus(config)#`
`bridge <id>`
	Enter your bridge **entity** ID. This is also known as a bridge-group

Step 2. **Configure the bridge**	

`awplus(config)#`
`interface br<id>`
	Enter interface configuration mode.

`awplus(config-if)#`
`ageing-time <ageing-timer>`
	(Optional), enter the time that an entry will stay in the MAC address table for the bridge before being deleted. Note: The default is 300 seconds (5 minutes).

`awplus(config-if)#`
`ip address <ipadd>`
	(Optional), enter the ip address of the bridge.

`awplus(config-if)#`
`exit`
	Exit back to Configuration mode.

`awplus(config)#`
`interface <interface-name>`
	Enter the interface name that you want add as a bridge port member. For example, vlan1.

`awplus(config-if)#`
`bridge-group <id>`
	Enter the bridge-group ID.

`awplus(config-if)#`
`exit`
	Exit back to Configuration mode.

Step 3. **Verify the configuration**	

`awplus#`
`show bridge`
	Verify the configuration using the **show bridge** command.
	For more detail, see "Using the show commands" on page 11.

Step 4. **Removing a bridge**	

`awplus#`
`configure terminal`
	Enter Configuration mode

`awplus#`
`no bridge <id>`
	Enter the no variant of the bridge command

Using the show commands

Use the **show bridge** command to check and verify your bridge configuration.

Output 1: Example output from **show bridge**

```
awplus#show bridge
Bridge Name     Aging Timer     Interfaces
------------------------------------------
br10                    300     eth1
                                eth2
br11                    100

br15                    300
```

Use the **show interface br*<id>*** command to display detailed information about the specified bridge.

Output 2: Example output from **show interface br*<id>***

```
awplus#show interface br1
Interface br1
  Link is UP, administrative state is UP
  Hardware is Bridge
  IPv4 address 192.168.1.13/24 broadcast 192.168.1.255
  index 33555969 metric 1
  MAC ageing time 300
  <UP,BROADCAST,RUNNING,MULTICAST>
  SNMP link-status traps: Disabled
    input packets 782, bytes 172480, dropped 0, multicast packets 0
    output packets 3, bytes 180, multicast packets 0 broadcast packets 0

  Time since last state change: 2 days 16:37:48
```

Use the **show bridge macaddr** command to display MAC addresses that a bridge knows about.

Output 3: Example output from **show bridge macaddr**

```
awplus#show bridge macaddr
Bridge Name      Interface        mac addr             is local?  ageing
------------------------------------------------------------------------
br10             eth1             52:54:83:e2:8b:99    no              2
br10             eth1             52:54:c0:26:73:a4    yes             0
br10             eth1             96:58:3e:02:17:8f    no            211
br10             eth2             52:54:57:14:32:13    no              6
br10             eth2             52:54:9e:c4:7f:97    yes             0
br10             eth2             a6:d0:62:b8:d5:16    no            211
```

The ‘is local?’ column ‘no’ entries reference MAC addresses that are associated with interfaces that are assigned to the bridge itself. The ‘yes’ entries reference MAC addresses that are dynamically learned. The ‘ageing’ column is a count of how many seconds it has been since the MAC address was last seen. If not refreshed, once this reaches the ageing timer value, the dynamic entry is removed from the MAC address table.

Disabling MAC-learning on bridges

From version 5.4.7-0.1 onwards, you can disable FDB MAC address learning on bridges. In some circumstances, FDB MAC address learning on a software-based router bridge is not useful, and it is better to flood the traffic within interfaces associated with the bridge instance, to ensure the traffic reaches its destination.

Use the **mac-learning** command to turn learning on and off for the desired bridge interface. For example, to turn off learning on bridge 2, use the following commands:

```
awplus(config)#interface br2
awplus(config-if)#no mac-learning
```

To turn on learning on bridge 2:

```
awplus(config)#interface br2
awplus(config-if)#mac-learning
```

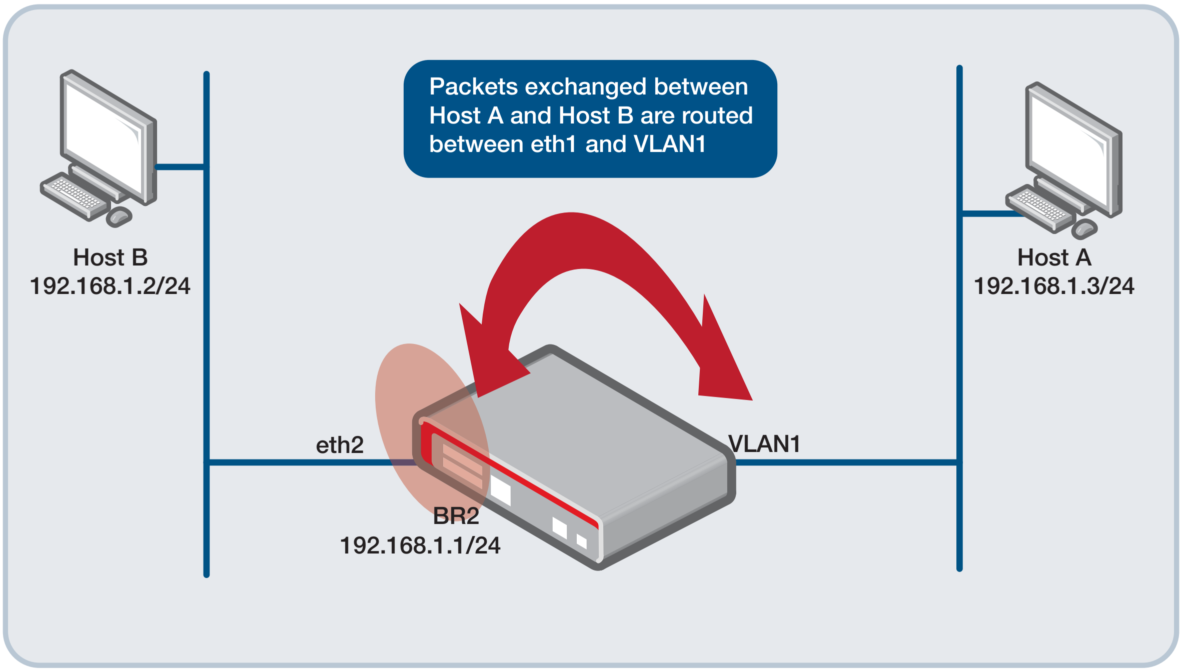
Learning is **enabled** by default.

Bridge configuration examples

Example 1: Simple bridge configuration

This example shows how to create a bridge with the ID of 2, and to assign the IP address 192.168.1.1/24. Interface vlan1 is added to bridge-group 2 and Interface eth2 is added to the bridge-group 2.

Figure 4: Example bridge configuration


Host B
192.168.1.2/24
Packets exchanged between Host A and Host B are routed between eth1 and VLAN1
Host A
192.168.1.3/24
eth2
VLAN1
BR2
192.168.1.1/24

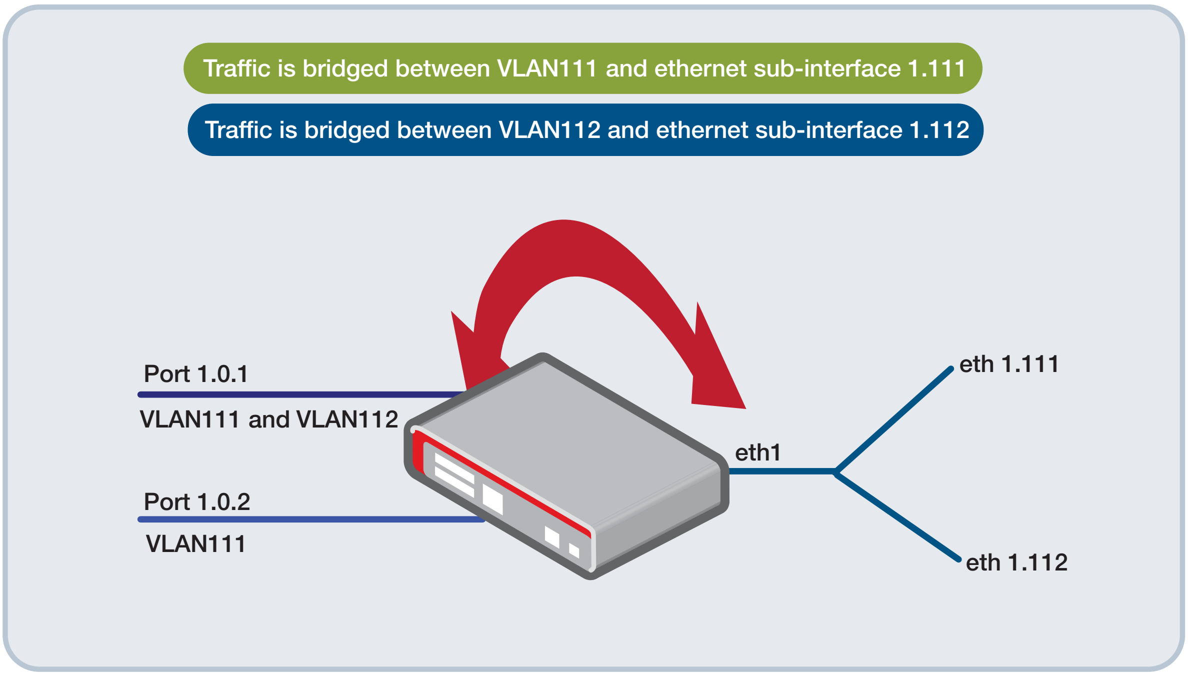
The steps to configure this example are listed below:

awplus# `configure terminal`	Enter Configuration mode.
awplus(config)# `bridge 2`	Enter your bridge-group ID.
awplus(config)# `interface br2`	Enter into Interface mode on the bridge.
awplus(config-if)# `ip address 192.168.1.1/24`	Enter the IP address.
awplus(config-if)# `exit`	Exit back to Configuration mode.
awplus(config)# `interface vlan1`	Enter into Interface mode on vlan1.
awplus(config-if)# `bridge-group 2`	Enter the bridge-group for vlan1.
awplus(config-if)# `interface eth2`	Enter into Interface mode on eth2.
awplus(config-if)# `bridge-group 2`	Enter the bridge-group for eth2.
awplus(config-if)# `exit`	Exit back to Configuration mode.
awplus(config)# `exit`	Exit back to Global Configuration mode.

Example 2: Bridging between multiple VLANS and Ethernet interfaces

This example shows how to bridge traffic between VLAN and Ethernet interfaces for multiple VLAN IDs.

Figure 5: Example bridge configuration with multiple VLANs


Traffic is bridged between VLAN111 and ethernet sub-interface 1.111
Traffic is bridged between VLAN112 and ethernet sub-interface 1.112
Port 1.0.1
VLAN111 and VLAN112
Port 1.0.2
VLAN111
eth1
eth 1.111
eth 1.112

Step 1: First, for each VLAN to be bridged, configure a bridge entity.

In this example, two VLANs are to be bridged, so two bridge entities are configured.

```
!
bridge 1
bridge 10
!
```

Step 2: Configure the VLAN IDs to be bridged in the VLAN database.

```
!
vlan database
vlan 111-112 state enable
!
```

Step 3: Associate the switch ports with the VLANs.

In this example switch port1.0.1 is 802.1q tagged member of VLANs 111 and 112, and port1.0.2 is untagged member of VLAN111.

```
!
interface port1.0.1
switchport mode trunk
switchport trunk allowed vlan add 111-112
switchport trunk native vlan none
!
interface port1.0.2
switchport access vlan111
!
```

Step 4: Configure Ethernet WAN interface with 802.1q tagged Ethernet sub interfaces and associated each VLAN ID to be bridged.

```
interface eth1
encapsulation dot1q 111
encapsulation dot1q 112
!
```

Step 5: Associate each VLAN and Ethernet sub interface with a bridge entity ID.

```
interface vlan111
bridge-group 1
!
interface vlan112
bridge-group 10
!
interface eth1.111
bridge-group 1
!
interface eth1.112
bridge-group 10
!
```

Any traffic associated with VLAN111 (bridge-group 1) remains isolated from traffic associated with VLAN112 (bridge-group 10). There is no Layer 2 traffic flows between bridge entities. There is no Layer 2 traffic flow between interfaces associated with different VLAN IDs as each VLAN is associated with a different bridge entity.

Ethernet frames via:

- Ethernet sub-interface eth1.111 are tagged with VLAN ID 111
- Ethernet sub-interface eth1.112 are tagged with VLAN ID 112
- trunked port1.0.1 have appropriate 802.1q VLAN ID tag 111 or 112 applied.
- access port1.0.2 (VLAN 111) will remain untagged.

Use the **show bridge** command to display your configuration:

Output 4: Example output from **show bridge**

```
awplus#show bridge
Bridge Name     Aging Timer     Interfaces
------------------------------------------
br1                     300     eth1.111
                                vlan111
br10                    300     eth1.112
                                vlan112
```

Use the **show interface brief** command to display a list of interfaces configured on the device.

Output 5: Example output from **show interface brief**

```
awplus#show interface brief
Interface              Status          Protocol
port1.0.1              admin up        running
port1.0.2              admin up        running
port1.0.3              admin up        down
port1.0.4              admin up        down
port1.0.5              admin up        down
port1.0.6              admin up        down
port1.0.7              admin up        down
port1.0.8              admin up        running
eth2                   admin up        down
eth1                   admin up        running
lo                     admin up        running
vlan1                  admin up        running
vlan111                admin up        running
vlan112                admin up        running
br1                    admin up        running
br10                   admin up        running
eth1.112               admin up        running
eth1.111               admin up        running
```

Example 3: Bridging a single VLAN via IPSec protected L2TPv3 VPN

You can connect to two physically separated networks, such as remote and main office networks, via an L2TPv3 Ethernet pseudo-wire VPN. This is achieved by bridging each office VLAN to a Virtual Tunnel Interface (VTI).

In the example below, the VTI is named TUNNEL1. The VTI interface is configured for tunnel mode L2TPv3 on each router. This setup shows how to bridge VLAN1 traffic between the local office, across the Internet via the L2TPv3 Ethernet pseudo-wire, and the remote office.

Frames exceeding the MTU are silently dropped because the DF (Don't Fragment) bit in the L2TPv3 IP header is set by default. When bridging, an ICMP “fragmentation needed and DF set” message is not generated. Similarly, IP TCP MSS clamping is not supported on tunnel interfaces that are a bridge member. Ideally, the hosts should support Path MTU Discovery, thus ensuring they do not generate packets that are too large for the path.

However, if hosts generating traffic to be bridged via the intermediate VPN cannot themselves perform path MTU discovery, then alternatively you can configure both **tunnel df clear}** and **MTU 1500** on the VTI. The combination of these two commands results in;

- Packets greater than MTU 1446 are not silently dropped
- L2TPv3 tunnel packets greater than the WAN MTU are fragmented, to be reassembled by the VPN peer.

The purpose of the **tunnel df {set|clear}** command (supported in 5.4.8-2.6 onwards) is to specify whether the DF bit should be set or cleared or set on L2TPv3 packets egressing the WAN. If the **tunnel DF** command is not used, then by default the DF bit is always **set** on outgoing L2TPv3 packets.

Note: If fragmentation of larger packets occur as a result of setting MTU and tunnel Do Not Fragment bit to **clear** on the VTI, this can slightly increase latency, due to the fragmentation and re-assembly that occurs. This may not be suitable for latency-sensitive applications.

Traffic transported via the L2TPv3 Ethernet pseudo-wire is optionally secured via tunnel protection IPsec configuration. For more information about IPSec, see the Internet Protocol Security (IPSec) Feature Overview and Configuration Guide.

Figure 6: Bridging a single VLAN via L2TPv3 VPN

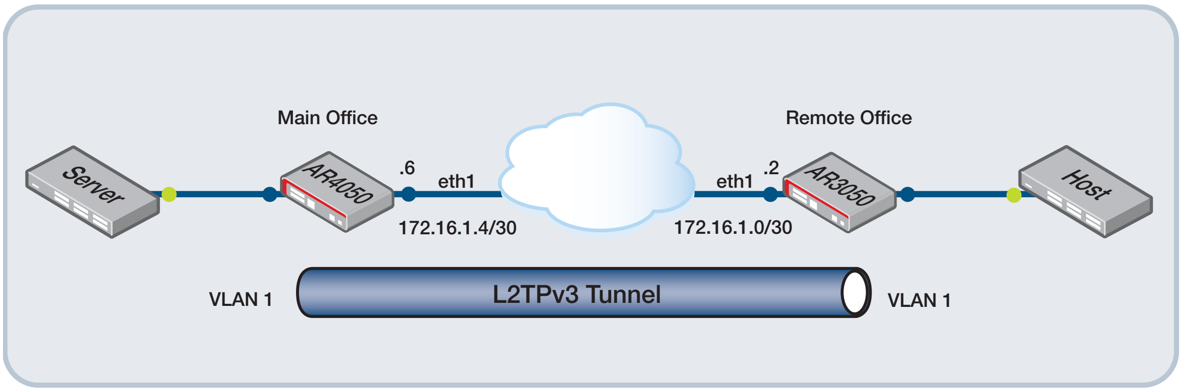
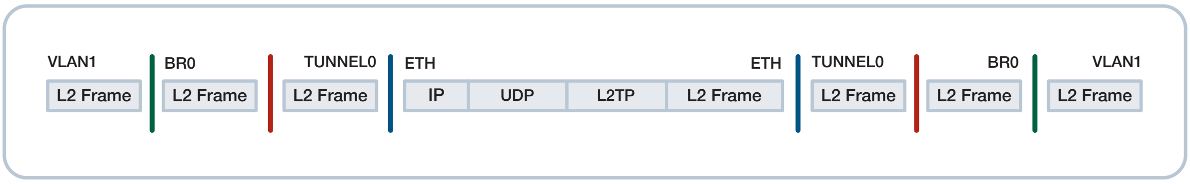


Figure 7: Encapsulation packet header



Output 6: Example AR4050S configuration

```
crypto isakmp key <sample-key> address 172.16.1.2
!
bridge 1
!
interface tunnel1
 bridge-group 1
 mtu 1500
 tunnel source eth1
 tunnel destination 172.16.1.2
 tunnel df clear
 tunnel local id 2
 tunnel remote id 1
 tunnel protection ipsec
 tunnel mode l2tp v3
!
interface vlan1
 bridge-group 1
!
```

Output 7: Example AR3050S configuration

```
!
crypto isakmp key <sample-key> address 172.16.1.6
!
bridge 1
!
interface tunnel1
 bridge-group 1
 mtu 1500
 tunnel source eth1
 tunnel destination 172.16.1.6
 tunnel df clear
 tunnel local id 1
 tunnel remote id 2
 tunnel protection ipsec
 tunnel mode l2tp v3
!
interface vlan1
 bridge-group 1
!
```

Example 4: Bridging three VLANs via IPSec protected L2TPv3 VPN

You can connect to two physically separated networks, such as remote office and main office networks, via an L2TPv3 Ethernet pseudo-wire. This is achieved by bridging each office VLAN to a Virtual Tunnel Interface (VTI). In the example below, the VTI is named TUNNEL1. Each VTI Interface is configured for tunnel mode L2TPv3.

This setup shows how to bridge VLAN1, VLAN10, and VLAN20 between the local office, across the Internet via the L2TPv3 Ethernet pseudo-wire and on to the remote office.

Three bridges are configured, one for each VLAN to be bridged. VLAN1 and the parent VTI interfaces are allocated to a bridge. Additionally, two 802.1q sub interfaces are configured on the VTI for VLAN10 and VLAN20. VLANs and dot1q VTI sub-interfaces are allocated to each bridge based on vlan ID.

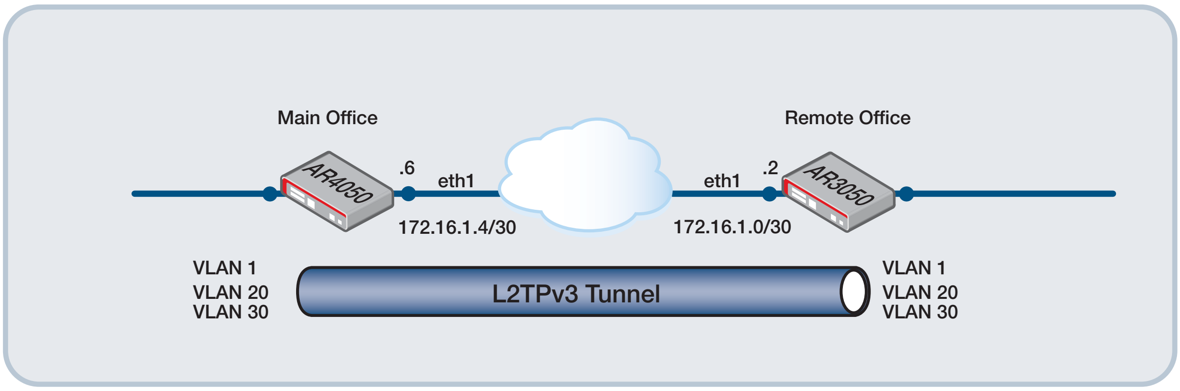
This ensures Layer 2 traffic separation for the Ethernet frames traversing the VPN.

Optionally, **mtu 1500** and **df clear** commands can be configured on the L2TPv3 mode VTI if the end hosts do not support Path MTU Discovery to allow the bridge to fragment and re-assemble tagged or untagged Ethernet frames traversing the VPN. See previous Example 3 for more details.

- Traffic transported via the L2TPv3 Ethernet pseudo-wire is optionally secured via tunnel protection IPSec configuration.

  For more information about IPSec, see the Internet Protocol Security (IPSec) Feature Overview and Configuration Guide.

Figure 8: Bridging VLANs via IPSec protected L2TPv3 VPN



Output 8: Example AR4050S configuration

```
!
crypto isakmp key <sample-key> address 172.16.1.2
!
bridge 1
bridge 2
bridge 3
!
interface vlan1
 bridge-group 1
!
interface vlan20
 bridge-group 2
!
interface vlan30
 bridge-group 3
!
interface tunnel1
 encapsulation dot1q 20
 encapsulation dot1q 30
 bridge-group 1
 mtu 1500
 tunnel source eth1
 tunnel destination 172.16.1.2
 tunnel df clear
 tunnel local id 2
 tunnel remote id 1
 tunnel protection ipsec
 tunnel mode l2tp v3
!
interface tunnel1.20
 bridge-group 2
!
interface tunnel1.30
 bridge-group 3
!
```

Output 9: Example AR3050S configuration

```
!
crypto isakmp key <sample-key> address 172.16.1.6
!
bridge 1
bridge 2
bridge 3
!
interface vlan1
 bridge-group 1
!
interface vlan20
 bridge-group 2
!
interface vlan30
 bridge-group 3
!
interface tunnel1
 encapsulation dot1q 20
 encapsulation dot1q 30
 bridge-group 1
 mtu 1500
 tunnel source eth1
 tunnel destination 172.16.1.6
 tunnel df clear
 tunnel local id 1
 tunnel remote id 2
 tunnel protection ipsec
 tunnel mode l2tp v3
!
interface tunnel1.20
 bridge-group 2
!
interface tunnel1.30
 bridge-group 3
!
```

Example 5: Encapsulated bridge interface

With the VLAN aware bridge 0, encapsulated bridge interfaces can be created and interfaces can be attached on a specific VLAN.

To add an encapsulated bridge interface to bridge 0, use the commands:

```
awplus(config)#interface br0
awplus(config-if)#encapsulation dot1q2
awplus(config-if)#interface br0.2
awplus(config-if)#ip address 192.168.1.1/24
```

To assign an interface to a specific VLAN, use the following commands:

```
awplus(config)#interface eth0
awplus(config-if)#bridge-group 0 vid2
```

MAC address bridge filtering

Basic filtering can be configured on the bridge to block/allow frames based on destination and source MAC address, and Ethernet protocol type. In this example, the goal is to filter some frames from specific MAC addresses.

The initial configuration of the devices is as follows:

- Rule ‘a’ configures a MAC-filter to filter traffic from 0000.0c00.0200 to any destination while allowing all other traffic on br2.
- Rule ‘b’ ensures all other traffic within the bridge entity is not blocked by the implicit deny all filter that is created when the bridge filtering is used within a bridge entity.

The following configuration is added:

Output 10: Configuration for adding a MAC-filter

```
mac-filter onBr2
 rule a deny smac 0000.0c00.0200
 rule b permit
!
interface br2
 mac-filter-group onBr2
!
```

Use the **show mac-filter** command to display current filters:

Output 11: Example output from **show mac-filter**

```
awplus#show mac-filter
Bridge         Rule              Options                            Pkt Count
               Dir/Action                                          Byte Count
-----------------------------------------------------------------------------
br2            a                DMAC  : any                             10254
               in/deny          SMAC  : 0000.0c00.0200                 471684
                                Proto : any
br2            b                DMAC  : any                            820020
               in/permit        SMAC  : any                           3772920
                                Proto : any
br2            Rule(default action)                                         0
               in/deny                                                      0
```

Port-protected bridge filtering

The port-protected feature is a form of bridge filtering. It creates internal filters which limit the communication possible between devices attached to specific ports within a bridge.

If you need to block all communications between specific ports within a bridge, then this feature is easy to use and avoids the need for configuring complicated filters.

Configuring bridge port-protection

To configure port-protection on a bridge:

- First, create the bridge using the **bridge** command.
- Then use the **bridge-group** command to add interfaces to the bridge.
- Use the optional **port-protected** parameter to configure the specific interface(s) you wish to enable port-protection on.

```
awplus(config-if)# bridge-group <0-255> port-protected
```

If the **port-protected** parameter is not used, then (by default) the port is added as an unprotected bridge port member.

The **port-protected** parameter ensures that no (unicast, broadcast, or multicast) traffic is forwarded between specific port members within a bridge. Traffic will continue to be forwarded (Layer 2 bridged) between protected and unprotected members within a bridge.

This type of filtering **is not** applied to:

- Layer 2 or Layer 3 control plane data. Control plane data is still received via the bridge and processed by the router CPU.
- Layer 3 traffic routed out of the bridge to another Layer 3 interface.

Example configuration

In the example configuration below:

- eth1 is a common (default unprotected) bridge port member.
- vlan10, vlan20, and vlan30 are configured as protected bridge port members.

This configuration ensures that any traffic received on any bridge-group protected port members (i.e. vlan10, vlan20, vlan30) **can** access eth1 and vice-versa. However traffic bridged between the VLAN interfaces is blocked.

Output 12: Example port-protection configuration

```
!
bridge 1
!
vlan database
 vlan 10,20,30 state enable
!
interface port1.0.1
 switchport access vlan 10
!
interface port1.0.2
 switchport access vlan 20
!
interface port1.0.3-1.0.4
 switchport access vlan 30
!
interface eth1
 bridge-group 1
!
interface vlan10,vlan20,vlan30
 bridge-group 1 port-protected
!
```

Note 1 Typically, each protected bridge port member is either a single Ethernet interface, single Ethernet sub-interface, or a VLAN interface containing only a single switchport. If a VLAN consists of more than one switchport (vlan30 in example above), then you should be aware that communications between switchports within the same VLAN (switchports 1.0.3 and 1.0.4 in example above) are not blocked. **Port-protected bridge filtering is applied at the bridge interface level only.** This filtering is not applied at the physical ***switchport*** level.

Output 13: Example output from **show bridge**

```
awplus#sh bridge
Bridge Name     Ageing Timer    Interfaces   Protected
------------------------------------------------------
br1                     300             eth1       no
                                      vlan10      yes
                                      vlan20      yes
                                      vlan30      yes
```

Note 2 If the bridge is configured with primary and secondary IP addresses as follows:

```
!
interface br1
 ip address 192.168.1.1/24
 ip address 192.168.2.1/24 secondary
!
```

Then traffic will still be Layer 3 routed between different subnets within the same bridge without being blocked, including between bridge-group protected port members. This means a host computer within the primary subnet can still communicate with a device in the secondary subnet, even though both hosts are attached to the same bridge. **Bridge filtering is not applied at the *routing* level.**

- Routed communications between the hosts can be blocked at Layer 3 by configuring firewall entities, and associated firewall rules if required.

Advanced bridge MAC filtering

Basic bridge MAC filtering enables Layer 2 filtering of frames via the bridge interface, bridge member ports and potential bridge member ports. Filtering can be configured on destination and source MAC address, and Ethernet protocol type.

Advanced Bridge MAC filtering allows the bridge to look deeper within the bridged Ethernet frames beyond the Ethernet frame header. This additionally provides:

1. **Flexible string match offset bridge filters**

   Offset bridge filtering allows packet filtering on bridge interfaces based on string match at a specific offset of the Ethernet data (payload). This is useful to check certain fields of uncommon protocols such as BACnet encapsulated within Ethernet frames.

2. **IPv4/6 L3/L4 protocol bridge filters**

   Packet filtering on bridge interfaces by IP protocol supports finer grain filtering on any combination of the following: Destination address, source address, and protocol type.

3. **Ethernet frame protocol bridge filters**

   Protocol bridge filters allow/deny Ethernet frame types (e.g. Ethernet 11, SAP, SNAP or Novell). The protocol filters examine bridged traffic before other filters configured by a ‘rule’ command.

   Protocol bridge filters have two phases:

   - Phase 1: limits packets by Ethernet frame type or ether-type/sap-type/snap-type. Permitted packets go to Phase 2 and denied packets are dropped.
   - Phase 2: performs finer-grained filtering by smac/dmac/proto, offset or IP match.

4. **Simultaneous support for PPPoE Pass-through and PPPoE Client on bridge**

   For more information on PPPoE pass-through and PPPoE client on a bridge, see the section titled: "Example filter-group configuration" on page 29.

There are two advanced bridge MAC filtering examples provided next:

- Example 1: Ethernet protocol type matching and offset filtering
- Example 2: Fine-grain IPv4/6 L3/L4 protocol filtering

Example 1: Ethernet protocol type matching and offset filtering

The following example shows how to configure the various advanced bridge filtering features.

What’s in the advanced configuration?

This sample advanced configuration consists of a single mac-filter group named **customer**. You can apply only one mac-filter group to a bridge for ingress traffic and only one mac-filter group to a bridge for egress traffic.

This mac-filter group (customer) is applied to the bridge ingress traffic and contains multiple protocol and rule actions. Protocol actions (for matching traffic based on common ethertype names, such as snap, ethii, or SAP), and rule actions (to allow or deny, based on source or destination MAC addresses). Rule actions match on a sequence of data starting from a given offset relative to the encapsulated network packet.

The sample configuration and associated show command output is as follows:

Output 14: Sample advanced bridge filtering configuration

```
!
bridge 1
!
mac-filter customer
 protocol 1 deny sap sap-type f0
 protocol 2 permit sap
 protocol 3 permit snap
 protocol 4 permit ethii ether-type ip
 protocol 5 permit ethii ether-type arp
 default-protocol-action deny
 rule 1 deny dmac ffff.ffff.ffff offset 28 hex-string f2
 rule 2 deny dmac ffff.ffff.ffff offset 28 hex-string 13
 rule 3 deny dmac ffff.ffff.ffff offset 28 hex-string 14
 rule 5 deny offset 22 hex-string 0087
 default-action permit
!
interface eth1
 bridge-group 1
!
interface vlan1
 bridge-group 1
!
interface br1
 mac-filter-group ingress customer
!
```

Here is a description of the filtering behaviors for entries in the configuration above.

ENTRY	BEHAVIOR
protocol 1 deny sap sap-type f0	All 802.2 LLC frames with DSAP and SSAP 0xF0 will be dropped.
protocol 2 permit sap	All 802.2 LLC frames will be permitted (except LSAP 0xF0 which will be dropped by "protocol 1"). Permitted by "protocol" filter will skip the rest of protocol filters and continue to rules.

ENTRY	BEHAVIOR
protocol 3 permit snap	All SNAP packets will be permitted. Note that actually "protocol 2" matches with all 802.2 frames including SNAP and Novell) so this protocol filter is not needed.
protocol 4 permit ethii ether-type ip / arp	Permit IPv4 and ARP packets. Other protocols using Ethernet II will be dropped as the default protocol action is "deny"
default-protocol-action deny	This means if there is no match in "protocol" filters, the packet will be dropped.
rule 1 / 2 / 3 deny dmac ffff.ffff.ffff offset 28 hex-string f2 / 13 / 14	Drop any broadcast packet that the data at offset 28 (note in AlliedWare Plus, the offset starts from 0) is 0xF2, 0x13 or 0x14.
rule 5 deny offset 22 hex-string 0087	Drop any packet that the data at offset 22 is 0x0087 (2 bytes match).
default-action permit	Packets not matching on any rule will be permitted.
interface br1 / mac-filter-group ingress customer	The protocol and rule filters are applied to all incoming traffic on "br1".

Output 15: Output from **show mac-filter**

```
awplus#show mac-filter
Iface            Rule         Options                                  Pkt Count
                 Dir/Action                                           Byte Count
--------------------------------------------------------------------------------
br1              1            Protocol   : SAP                                 0
                 in/deny      Sap-type   : 0xf0                                0
br1              2            Protocol   : SAP                                 0
                 in/permit    Sap-type   : any                                 0
br1              3            Protocol   : SNAP                                0
                 in/permit    Snap-type  : any                                 0
br1              4            Protocol   : Ethernet II                        41
                 in/permit    Ether-type : ip                               3276
br1              5            Protocol   : Ethernet II                         9
                 in/permit    Ether-type : arp                               414
br1              Protocol (default action)                                     0
                 in/deny                                                       0
br1              1            DMAC  : ffff.ffff.ffff                           0
                 in/deny      SMAC  : any                                      0
                              Proto : any
                              Offset: 28
                              String: f2
br1              2            DMAC  : ffff.ffff.ffff                           0
                 in/deny      SMAC  : any                                      0
                              Proto : any
                              Offset: 28
                              String: 13
br1              3            DMAC  : ffff.ffff.ffff                           0
                 in/deny      SMAC  : any                                      0
                              Proto : any
                              Offset: 28
                              String: 14
br1              5            DMAC  : any                                      0
                 in/deny      SMAC  : any                                      0
                              Proto : any
                              Offset: 22
                              String: 0087
br1              Rule (default action)                                        50
                 in/permit                                                  3690
```

Example 2: Fine-grain IPv4/6 L3/L4 protocol filtering

In the following example, two bridges are configured: bridge 1 and bridge 2. Interfaces vlan1 and vlan2 are a member of bridge 1. Interfaces vlan3 and vlan4 are a member of bridge 2.

A mac filter-group named IP is configured. This filter-group contains fine-grained filters matching specific source and destination IPv4 and IPv6 address ranges, matching telnet (TCP port 23), as well as filters matching ARP, IPv4 ICMP and IPv6 ICMP protocol numbers for IP packets encapsulated within bridged Ethernet frames.

The filters are applied to traffic ingressing bridge1 and also vlan3. The configuration example is followed by the associated **show mac-filte**r command output on page 30.

You'll notice that interfaces bridge 2 and vlans1, 2, 4 are not displayed in the **show mac-filter** command output. This is because there are no MAC filters applied to those interfaces.

Output 16: Example filter-group configuration

```
!
bridge 1
bridge 2
!
mac-filter ip
  rule 1 permit ip src 192.168.1.1 dst 10.0.0.0/8 proto tcp dport 23
  rule 2 permit ip src 10.0.0.0/8 dst 192.168.1.1 proto tcp sport 23
  rule 3 permit ipv6 src 2001::1 dst 3001::/64 proto tcp dport 23
  rule 4 permit ipv6 src 3001::/64 dst 2001::1 proto tcp sport 23
  rule 5 permit proto 0x0806
  rule 6 permit ip proto 1
  rule 7 permit ipv6 proto 58
!
interface vlan1-2
  bridge-group 1
!
interface vlan3
  bridge-group 2
  mac-filter-group ingress ip
!
interface vlan4
  bridge-group 2
!
interface br1
  mac-filter-group ingress ip
!
```

Output 17: Output from **show mac-filter**

```
awplus#show mac-filter
Iface            Rule         Options                                 Pkt Count
                 Dir/Action                                          Byte Count
-------------------------------------------------------------------------------
br1              1            IPv4 Src   : 192.168.1.1                      126
                 in/permit         Dst   : 10.0.0.0/8                      6775
                                   Proto : tcp (any:23)
br1              2            IPv4 Src   : 10.0.0.0/8                        83
                 in/permit         Dst   : 192.168.1.1                     8467
                                   Proto : tcp (23:any)
br1              3            IPv6 Src   : 2001::1                           97
                 in/permit         Dst   : 3001::/64                       7192
                                   Proto : tcp (any:23)
br1              4            IPv6 Src   : 3001::/64                         65
                 in/permit         Dst   : 2001::1                         5516
                                   Proto : tcp (23:any)
br1              5                 DMAC  : any                                9
                 in/permit         SMAC  : any                              414
                                   Proto : 0x0806
br1              6            IPv4 Src   : any                               10
                 in/permit         Dst   : any                              840
                                   Proto : 1
br1              7            IPv6 Src   : any                               15
                 in/permit         Dst   : any                             1136
                                   Proto : 58
br1              Rule (default action)                                       21
                 in/deny                                                   1452
vlan3            1            IPv4 Src   : 192.168.1.1                        0
                 in/permit         Dst   : 10.0.0.0/8                         0
                                   Proto : tcp (any:23)
vlan3            2            IPv4 Src   : 10.0.0.0/8                         0
                 in/permit         Dst   : 192.168.1.1                        0
                                   Proto : tcp (23:any)
vlan3            3            IPv6 Src   : 2001::1                            0
                 in/permit         Dst   : 3001::/64                          0
                                   Proto : tcp (any:23)
vlan3            4            IPv6 Src   : 3001::/64                          0
                 in/permit         Dst   : 2001::1                            0
                                   Proto : tcp (23:any)
vlan3            5                 DMAC  : any                                0
                 in/permit         SMAC  : any                                0
                                   Proto : 0x0806
vlan3            6            IPv4 Src   : any                                0
                 in/permit         Dst   : any                                0
                                   Proto : 1
vlan3            7            IPv6 Src   : any                                0
                 in/permit         Dst   : any                                0
                                   Proto : 58
vlan3            Rule (default action)                                        0
                 in/deny                                                      0
```

Support for PPPoE pass-through and PPPoE client on a bridge

By default, all Ethernet frames are Layer 2 bridged between bridge port member interfaces.

This includes bridging of PPPoE sessions operating between PPPoE clients attached to a bridge interface and a PPPoE Access Concentrator attached to a different interface on the same bridge.

This is commonly referred to PPPoE pass-through.

However, the **bridge itself** can also be simultaneously configured as a PPPoE client, used to establish its own PPPoE sessions. If the destination MAC address of the PPPoE frame is the bridge itself, then the PPPoE frame is terminated by the PPPoE client running on the bridge. If the destination MAC address is not the bridge, the frame will pass-through.

Note: in most situations **PPPoE relay** should be used - particularly in the case where there is a need to relay traffic from PPPoE clients located in internal LAN(s), to reach PPPoE concentrators located via WAN.
For more information about PPPoE relay, please see the section titled: ***PPPoE relay*** in the PPP Feature Overview and Configuration Guide.

PPPoE pass-through with bridging (with or without the router WAN simultaneously operating as a PPPoE client) should typically be used only in the case where there is a specific requirement to additionally use L2 bridge specific features, such as advanced bridge MAC filtering, or port-protected bridge filtering.

Example configuration

In this example, a PPPoE client (using the default PPPoE service name **any**) is configured **on the bridge**. IP traffic from vlan10 is Layer 3 routed via the bridge onto the PPPoE client WAN link.

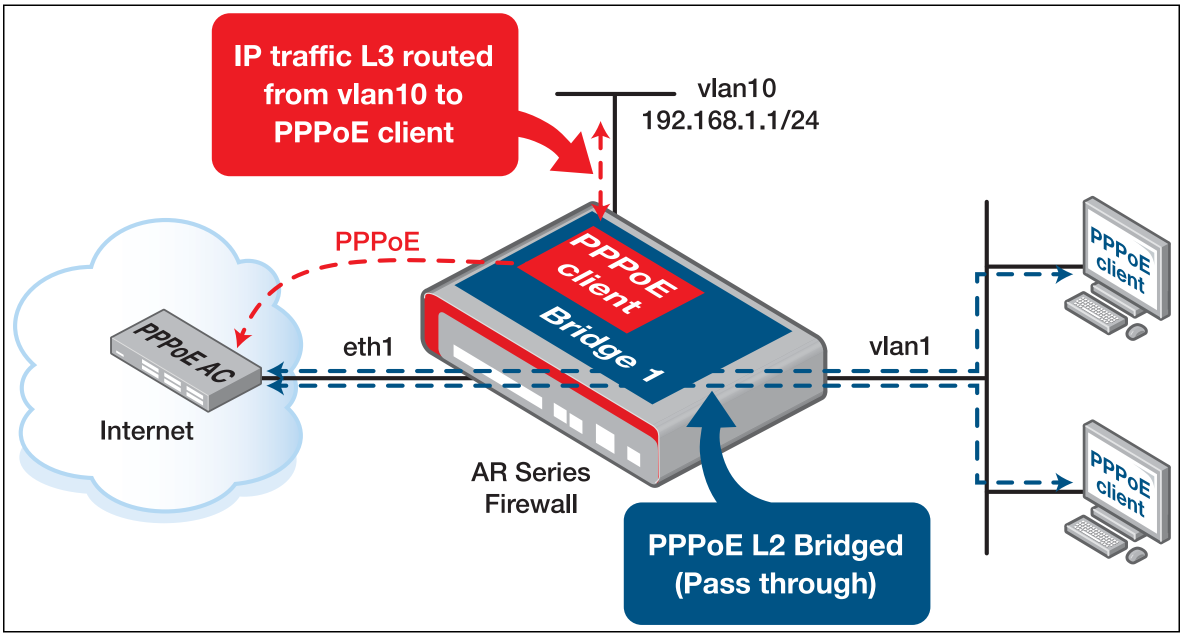
The service provider PPPoE Access Concentrator (PPPoE AC) is accessed via the eth1 WAN interface. PPPoE client terminal PC’s are attached to vlan1.

Ethernet frames are Layer 2 bridged between eth1 and vlan1. This automatically includes Layer 2 bridging of PPPoE frames from PPPoE client terminal PC’s passed-through the bridge to reach the PPPoE AC.

Notes

- Layer 2 bridge filters are not required, because the router automatically detects PPPoE frames destined for the AR-Series PPPoE client configured on the bridge. Frame detection is based on the router PPPoE Ethernet frame **dmac** address. All other PPPoE frames to reach the internal PPPoE client terminal PC are bridged from eth1 WAN to the internal LAN.
- The PPPoE client, vlan10, and the bridge itself (if configured with an IP address) can all be added as firewall interfaces. However since eth1 and vlan1 interfaces are members of the bridge, traffic bridged between these interfaces is not processed by the normal Layer 3 packet forwarding path, such as routing and firewall.

The topology diagram is as follows:



The sample configuration is as follows:

Output 18: Sample PPPoE pass-through and PPPoE client on bridge configuration

```
!
bridge 1
 encapsulation ppp 0
!
vlan database
 vlan 10 state enable
!
interface port1.0.1
 switchport access vlan 10
!
interface eth1
 bridge-group 1
!
interface vlan1
 bridge-group 1
!
interface vlan10
 ip address 192.168.1.1/24
!
interface ppp0
 ppp ipcp dns request
 keepalive
 ip address negotiated
 ppp username <username>
 ppp password <password>
!
ip route 0.0.0.0/0 ppp0
```

Allied Telesis NETWORK SMARTER

North America Headquarters | 19800 North Creek Parkway | Suite 100 | Bothell | WA 98011 | USA | T: +1 800 424 4284 | F: +1 425 481 3895
Asia-Pacific Headquarters | 11 Tai Seng Link | Singapore | 534182 | T: +65 6383 3832 | F: +65 6383 3830
EMEA & CSA Operations | Incheonweg 7 | 1437 EK Rozenburg | The Netherlands | T: +31 20 7950020 | F: +31 20 7950021

alliedtelesis.com