Technical Guide

Virtual LANs (VLANs)

Feature Overview and Configuration Guide

VLAN introduction

This guide describes Virtual LANs (VLANs), VLAN features and configuration on the switch. It begins with a description of what a VLAN is, its evolution and purpose, and also provides the meaning of some common VLAN terminology.

This is followed with a detailed look at VLAN implementation. Port-based VLAN membership is the most common way to split a network into sets of virtual LANs. We look at how this is achieved using the VLAN tagging.

The use of double-tagging (or VLAN stacking) to tunnel VLANs across Layer 2 networks is described, and an example is provided for the configuration of VLAN stacking.

Next we describe VLAN ID translation, which translates a VLAN's VLAN ID to another value for use on the wire.

Next we discuss private VLANs and the communication rules that limit what is possible between devices operating within the VLAN. AlliedWare Plus™ has three private VLAN solutions. They are private VLANs for ports in:

- **Access** mode
- **Trunked** mode
- **Upward Forwarding Only** (UFO) mode

Configuration examples are provided for these solutions.

Then, we look at combining private VLANs with other features, such as: EPSR, ARP, LLDP, GVRP, Link aggregation, and management servers. The guide ends with a section on configuring protocol based VLANs and then describes how data counters are used to count both the number of received frames or the number of received bytes (octets) belonging to a particular VLAN.

Products and software version that apply to this guide

This guide mainly applies to all AlliedWare Plus™ products, running version **5.4.4** or later.

Upward Forwarding Only (UFO) mode and VLAN ID Translation are supported by various software versions on a range of switches.

To see whether a product supports a particular feature or command, 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.

- Version 5.4.9-1.1 and later support VLAN double-tagging and VLAN ID translation on the same port on SBx908 GEN2, x950, x930, x510, x510L, IE510-28GSX, IE340L, and IE300 Series switches.
- Version 5.5.0-1.1 and later support VLAN-based Q-in-Q on the SBx8100 Series switch.
- Version 5.5.0-2.1 and later support VLAN double-tagging and VLAN ID translation on the same port on x530 Series switches.
- Version 5.5.1-0.3 and later support VLAN-based Q-in-Q with DHCP snooping on the SBx8100 Series switch.

Contents

Virtual LANs

A VLAN is a logical, software-defined subnetwork. It allows similar devices on the network to be grouped together into one broadcast domain, irrespective of their physical position in the network. Multiple VLANs can be used to group workstations, servers, and other network equipment connected to the switch, according to similar data and security requirements.

What is a VLAN?

In simple terms, a VLAN is a set of workstations within a LAN that can communicate with each other as though they were on a single, isolated LAN. What does it mean to say that they "***communicate with each other as though they were on a single, isolated LAN***"?

Among other things, it means that:

- broadcast packets sent by one of the workstations will reach all the others in the VLAN.
- broadcasts sent by one of the workstations in the VLAN will not reach any workstations that are not in the VLAN.
- broadcasts sent by workstations that are not in the VLAN will never reach workstations that are in the VLAN.
- the workstations can all communicate with each other without needing to go through a gateway. For example, IP connections would be established by ARPing for the destination.
- there is no need to send packets to the IP gateway to be forwarded on.
- the workstations can communicate with each other using non-routable protocols.

The purpose of VLANs

The basic reason for splitting a network into VLANs is to reduce congestion on a large LAN. To understand this problem, we need to look briefly at how LANs have developed over the years. Initially LANs were very flat—all the workstations were connected to a single piece of coaxial cable, or to sets of chained hubs. In a flat LAN, every packet that any device puts onto the wire gets sent to every other device on the LAN.

As the number of workstations on the typical LAN grew, they started to become hopelessly congested; there were just too many collisions, because most of the time when a workstation tried to send a packet, it would find that the wire was already occupied by a packet sent by some other device.

This next section describes the three solutions for this congestion that were developed:

- Using routers to segment LANs on page 6
- Using switches to segment LANs on page 6
- Using VLANs to segment LANs on page 7

Using routers to segment LANs

The early solution to this problem was to segment the network using routers. This would split the network into a number of smaller LANs. There would be less workstations on each LAN, and so less congestion.

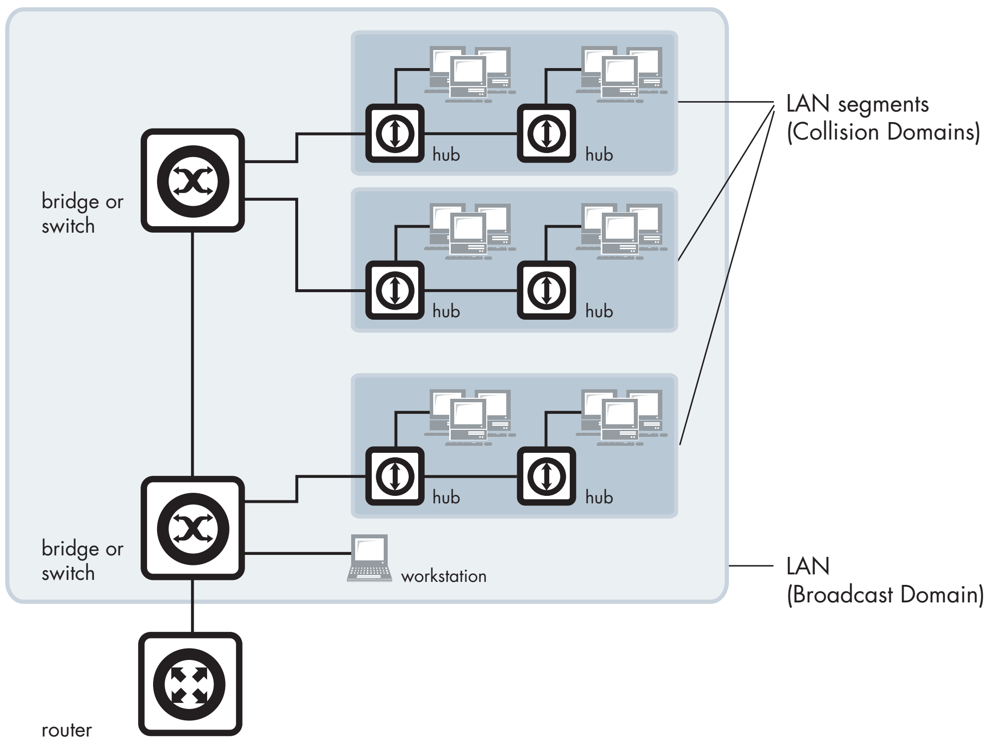
Of course, routable data being sent between LANs would have to be routed, so the layer 3 addresses would have to be organized so that each LAN had an identifiable set of addresses that could be routed to—such as an IP subnet or an Apple Talk zone. Non-routable protocols would have to be bridged, which is not quite so congestion-reducing, because bridges forward all broadcasts. But, at least for unicast packets, a bridge only forwards packets if it knows that the destination address is not in the originating LAN.

Using switches to segment LANs

As switches became more available, there was a move from chained hubs to a set of hubs connected to a switch. A switch only sends traffic to a given port if the traffic has to go to that port. So switches have the effect of reducing congestion at workstations, by stopping the workstations from seeing all the traffic from the other ports of the switch.

A simple switched network, though, still needs routers to set the boundaries of where broadcasts are sent (referred to as ‘broadcast containment’). So, the typical LAN was set up as shown below.

Figure 1: Typical VLAN



Domain terminology

The above figure introduces the concept of a LAN segment. This is also referred to as a collision domain, because when a device is trying to send a packet, it can only collide with packets sent by other devices on the same segment. Each LAN segment consists of all the devices attached to a single switch port—the switch stops packets from different ports from colliding with each other.

The LAN itself is referred to as a broadcast domain, because if any device within the LAN sends out a broadcast packet, it will be transmitted to all devices in that LAN, but not to devices beyond the LAN.

Using VLANs to segment LANs

As LANs became larger, data rates became faster, and users desired greater flexibility, the routers in a network started to become a bottleneck. This is because:

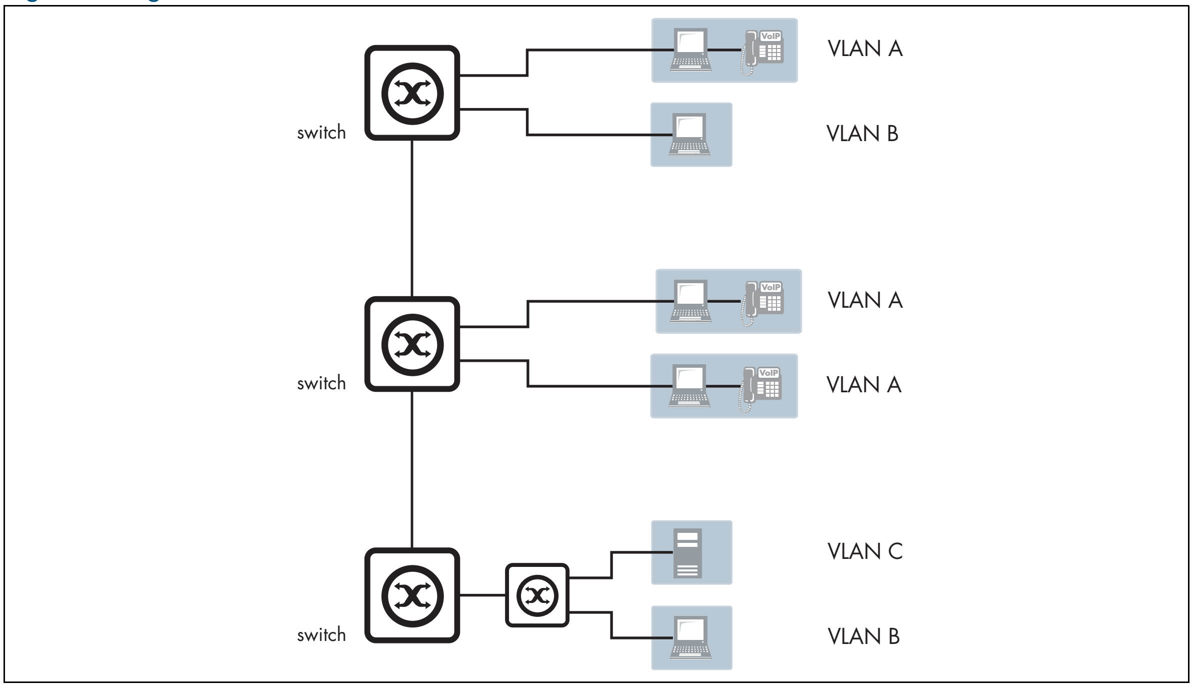
- routers typically forward data in software, and so are not as fast as switches.
- splitting up a LAN using routers meant that a LAN typically corresponded to a particular physical location. This became limiting when many users had laptops, and wanted to be able to move between buildings, but still have the same network environment wherever they plugged in.

Switch vendors started implementing methods for defining ’virtual LANs’—sets of switch ports, usually distributed across multiple switches, that somehow interacted as though they were in a single isolated LAN. This way, workstations could be separated off into separate LANs without being physically divided up by routers.

At about the same time, hubs became less popular and have been largely replaced by L2 switches. This has made the whole concept of a collision domain somewhat historical. In modern networks, a ‘collision domain’ mostly consists of a single device attached to an L2 switch port, or possibly a PC with something like an IP phone attached to it.

So, the layout of the LAN has become more like this next diagram:

Figure 2: Segmented VLAN



Instead of the LANs corresponding to physical areas divided from each other by routers, there are virtual LANs distributed across the network. For example, all the devices in the various areas labeled 'LAN A' all belong to a single virtual LAN—i.e. a single broadcast domain.

Advantages of using VLANs:

1. **Performance**. As mentioned above, routers that forward data in software become a bottleneck as LAN data rates increase. Doing away with the routers removes this bottleneck.
2. **Formation of virtual workgroups**. Because workstations can be moved from one VLAN to another just by changing the configuration on switches, it is relatively easy to put all the people working together on a particular project all into a single VLAN. They can then more easily share files and resources with each other. To be honest, though, virtual workgroups sound like a good idea in theory, but often do not work well in practice. It turns out that users are usually more interested in accessing company-wide resources (file servers, printers, etc.) than files on each others' PCs.
3. **Greater flexibility**. If users move their desks, or just move around the place with their laptops, then, if the VLANs are set up the right way, they can plug their PC in at the new location, and still be within the same VLAN. This is much harder when a network is physically divided up by routers.
4. **Ease of partitioning off resources**. If there are servers or other equipment to which the network administrator wishes to limit access, then they can be put off into their own VLAN. Then users in other VLANs can be given access selectively.

Implementing VLANs

Port-based VLANs

In the previous section, we simply stated that the network is split up into sets of virtual LANs. It is one thing to say this, it is quite another thing to understand how this is actually achieved.

Fundamentally, the act of creating a VLAN on a switch involves defining a set of ports, and defining the criteria for VLAN membership for workstations connected to those ports. By far the most common VLAN membership criterion is port-based. With port-based VLANs, the ports of a switch are simply assigned to VLANs, with no extra criteria.

Table 1: Port-based VLAN assignment

PORT	VLAN
1	1
2	1
3	2
4	1

All devices connected to a given port automatically become members of the VLAN to which that port was assigned. In effect, this just divides a switch up into a set of independent sub-switches.

Distributing a single VLAN across multiple switches

The figure "Segmented VLAN" on page 8, is an example of a VLAN-based network. It shows some of VLAN A connected to one switch, and some more of VLAN A connected to another switch. You may be asking "***Are these both part of the same VLAN A, or separate VLANs that all happen to be called VLAN A?***" The answer is that they are all parts of the same VLAN—there is a single VLAN A that is spread across two switches.

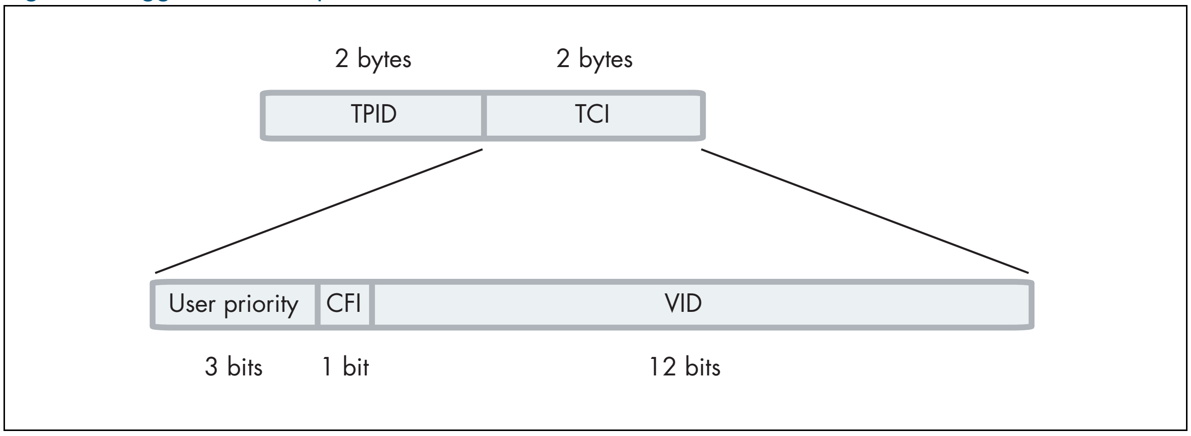
How is this achieved? How does one switch know that when it receives a broadcast packet that it associates to VLAN A that it must also forward that broadcast to other switches?

This can be done in a number of different ways, and in the early days of VLANs, just about every one of these ways was tried. Some vendors had their switches use a proprietary protocol to inform each other of their VLAN tables; some vendors used time-divided multiplexing in which different time slots were allocated to different VLANs; other vendors used frame tagging. In the end, frame tagging became the accepted standard. As we will see, in most respects this is a simple and elegant solution. However, it initially had one big downside: it required a fundamental change to the format of the Ethernet header. This split the world's Ethernet devices into those that recognized tagged headers and those that did not recognize tagged headers. In other words, a lot of Ethernet equipment was rendered obsolete.

How does tagging work?

Simply, 4 bytes are inserted into the header of an Ethernet packet. This consists of 2 bytes of Tag Protocol Identifier (TPID) and 2 bytes of Tag Control Information (TCI), as shown in the diagram below:

Figure 3: Tagged Ethernet packet



TPID is the tag protocol identifier, which indicates that a tag header is following and contains the user priority, Canonical Format Indicator (CFI), and the VLAN ID.

User Priority is a 3-bit field that allows priority information to be encoded in the frame. Eight levels of priority are allowed, where zero is the lowest priority and seven is the highest priority.

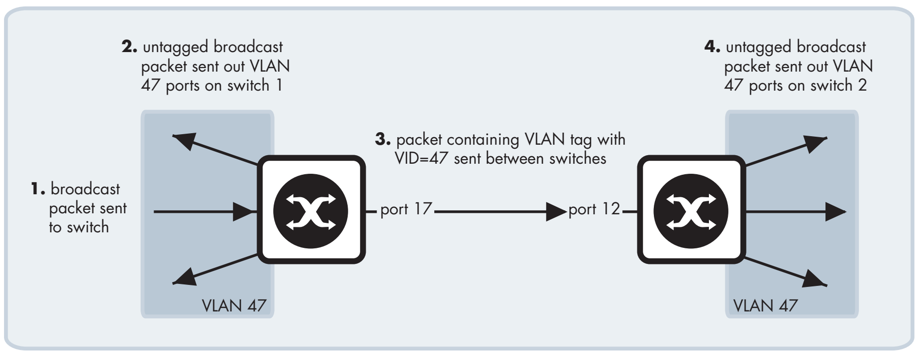
The **CFI** is a 1-bit indicator that is always set to zero for Ethernet switches. CFI is used for compatibility between Ethernet and Token Ring networks. If a frame received at an Ethernet port has a CFI set to 1, then that frame should not be bridged to an untagged port.

Then, the **VID** field contains the identifier of the VLAN. Actually, it is only the VID field that is really needed for distributing VLANs across switches—but the IEEE decided that while they were altering the format of the Ethernet header, they might as well add the User Priority and CFI too.

Let us see how this tag makes it simple to distribute VLANs across switches.

- Consider a broadcast packet arriving at a switch port. By some criterion, the packet is associated with VLAN 47, i.e. a VLAN with VLAN ID=47. Now, port 17 of this switch is connected to port 12 of another switch that also has some ports in VLAN 47.
- The network administrator needs to configure port 17 of switch 1 and port 12 of switch 2 as 'tagged' member ports of VLAN 47. This tells switch 1 to send the broadcast out port 17 as a tagged packet, with VID=47 in the tag. And it tells switch 2 to accept that tagged packet and associate it with VLAN 47.
- Then, switch 2 will send the packet out all its member ports of VLAN 47, because that is what it does with broadcasts that it has associated with VLAN 47.

Figure 4: Distribute VLANs across multiple switches


2. untagged broadcast packet sent out VLAN 47 ports on switch 1
4. untagged broadcast packet sent out VLAN 47 ports on switch 2
3. packet containing VLAN tag with VID=47 sent between switches
1. broadcast packet sent to switch
port 17
port 12
VLAN 47
VLAN 47

The tag makes it very easy for the second switch to know what to do with the packet, because the tag marks this packet as belonging to VLAN 47, and switch 2 knows exactly what it should do with packets that belong to VLAN 47.

So, there really are only **two simple rules**:

- If a port is a tagged member of a VLAN, then any packets sent out that port by that VLAN must have a tag inserted into the header.
- If a tagged packet arrives in at a port, and the port is a tagged member of the VLAN corresponding to the VID in the packet's tag, then the packet is associated with that VLAN.

With these two simple rules, it is possible to distribute VLANs across multiple switches.

Mixing tagged and untagged packets on the same port

In the previous section, we discussed using tags to indicate the VLAN membership of packets that are transferred from one switch over to another. But, it is also possible that untagged packets will be transported across that link that joins the two switches.

For example, it could be that port 17 of switch 1 is an untagged member of VLAN 56, and port 12 of switch 2 is an untagged member of VLAN 56. In this case, if switch 1 needed to transport VLAN 56 packets over to switch 2, it would send them untagged.

When those untagged packets arrived at switch 2, what VLAN would switch 2 decide to associate these packets with, given that they do not have a tag to indicate their VLAN membership? Well, in fact, switch 2 would realize that VLAN 56 is the untagged VLAN on the receiving port, so untagged packets would be deemed to belong to VLAN 56.

Obviously, a port can be an untagged member of only one port-based VLAN, otherwise there would be uncertainty about what VLAN incoming untagged packets belonged to. This VLAN is often referred to as the **native** VLAN of the port. Often, you might not want to associate a native VLAN with the port that connects a switch to another switch, so that all packets coming into that port must use a VLAN tag to indicate their VLAN membership. This stops the switch from accepting any untagged packets on the port. In AlliedWare Plus, this is achieved by configuring a port to Trunk mode and not configuring a native VLAN on it.

Only accepting packets that match the port's VLAN configuration (ingress filtering)

Consider a port that is connected to a normal workstation. Normal applications on the workstation will never send tagged packets, so there is no requirement for the switch port to accept tagged packets.

But, is there any harm if the port does accept tagged packets if they happen to come along? Well, the answer is "***quite possibly yes***". If the workstation does send tagged packets, then it is very likely doing so for malicious reasons.

To guard against such maliciousness, most switches provide the ability to configure **ingress filtering**. When ingress filtering is applied to a port, packets will only be accepted into a port if they match the VLAN configuration of that port. So, if the port is an untagged member of one VLAN, and nothing else, then only untagged packets will be accepted on the port. If the port is tagged for a set of VLANs, then a tagged packet will be accepted into the port only if it is tagged with the VID of one of the tagged VLANs configured on the port.

We highly recommend that you configure ingress filtering on all switch ports, because there is seldom a good reason for a port to accept packets from VLANs that are not configured on that port. Under AlliedWare Plus, ingress filtering is enabled on all ports by default.

The switch passes VLAN status information to the Internet Protocol (IP) module that indicates whether a VLAN is up or down. This information is used to determine route availability.

The device supports up to 4094 VLANs (the maximum allowed by the VID field in the 802.1Q tag). On some devices a few of these VLANs may be reserved for management purposes. When the switch is first powered up (and therefore unconfigured), it creates a default VLAN with a VID of 1 and an interface name of **vlan1**. In this initial condition, the switch attaches all its ports to this default VLAN.

The default VLAN cannot be deleted, and ports can only be removed from it if they also belong to at least one other VLAN. If all the devices on the physical LAN belong to the same logical LAN, that is, the same broadcast domain, then the default settings will be acceptable, and no additional VLAN configuration is required.

Configuring VLANs

Defaults By default, all switch ports are in Access mode, are associated with the default VLAN (**vlan1**), and have ingress filtering on. You cannot delete **vlan1**.

VLAN names When you create a VLAN (using the **vlan** command), you give it a numerical VLAN Identifier (VID)—a number from 2 to 4094, which is included in VLAN-tagged Ethernet frames to and from this VLAN. If tagged frames are transmitted from this VLAN, they will contain this VID in their tag. You may also give it an arbitrary alphanumeric name containing a meaningful description, which is not transmitted to other devices.

When referring to a VLAN, some commands require the VLAN to be specified by its VID while some commands require it to be specified by its interface name: vlan<VID>. In command output, the VLAN may be referred to by its VID, its interface name (vlan<VID>), or its VLAN name (the arbitrary alphanumeric string). You can name a VLAN with a string containing ‘vlan’ and its VLAN Identifier (VID). To avoid confusion, we recommend not naming it ‘vlan’ followed by any number different from its VID.

Access mode A switch port in Access mode sends untagged Ethernet frames, that is, frames without a VLAN tag. Each port is associated with one VLAN (the port-based VLAN, by default, vlan1), and when it receives untagged frames, it associates them with the VID of this VLAN. You can associate the port with another VLAN (using the **switchport access vlan** command). This removes it from the default VLAN. Use Access mode for any ports connected to devices that do not use VLAN tagging, for instance PC workstations.

Trunk mode A switch port in Trunk mode is associated with one or more VLANs for which it transmits VLAN-tagged frames, and for which it identifies incoming tagged frames with these VIDs.

To allow a switch port to distinguish and identify traffic from different VLANs, put it in Trunk mode (using the **switchport mode trunk** command), and add the VLANs (using the **switchport trunk allowed vlan** command). Use Trunk mode for ports connected to other switches which send VLAN-tagged traffic from one or more VLANs.

A Trunk mode port may also have a native VLAN (by default vlan1), for which it transmits untagged frames, and with which it associates incoming untagged frames (using the **switchport trunk native vlan** command).

Ports in Trunk mode can be enabled as promiscuous ports for private VLANs (using the **switchport mode private-vlan trunk promiscuous** command) and secondary ports for private VLANs (using the **switchport mode private-vlan trunk secondary** command). For more information about promiscuous ports, see "Promiscuous ports" on page 35.

Mirror ports A mirror port cannot be associated with a VLAN. If a switch port is configured to be a mirror port (using the **mirror interface** command), it is automatically removed from any VLAN it was associated with.

VLANs and channel groups All the ports in a channel group must have the same VLAN configuration: they must belong to the same VLANs and have the same tagging status, and can only be operated on as a group.

Procedure for configuring VLANs

Step 1: Create VLANs.

```
awplus# configure terminal
awplus(config)# vlan database
```

Create the VLANs.

```
awplus(config-vlan)# vlan <vid> [name <vlan-name>] [state {enable|
disable}]
```

or

```
awplus(config-vlan)# vlan <vid-range> [state {enable|disable}]
```

Step 2: **Associate switchports with VLANs in access mode**

Enter Interface Configuration mode for the switch ports that will be in Access mode for a particular VLAN:

```
awplus(config-vlan)# interface <port-list>
```

- associate switch ports in Access mode with VLANs.
- associate the VLAN with these ports in access mode.
- repeat for other VLANs and ports in access mode.

```
awplus(config-if)# switchport access vlan <vlan-id>
```

Step 3: **Associate switchports with VLANs in trunk mode**

Enter Interface Configuration mode for all the switch ports that will be in Trunk mode for a particular set of VLANs:

```
awplus(config-vlan)# interface <port-list>
```

Associate switch ports in Trunk mode with VLANs. Set these switch ports to Trunk mode:

```
awplus(config-if)# switchport mode trunk [ingress-filter {enable|disable}]
```

Allow these switch ports to trunk this set of VLANs:

```
awplus(config-if)# switchport trunk allowed vlan all
```

or

```
awplus(config-if)# switchport trunk allowed vlan add <vid-list>
```

By default, a Trunk mode switch port’s native VLAN, the VLAN that the port uses for untagged packet, is VLAN 1. Some control packets are untagged, including MSTP CIST BPDUs.

If required, change the native VLAN from the default. The new native VLAN must already be allowed for this switch port.

```
awplus(config-if)# switchport trunk native vlan {<vid>|none}
```

Step 4: **Confirm VLAN configuration.**

```
awplus(config-if)# exit
awplus(config)# exit
awplus# show vlan {all|brief|dynamic|static|auto|static-ports <1-4094>}
```

Port-based Q-in-Q (VLAN stacking)

Double-tagged VLANs are used to overlay a private Layer 2 network over a public Layer 2 network. The feature is also known as Nested VLANs, VLAN stacking, and Q-in-Q. It provides a method of transporting different clients' traffic across a shared Ethernet infrastructure.

Network service providers often have customers whose VLAN IDs overlap, therefore, a solution is required to control each client's traffic when the traffic from different clients is mixed together within the service providers' infrastructure, (i.e. different customers will allocate the same VIDs to their VLANs). With a nested VLAN configuration, each customer is given a customer-ID (S-Tag), which is a unique identifier within the service provider infrastructure. Traffic from individual customers is tagged with the S-Tag and segregated from other customer’s traffic. The VLAN identification of the customer’s network can be preserved while the traffic is tunneled through the network service provider’s infrastructure. You may need a special feature license to use nested VLANs. See your Allied Telesis distributor or reseller for more information.

Note that this section describes port-based Q-in-Q. For other Q-in-Q features, see also: For trunk ports, see:

- "VLAN-based Q-in-Q (VLAN stacking)" on page 20
- "VLAN double-tagging and VLAN translation on the same port" on page 27.

These are only available on some products.

How Q-in-Q works

In a nested VLAN environment VLAN tagging exists at two levels:

- Client tagging (C-tag)
- Service provider tagging (S-tag)

When nested VLAN functionality is enabled, the service provider assigns to each of its clients an individual 12 bit customer VID called an **S-tag**. The S-tag field has an identical structure to the VLAN tag field.

The switch that performs the double-tagging has two sets of specially designated ports:

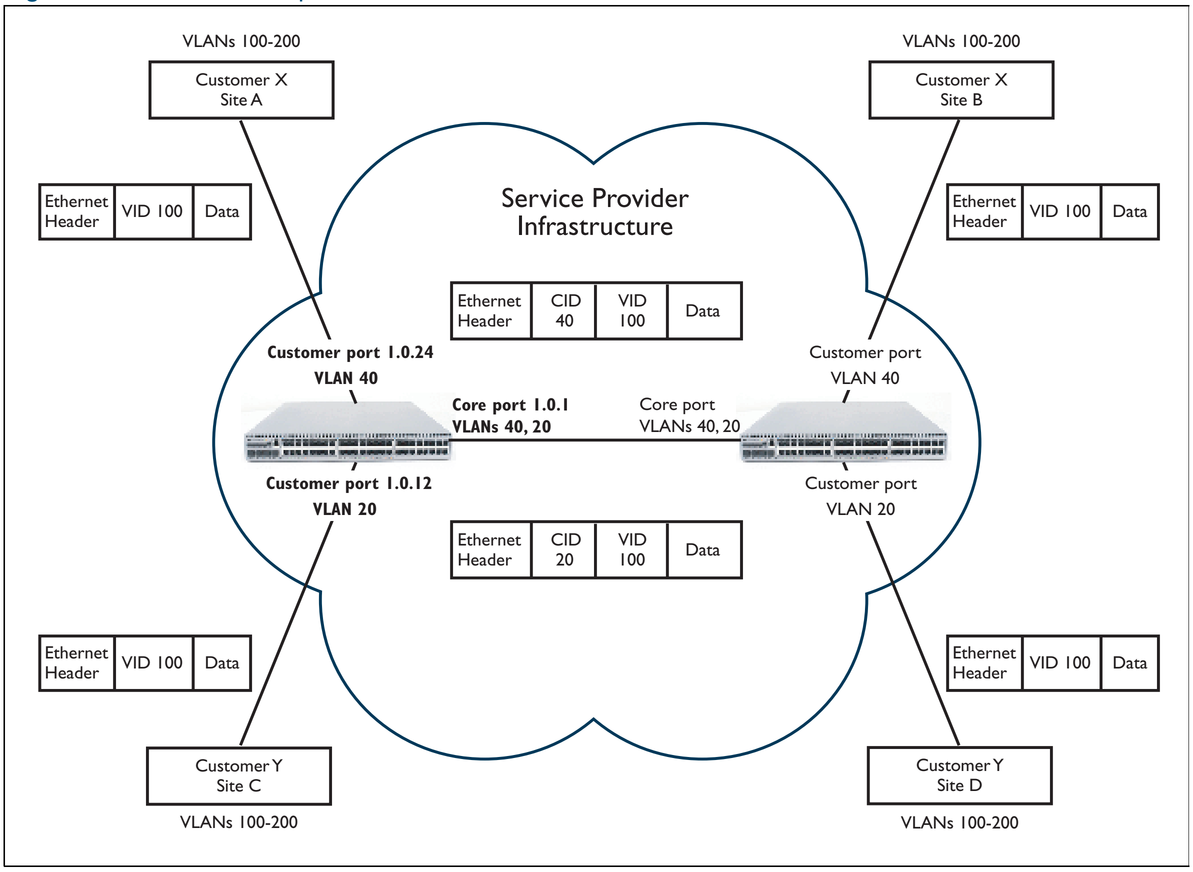
- Customer edge ports—that face the customer networks from which single-tagged packets are arriving.
- Provider ports (or Core ports)—that connect into the service provider infrastructure, on which double-tagged packets are arriving and leaving.

A customer edge port will always be a member of ONE service provider VLAN. A provider port can be a member of multiple service provider VLANs. Packets entering the customer port of the service provider switch are VLAN tagged packets with original VLAN identifiers (C-tag VIDs) from the customer network. When the packets enter the switch via a customer edge port, the switch adds an S-tag (outer tag) on top of the C-tag (inner tag). If the packet was originally untagged, then the S-tag becomes the packets one and only tag. Within the service provider infrastructure, the C-tag (inner tag) is ignored and bridging is based on the value of the S-tag. When the S-tag tagged packets exit the service provider network via a customer edge port of the destination switch, the S-tag (outer tag) is removed. Therefore, when the packets exit the customer port, the original VLAN tags are preserved.

Once the S-tag is removed from the packet, it is forwarded ‘as is’ out of the customer-edge port. The tagged status of the customer port is ignored on egress.

Nested VLAN operation is shown in the diagram below.

Figure 5: Nested VLAN operation


VLANs 100-200
Customer X Site A
VLANs 100-200
Customer X Site B
Ethernet Header | VID 100 | Data
Service Provider Infrastructure
Ethernet Header | CID 40 | VID 100 | Data
Customer port 1.0.24 VLAN 40
Customer port VLAN 40
Core port 1.0.1 VLANs 40, 20
Core port VLANs 40, 20
Customer port 1.0.12 VLAN 20
Customer port VLAN 20
Ethernet Header | CID 20 | VID 100 | Data
Customer Y Site C
VLANs 100-200
Customer Y Site D
VLANs 100-200

The Ethertype of the outer tag is configurable and can be set by changing the Tag Protocol Identifier (TPID). By default, the Ethertype is set to 0x8100.

In summary, the nested VLAN functionality makes use of the tag-in-tag technique. The inner tag comes from the customer; the outer tag comes from the core (provider) switch. The frame formats at different stages of nested VLAN operation are shown in Figure 6, Figure 7, and Figure 8 below.

Figure 6: Original standard Ethernet frame

Destination address (6 Bytes)	Source address (6 Bytes)	Length/Type (2 Bytes)	Data (0-1500 Bytes)	Frame checksum (4 Bytes)

Figure 7: 802.q VLAN tagged frame (frame entering the customer port)

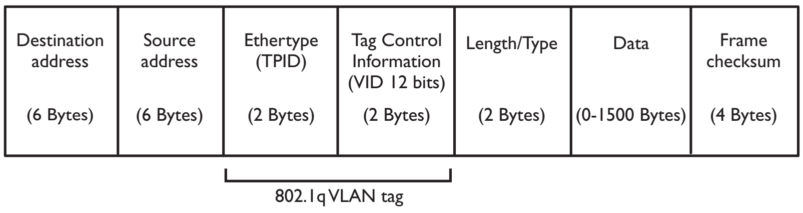
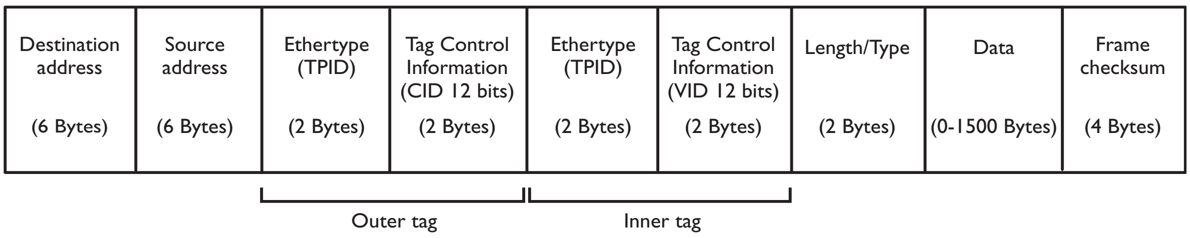


Figure 8: Nested VLAN frame with double tags (frame exiting the core port)


Destination address (6 Bytes)
Source address (6 Bytes)
Ethertype (TPID) (2 Bytes)
Tag Control Information (CID 12 bits) (2 Bytes)
Ethertype (TPID) (2 Bytes)
Tag Control Information (VID 12 bits) (2 Bytes)
Length/Type (2 Bytes)
Data (0-1500 Bytes)
Frame checksum (4 Bytes)
Outer tag
Inner tag

VLAN rules for double-tagging

These are the rules for when double-tagged VLANs are created on a:

Switch

- a nested VLAN belongs to only one customer and can have multiple customer-edge ports
- a port must be either a customer-edge port or a provider port, but cannot be both

Service provider port

- accepts only tagged packets
- transmits only tagged packets
- can be in many double-tagged VLANs

Customer edge port

- accepts both tagged and untagged packets
- transmits both tagged and untagged packets
- can be a member of only one nested VLAN

Configuring port-based Q-in-Q

You need a special feature license to use double-tagged VLANs. Contact your authorized Allied Telesis distributor or reseller for more information.

To configure a double-tagged VLAN, use the following steps:

1. Create the double-tagged VLAN.
2. Configure ports as members of the double-tagged VLAN.
3. Set the Tag Protocol Identifier (TPID).This is an optional step.

   If required, you can change the Tag Protocol Identifier (TPID) from its default (for VLAN stacking) of 0x8100 (specified as Hex notation), with the **platform vlan-stacking-tpid** command. Note that this command specifies the TPID value that applies to all VLANs used for double-tagged VLANs. You cannot set individual TPID values for different VLANs within a multi double-tagged VLAN network.
4. Set the Maximum Receive Unit (MRU).This is an optional step.

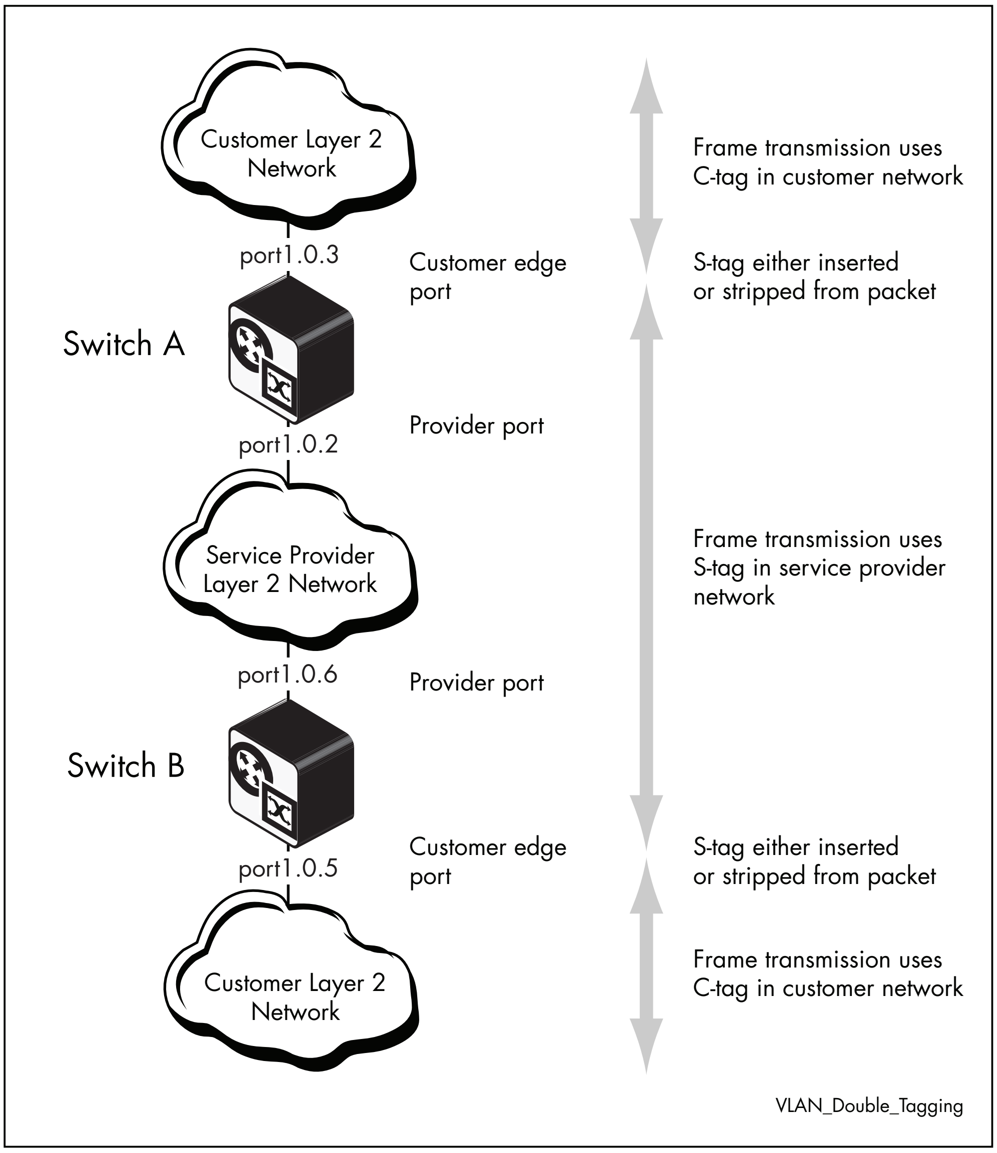
   Adding the S-tag can result in frame sizes that exceed the maximum of 1522 bytes. In order to cope with these larger than normal frames, you should increase the MRU size set for ports configured for double-tagged VLANs. Set the MRU size to:

   - 9710 bytes for ports that work at speeds of either 10 Mbps or 100 Mbps
   - 10240 bytes for ports that work at speeds of 1000 Mbps

For more information, see the Command Reference documentation of the **mru** command.

Double-tagged VLAN configuration example

Figure 9: VLAN double tagging


Customer Layer 2 Network
port1.0.3
Customer edge port
Switch A
Provider port
port1.0.2
Service Provider Layer 2 Network
port1.0.6
Provider port
Switch B
Customer edge port
port1.0.5
Customer Layer 2 Network
Frame transmission uses C-tag in customer network
S-tag either inserted or stripped from packet
Frame transmission uses S-tag in service provider network
S-tag either inserted or stripped from packet
Frame transmission uses C-tag in customer network
VLAN_Double_Tagging

Configuration procedure for double-tagged VLAN

Step 1: **Create and enable the service provider VLAN2**

This is the VLAN to be used in the outer tag, VID 2.

```
awplus# configure terminal
awplus(config)# vlan database
awplus(config-vlan)# vlan 2 state enable
awplus(config-vlan)# exit
```

Step 2: **Configure the provider-port member of the service provider VLAN**

Configure port 1.0.2 as a provider-port member of VLAN 2

```
awplus(config)# interface port1.0.2
```

Set the port to trunk mode and add the VLAN to be trunked over the port.

```
awplus(config-if)# switchport mode trunk
awplus(config-if)# switchport trunk allowed vlan add 2
```

Enable VLAN stacking (double-tagging) and set the port to be a provider port.

```
awplus(config-if)# switchport vlan-stacking provider-port
```

Step 3: **Set the Maximum Receive Unit (MRU)**

Specify the MRU size in bytes.

```
awplus(config-if)# mru 10240
awplus(config-if)# exit
```

Step 4: **Configure the customer edge port as a member of the VLAN 10**

Configure port 1.0.3 as a customer edge port member of VLAN 10

```
awplus(config)# interface port1.0.3
```

Set the port to access mode.

```
awplus(config-if)# switchport mode access
```

Associate the port with VLAN 2.

```
awplus(config-if)# switchport access vlan 2
```

Enable VLAN stacking (double-tagging) and set the port to be a customer edge port.

```
awplus(config-if)# switchport vlan-stacking customer-edge-port
awplus(config-if)# exit
```

VLAN-based Q-in-Q (VLAN stacking)

On the SBx8100 Series switch, AlliedWare Plus 5.5.0-1.1 adds support for VLAN-based Q-in-Q. VLAN-based Q-in-Q is also known as VLAN-based VLAN stacking or VLAN-based double-tagging. When you configure this VLAN stacking for a customer VLAN, it applies an outer-VLAN tag to all traffic from that VLAN as it traverses the service provider's network. The outer tag is removed for traffic from the provider network ingressing via the provider port, which is configured as a member of the outer VLAN.

Based on an inner-VLAN to outer-VLAN mapping, VLAN-based Q-in-Q:

- adds an outer-VLAN tag to traffic incoming on customer ports that belong to one or more inner VLANs
- sends the traffic out with double-tagging on the provider port or ports, which are configured as members of the outer VLAN
- removes the outer-VLAN tag from traffic incoming on the provider port or ports (members of the outer VLAN) before it goes to the customer VLAN.

The difference between VLAN-based and port-based Q-in-Q is that VLAN-based Q-in-Q applies to all ports that are members of the customer VLAN. This means that traffic incoming on any of those ports will egress on all ports that are tagged members of the outer VLAN (provider ports). The packets egress double-tagged and the outer tag is used to traverse the provider networks.

The inner-VLAN to outer-VLAN mapping is a global setting that applies on all ports that are members of the specified inner or outer VLANS.

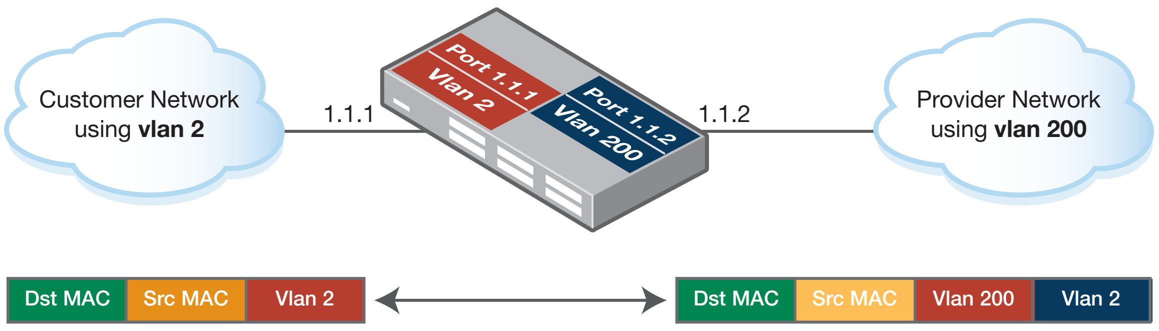
VLAN-based Q-in-Q requires the VLAN double-tagging (Q-in-Q) feature license.

Configuring VLAN-based Q-in-Q

In the following example:

- The customer network uses vlan2 and the provider network uses vlan200 for that customer network. The provider port 1.1.2 is a tagged member of vlan200; customer port 1.1.1 is a member of vlan2.
- A global mapping is added so that an outer-VLAN tag of 200 is added to incoming frames on ports that are members of vlan 2, including port 1.1.1. These frames may be untagged or tagged with VID 2.
- The outer tag is applied when the frames egress the device on port 1.1.2. In the other direction, incoming packets on provider port 1.1.2 that have an inner tag of vlan 2 and outer tag of vlan 200 will have the outer tag removed and will be sent to the customer network single-tagged with VID 2.

Figure 10: VLAN-based Q-in-Q configuration example



Configuration procedure: Add a VLAN-based Q-in-Q mapping

Step 1: **Increase TTI memory.**

By default, there is space in the switching hardware of the SBx8100 to configure up to six VLAN-based Q-in-Q mappings in TTI memory. TTI is Tunnel-Termination and Interface classification; each VLAN-based Q-n-Q mapping adds two rules to TTI. TTI rules are also used for VLAN classifiers. The memory space for the TTI rules is shared with IPv4 and IPv6 routes. You can increase the number of VLAN-based Q-in-Q mappings that can be supported by increasing the TTI rules, and this reduces the number of IPv4 and IPv6 routes supported. This setting must be configured when the device starts up.

Table 2: Platform setting to allow VLAN Q-in-Q mappings

Setting for TTI to route ratio	VLAN Q-in-Q mappings	TTI rules	Routing entries are reduced by:
default	6	12	0
low	120	240	960 IPv4 routes or 240 IPv6 routes
medium	600	1200	4800 IPv4 routes or 1200 IPv6 routes
high	1200	2400	9600 IPv4 routes or 2400 IPv6 routes

If your network will need more than six TTI rules, set the platform ratio.

```
awplus# configure terminal
awplus(config)# platform tti-to-routes {low|medium|high}
```

You will need to save the configuration and restart the device before this takes effect.

```
awplus# write
awplus# reboot
reboot system? (y/n): y
```

Step 2: **Create VLANs.**

Create and enable an outer VLAN (for the provider network) and a VLAN or range of VLANs to map to it (the customer network).

```
awplus(config)# vlan database
awplus(config-vlan)# vlan 2,200 state enable
awplus(config-vlan)# exit
```

Both inner and outer VLANs must have VIDs in the range 2 to 4094.

Step 3: **Add Q-in-Q mapping.**

Add a mapping from the customer inner VLAN to the provider outer VLAN. You add mappings globally, not per interface.

- You can configure more than one VLAN-based Q-in-Q mapping (in trunk mode) on a customer port as long as it is a tagged member of the relevant inner VLANs.

- An inner VLAN (customer VLAN) can only be mapped to one outer VLAN. Multiple inner VLANs (customer VLANs) can be mapped to the same outer VLAN.

```
awplus(config)# vlan-stacking vlan 2 outer-vlan 200
```

or from a range of customer VLANs to the provider outer VLAN:

```
awplus(config)# vlan-stacking vlan 2-5 outer-vlan 200
awplus(config)# end
```

When a VLAN-based Q-in-Q mapping is configured:

- Tagged packets incoming on an interface that is a member of the inner VID **and** that has a tag matching the inner VID have the outer-VID tag applied and are switched to interfaces that are members of the outer VLAN.
- Double-tagged packets incoming on interfaces that are tagged members of the outer VID **and** have an outer tag matching the outer VID and an inner tag matching the inner VID, will have the outer VLAN tag stripped and packets switched to interfaces that are members of the inner VLAN.
- Untagged packets incoming on ports that have an inner VLAN as the native VLAN (in trunk mode) or the access VLAN will be mapped and egress the outer VLAN ports single-tagged with the outer-VID tag.
- Packets tagged with unmapped VLANs incoming on ports that are members of the inner VLAN are switched normally; they are not mapped.

Step 4. **Add ports to the VLANs**

Add the customer port(s) to the inner VLAN.

- Customer ports can be in trunk or access mode and need to be added as members of the inner VLAN. Packets mapped with VLAN-based Q-in-Q that are outgoing on customer ports will be single-tagged with the inner VID whether the port is in trunk or access mode.
- Customer ports that are members of the inner VLAN must not be members of the outer VLAN.
- Customer ports that are in trunk mode must **not** be configured with **switchport trunk native vlan none**.

```
awplus(config)# interface port1.1.1
awplus(config-if)# switchport mode trunk
awplus(config-if)# switchport trunk allowed vlan add 2
```

Add the provider port(s) to the outer VLAN.

- Provider ports that are members of the outer VLAN must not be members of the inner VLAN.
- Provider ports must have VLAN tagging enabled (**switchport mode trunk** command) and need to be added as tagged members of the outer VLAN. A port that is in access mode cannot be added to the outer VLAN.
- If DHCP snooping is not enabled on the inner VLAN:

  ```
  awplus(config)# interface port1.1.2
  awplus(config-if)# switchport mode trunk
  ```

```
awplus(config-if)# switchport trunk allowed vlan add 200
```

- If DHCP snooping is enabled on the inner VLAN (from AlliedWare Plus 5.5.1-0.3), then the provider port must also be a trusted port and have ingress filtering disabled:

```
awplus(config)# interface port1.1.2
awplus(config-if)# switchport mode trunk ingress-filter disable
awplus(config-if)# switchport trunk allowed vlan add 200
awplus(config-if)# ip dhcp snooping trust
```

Step 5: **Check the configuration.**

```
awplus# show vlan stacking
```

```
awplus# show vlan stacking
VLAN Stacking (per-VLAN) mappings:
Outer VLAN ID  VLAN ID
=============  =============
200              2
```

Remove a VLAN-based Q-in-Q mapping

Step 1: **Remove the mapping.**

Remove the mapping for an inner VLAN or a range of inner VLANs.

```
awplus(config)# no vlan-stacking vlan 2-5
awplus(config)# end
```

Step 2: **Check the configuration.**

```
awplus# show vlan stacking
```

```
awplus#show vlan stacking
VLAN Stacking (per-VLAN) mappings:
No entries to show.
```

Interactions and restrictions

These restrictions and interactions with other features apply to VLAN-based Q-in-Q:

- VLAN-based Q-in-Q is supported on Ethernet switched interfaces, including, ports (for example, interface port1.0.3), static aggregators (for example, interface sa2) and dynamic aggregators (for example, interface po2), as long as all the ports in the aggregator are members of the appropriate inner or outer VLAN.
- VLAN-based Q-in-Q and VLAN translation are not supported on the same port.
- Port-based Q-in-Q and VLAN-based Q-in-Q are not supported on the same ports; they are supported in the same device.
- If an interface is using VLAN-based Q-in-Q, an ACL/QoS filter that matches on the inner VLAN will match on inner VLAN, but the new VLAN setting (e.g. UP/Cos) will apply to the outer-VLAN

tag. ACL/QoS filters that match on an unmapped VLANs will be unaffected, and will apply correct VLAN priority settings.

- VLAN-based Q-in-Q configuration will apply to all members within the same VCStack.
- If IGMP/MLD snooping is enabled on an inner or outer VLAN in a mapping, IGMP/MLD packets are sent to the CPU and not mapped. If IGMP/MLD snooping is disabled on the inner VLAN, IGMP/MLD packets are forwarded to the outer VLAN.
- For AlliedWare Plus version 5.5.1-0.3 and later, the SBx8100 Series supports VLAN-based Q-in-Q with DHCP snooping. The TEI part of VLAN (802.1Q) tag, the DEI (Drop Eligible Indicator, formerly CFI, part of the TEI) is 0 (zero). The priority tag (PCP field) of the outer VLAN is Q5; the priority tag of the inner VLAN is unchanged—the tag it is received with.
- For AlliedWare Plus versions 5.5.0-1.1 to 5.5.1-0.2, if DHCP snooping/relay is enabled on inner or outer VLAN in a mapping, DHCP packets are sent to the CPU and are not mapped. If DHCP snooping/relay is disabled on an inner VLAN, DHCP packets are forwarded to the outer VLAN.
- Layer 2 protocol packets including GVRP, EPSR, loop detection, UDLD and CFM incoming on inner or outer-VLAN ports are sent to the CPU and are not mapped.
- Layer 3 protocol packets, including OSPF, RIP and PIM incoming on inner- or outer-VLAN ports are mapped and forwarded to the outer-or inner-VLAN ports.
- VLAN-based Q-in-Q mappings (applied to tagged packets on trunk ports) and subnet VLAN classifiers (applied to untagged packets on access ports) do not affect each other’s operation.
- They share the same TTI memory. You can increase the number of VLAN classifiers and VLAN-based Q-in-Q mappings supported; this also reduces the number of routes supported.
- When an outer tag is added to traffic, and the traffic egresses from a provider port on the same card as the customer port, the traffic rate for this traffic is reduced by around 4%. When traffic is forwarded between cards, traffic is subject to normal inter-card limits.

VLAN ID translation

VLAN ID translation translates a VLAN ID to another value for use on the wire.

In Metro networks, it is common for the Network Service Provider (NSP) to give each customer their own unique VLAN, yet at the customer location, give all the customers the same VLAN ID for tagged packets to use on the wire. VLAN ID translation can be used by the Service Provider to change the tagged packet's VLAN ID at the customer location to the VLAN ID for tagged packets to use within the NSP's network.

VLAN ID translation is also useful in Enterprise environments where it can be used to merge two networks together without manually reconfiguring the VLAN numbering scheme. This situation can occur if two companies have merged and the same VLAN ID is used for two different purposes.

Similarly within a Network Service Provider’s network, Layer 2 networks may need to be rearranged, and VLAN translations make such rearrangement more convenient.

VLAN ID translation is supported on trunk ports on some products from AlliedWare Plus version 5.4.6, and other products from later versions. AlliedWare Plus version 5.4.9-1.1 and later also provide support on some products for "VLAN double-tagging and VLAN translation on the same port" on page 27) on trunk ports.

Enabling on x930 Series

On x930 Series switches, you need to allocate hardware space to VLAN ID translation by using the command:

```
awplus(config)# platform vlan translation enable
```

When you use this platform command, the number of L2 FDB entries reduces from 60K entries to 52K entries.

Interaction with mirroring on SBx8100 Series

Note that if you configure VLAN ID Translation on a port on an SBx81GT40, SBx81XS16 or SBx81CFC960 card, and then mirror that port’s traffic, the mirrored traffic may have the original VLAN instead of the translated VLAN.

This applies to both port mirroring (on the same switch) and remote mirroring, but only affects the mirrored copy. The original traffic will correctly egress its port with the translated VLAN.

Examples

To configure VLAN ID translation, use the following commands:

switchport vlan translation

To translate between internal vlan100 and wire-ID vlan200 on port1.0.1, use the commands:

```
awplus# configure terminal
awplus(config)# interface port1.0.1
awplus(config-if)# switchport vlan translation vlan 200 vlan 100
```

switchport vlan translation default drop

To drop inbound tagged packets if they do not match a VLAN translation entry, use the commands:

```
awplus# configure terminal
```

```
awplus(config)# interface port1.0.1
awplus(config-if)# switchport vlan translation default drop
```

show interface switchport vlan translation

To display VLAN translation information for port1.0.1 and port1.0.2, use the command:

```
awplus# show interface switchport vlan translation port1.0.1-port1.0.2
```

```
awplus#show interface switchport vlan translation port1.0.1-port1.0.2

Interface: port1.0.1
VLAN on Wire VLAN
------------- ---------------
1649 100
default drop
Interface: port1.0.2
VLAN on Wire VLAN
------------- ---------------
1650 100
default accept
```

VLAN double-tagging and VLAN translation on the same port

Version 5.4.9-1.1 and later support VLAN double-tagging and VLAN ID translation on the same port on SBx908 GEN2, x950, x930, x510, x510L, IE510-28GSX, IE340L and IE300 Series switches. Version 5.5.0-2.1 and later support this feature on x530 Series switches.

With this version, you can use VLAN double-tagging or VLAN ID translation or both on ports that support tagged packets, including switch ports in trunk mode, static channel groups, and dynamic (LACP) channel groups. VLAN double-tagging is also known as stacked VLANs, nested VLANs or Q-in-Q. On some products, previous and current versions support VLAN ID translation on ports in trunk mode ("VLAN ID translation" on page 26) and VLAN double-tagging on ports in access mode "Port-based Q-in-Q (VLAN stacking)" on page 15), but not both on the same port.

Using this feature in combination with CoS remarking is only supported on the x930 and x950 Series, and the SBx908 GEN2 switches.

In some networks, such as multi-unit buildings, customers' equipment (CPE) in different units may need to be connected to different Internet Service Providers (ISPs) via an intermediate provider network. Alternatively, a service provider may want to connect two customer premises in different locations though their provider network.

VLAN double-tagging allows a customer to connect to a service provider's network at multiple locations and use their own VLAN IDs, without requiring the service provider's equipment in between to know about the customer’s VLANs.

VLAN translation used together with VLAN double-tagging can be used to create a Layer 2 connection between two locations. The service provider can use VLAN translation and VLAN double-tagging to transport customers traffic across their network even if they are using overlapping VLANs, or can bundle customer VLANs over a single transport VLAN.

You can add one or more translation entries to a trunk port for.

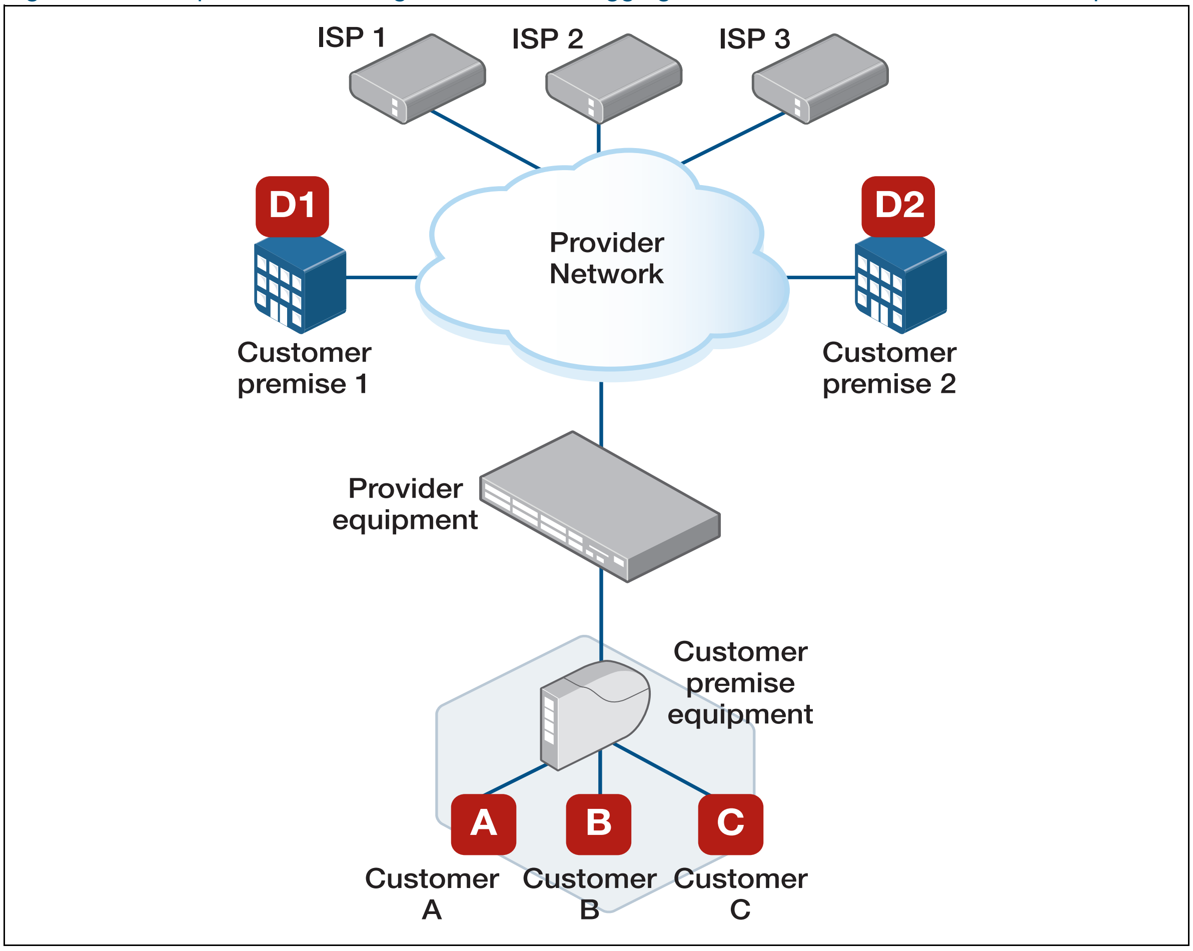
- VLAN ID translation
- VLAN double-tagging
- both VLAN ID translation and double-tagging on the same port

When an entry sets both VLAN ID translation and VLAN double-tagging for a port, the wire-VID is translated to an inner-tag VID, and an outer-tag VID is added. When packets are transmitted from the port, this is reversed—the outer tag is removed and the inner tag is translated to the wire-VID.

When VLAN double-tagging is enabled, the outer tag added or removed at the port is not included as part of the packet that is received for the purpose of MRU, so the MRU setting does not need to be changed.

The Figure 11 shows a network with three customers, A, B and C, connected to different ISPs via a single provider network by using VLAN ID translation and double-tagging, and a fourth customer D with two premises connected via the provider network by double-tagging. The translations and double-tagging are applied at the entry to the provider network.

Figure 11: Example network using VLAN double-tagging and VLAN translation on the same port



Configuration procedure: VLAN double-tagging and VLAN ID translation

Step 1: **Set the port to trunk mode.**

VLAN translation can only be configured on ports that support tagged packets. Set the port to trunk mode and allow packets with the outer-tag VID and internal translated VIDs.

```
awplus(config)# interface <switchport>
awplus(config-if)# switchport mode trunk
awplus(config-if)# switchport trunk allowed vlan add <outer-tag-vid>
```

Step 2: **Enable VLAN ID translation and/or VLAN double-tagging.**

You can create one or more rules to configure the port with VLAN ID translation, VLAN double-tagging and/or both on the same port.

- To configure VLAN ID translation, specify the wire VID to be translated and the internal VID to translate it to when received on this port.

  ```
  awplus(config-if)# switchport vlan translation vlan <wire-vid> vlan
  <vid>
  ```

- To configure VLAN double-tagging, set the outer-tag VID to be added to traffic with the specified wire VID received on this port (and removed when it is transmitted from this port).

  ```
  awplus(config-if)# switchport vlan translation vlan <wire-vid> outer-
  vlan <outer-tag-vid>
  ```

- To both configure a translation between the wire VID and the internal VID, and set an outer-tag VID to be added to these packets as they are received on the port (and removed when transmitted from the port), use the command:

  ```
  awplus(config-if)# switchport vlan translation vlan <wire-vid> vlan
  <vid> outer-vlan <outer-tag-vid>
  ```

- To drop inbound tagged packets if they do not match a VLAN translation entry, use the commands:

  ```
  awplus# configure terminal
  awplus(config)# interface <port>
  awplus(config-if)# switchport vlan translation default drop
  ```

Step 3: **Verify the configuration**

```
awplus# show interface switchport vlan translation interface <port>
```

Step 4: **Remove VLAN ID translation entries.**

To remove the VLAN ID translation and/or outer-tag settings for a specified wire VID on a port, use the command:

```
awplus(config)# interface <switchport>
awplus# no switchport vlan translation vlan <wire-vid>
```

To remove all VLAN ID translation and/or outer-tag settings on the port, use the command:

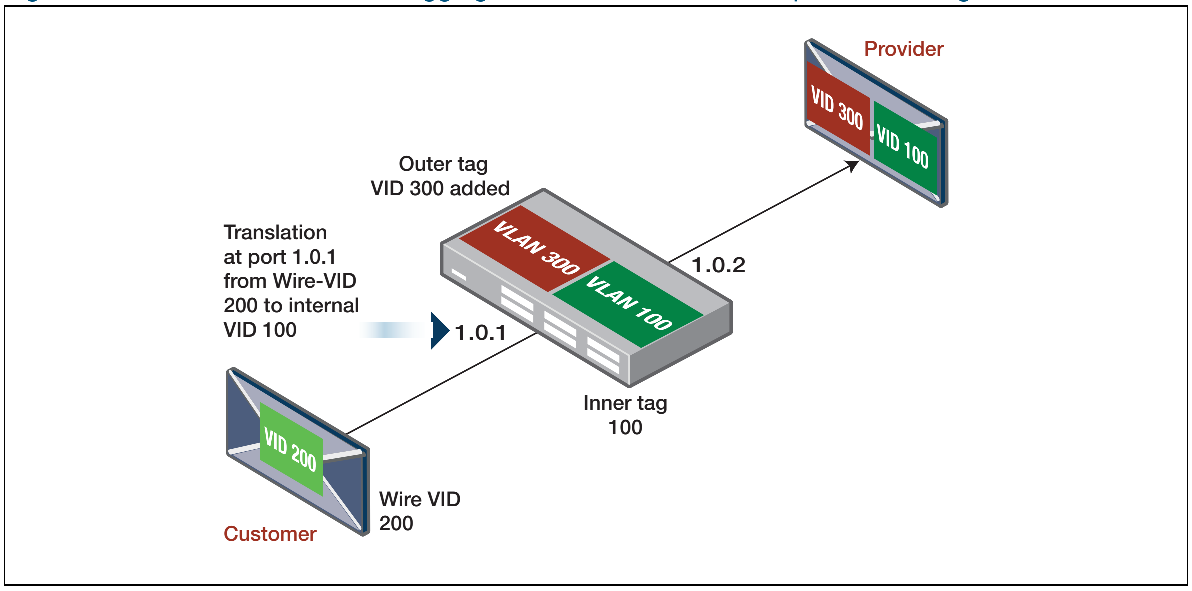
```
awplus# no switchport vlan translation all
```

Example: VLAN ID translation and double-tagging on the same port

In this example, three VLAN double-tagging and VLAN ID translation rules are applied to port1.0.1. The figures below illustrate each rule separately, but they are all configured at the same time on the same port. For the commands used to configure this example, see Figure 15 on page 1.31.

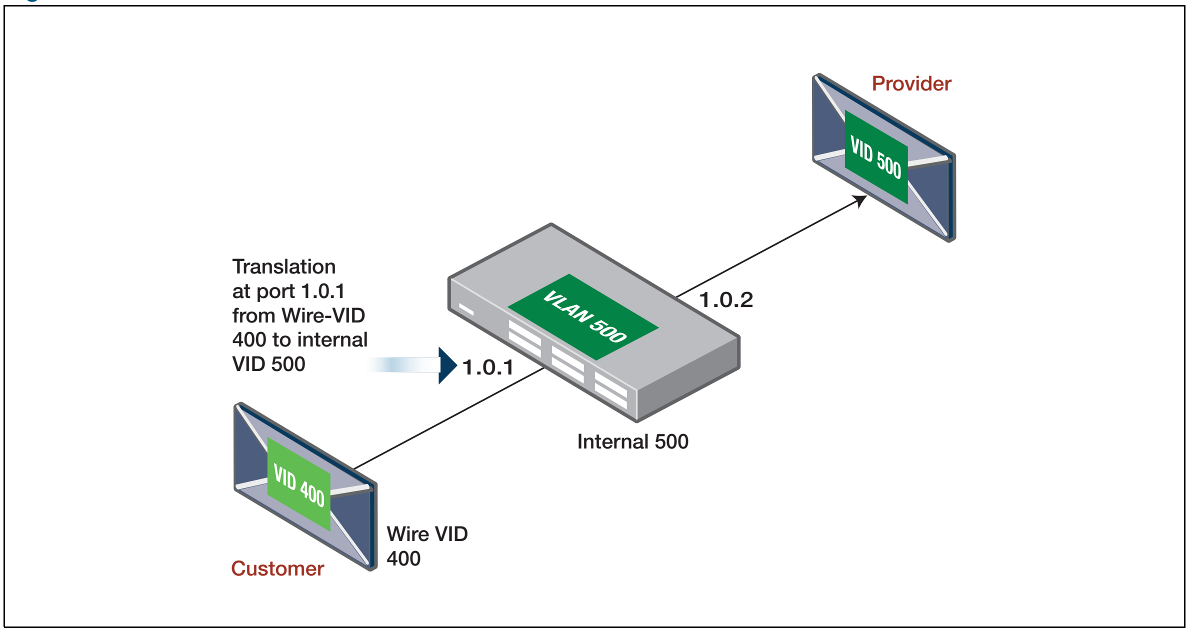
- In the first rule, VLAN translation is applied at port1.0.1 to translate wire VID 200 to an internal VID of 100. An outer-tag VID of 300 is added to the VLAN translated packets, converting them to double-tagged packets with an outer-tag VID of 300 and an inner-tag VID of 100.

Figure 12: Rule 1—VLAN double-tagging and VLAN translation on packets through a switch



- In the second rule, VLAN translation is applied at port1.0.1 to translate wire VID 400 to an internal VID of 500.

Figure 13: Rule 2—VLAN translation



- In the third rule, an outer-tag VID of 700 is added to incoming packets, converting them to double-tagged packets with an outer-tag VID of 700 and an inner-tag VID of 600.

Figure 14: Rule 3—VLAN double-tagging

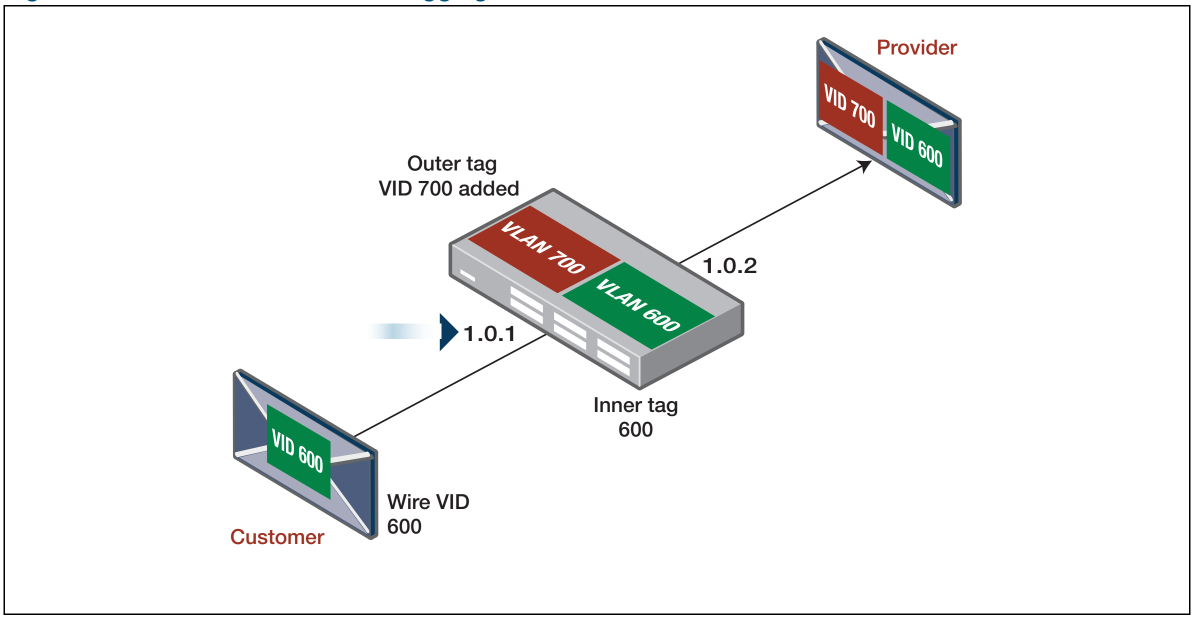


Figure 15: Example configuration for VLAN translation and double-tagging on the same port

```
!
vlan database
 vlan 100,200,300,400,500,600,700 state enable
!
...
interface port1.0.1
 description customer-side
 switchport
 switchport mode trunk
 switchport trunk allowed vlan add 100,300,500,700
 switchport vlan translation vlan 200 vlan 100 outer-vlan 300
 switchport vlan translation vlan 400 vlan 500
 switchport vlan translation vlan 600 outer-vlan 700
!
...
interface port1.0.2
 description provider-side
 switchport
 switchport mode trunk
 switchport trunk allowed vlan add 100,300,500,700
!
```

Note that switchports configured for VLAN translation must be tagged with all internally translated VIDs as well as all outer-tag VIDs being applied to incoming or VLAN translated traffic in order for the rules to work. No VLAN translation or VLAN double-tagging needs to be applied to the outgoing port (port1.0.2 in this example). Return traffic will automatically be translated back to the original VLAN and outer-VLAN tags removed.

The output from the following command confirms the translation and added outer tag configured for each wire-VID at port1.0.1:

```
awplus# show interface switchport vlan translation interface port1.0.1
```

```
awplus#show interface switchport vlan translation interface port1.0.1
VLAN on Wire             VLAN                     Outer Vlan
=================        ===================      ===================
200                      100                      300
400                      500                      0
600                      0                        700
default                  accept
```

Private VLANs

Private VLANs are VLANs with additional rules.These rules limit the communication that is possible between devices operating within the VLAN. This may be necessary for security reasons. AlliedWare Plus private VLANs provide the ability to divide a VLAN's ports into separate 'groups' and to allow communication within any individual group, but not between groups.

A group can be any number of ports or a single port. Private VLANs combine the network advantages of conventional VLANs, with an added degree of privacy obtained by limiting the connectivity between selected ports.

This section provides an introduction to:

- Private VLANs for ports in Access mode
- Private VLANs for trunked ports
- Private VLANs for ports in Upstream Forwarding Mode (UFO)

Private VLAN functionality allows your network administrator greater control over the information end users may access on the LAN.

On public networks in particular, users can be vulnerable to attack from other users on the same LAN. In addition, there is typically no real need for these users to be able to communicate directly to one another. A private VLAN is a sensible solution. It creates a situation where users are isolated from each other, and are only able to exchange packets with ports that connect to the upstream network.

Some typical scenarios that would benefit from private VLANs include:

- Hotels
- Universities, particularly student accommodation
- Libraries
- Internet cafés
- Hospitals
- Multi Dwelling Unit Internet access via a shared LAN

An example application of a private VLAN would be a library in which user booths each have a PC with Internet access. In this situation it would usually be undesirable to allow communication between these individual PCs. Connecting the PC to ports within a private isolated VLAN would enable each PC to access the Internet or a library server via a single connection, whilst preventing access between the PCs in the booths.

Another application might be to use private VLANs to simplify IP address assignment. Ports can be isolated from each other whilst still belonging to the same subnet.

AlliedWare Plus private VLAN solutions

AlliedWare Plus has three private VLAN solutions:

- Private VLANs for ports in Access mode
- Private VLANs for ports in Trunked mode
- Private VLANs for switchports ports in UFO mode

Access mode Private VLANs for ports in Access mode are based on the industry standard which have the following features:

- Primary and secondary VLANs
- Isolated and community ports are used in the secondary VLANs
- No trunked ports

Trunked mode Private VLANs for ports in Trunked mode have the following features:

- One isolated VLAN
- No community ports or VLANs
- Both the promiscuous and secondary ports are trunked members of the VLAN

UFO mode Private VLANs for switchports in UFO mode are:

- Configured on a per VLAN basis
- Can coexist with non-private VLANs
- Have member interfaces that are considered as downstream or upstream.

Let us take a look at each of these private VLAN solutions in more depth.

Private VLANs for ports in access mode

This type of private VLAN is actually a set of associated VLANs. This set consists of one primary VLAN, and one or more Secondary VLANs.

Primary VLAN

The primary VLAN is the main VLAN.

The secondary VLANs in the associated set use the primary VLAN to communicate to the rest of the network. In this way, it functions as the 'front' VLAN. The primary VLAN contains the promiscuous port, which carries the private VLAN traffic to and from the rest of the network. The promiscuous port is explained in further detail below.

Secondary VLANs

There are two types of secondary VLANs:

- Community VLAN - Ports within a community VLAN can communicate with each other, as well as with the promiscuous port. Within a community VLAN, the normal Layer2 switching functionality applies - flooding of broadcast, multicast and unknown unicast packets occurs, just as in a standard VLAN.
- Isolated VLAN - The ports in an isolated VLAN are only allowed to communicate via the promiscuous port. Note that ports within different secondary VLANs cannot communicate with each other.

Membership rules for private VLANs in access mode

The following membership rules apply when creating and operating private VLANs in access mode.

Each private VLAN:

- must contain one promiscuous port (or aggregated link)
- may contain multiple host ports
- can be configured to span switch instances
- can only contain promiscuous and host ports
- cannot use the default VLAN (vlan1)
- a private **isolated** VLAN can only contain a single promiscuous port
- a private **community** VLAN can contain more than one promiscuous port

A promiscuous port:

- is a member of the primary VLAN and all its associated secondary VLANs
- cannot be a member of both private and non-private VLANs

A host port:

- can be a member of multiple private (community) VLANs, but all these VLANs must share the same promiscuous port
- cannot be a host port in some VLANs and a non-host port in others
- cannot be a promiscuous port in another VLAN

Promiscuous ports

A promiscuous port (also known as the **uplink** port), is the one port that can communicate with all ports that are members of its associated secondary VLANs. Multiple promiscuous ports can exist in a primary VLAN, but only if the primary VLAN is only associated with community VLANS (that is, that there are no isolated VLANs associated with this primary VLAN).

A promiscuous port is a member of the primary VLAN and all associated secondary VLANs. Its Port VID is set to the VLAN ID of the primary VLAN.

The switch should always use the promiscuous port to connect to the rest of the network.

To configure a promiscuous port, use the following commands:

```
awplus# conf t
awplus(config)# int port1.0.1
awplus(config-if)# switchport mode private-VLAN promiscuous
```

Host ports

All the ports in the private VLAN, other than the Promiscuous Port(s), are referred to as **host ports**. All host ports are members of the Primary VLAN, and of one Secondary VLANs. The PVID of a host port is the VLAN ID of the Secondary VLAN it belongs to. Host ports have two levels of connectivity depending on whether they exist in an isolated or a community VLAN:

1. Host ports within an isolated VLAN

These ports are only allowed to communicate with their VLAN's promiscuous port, even though they share their secondary (isolated) VLAN with other hosts. The host ports receive their data from the promiscuous port via the primary VLAN, and **individually** transmit their data to the promiscuous port via their common secondary VLAN.

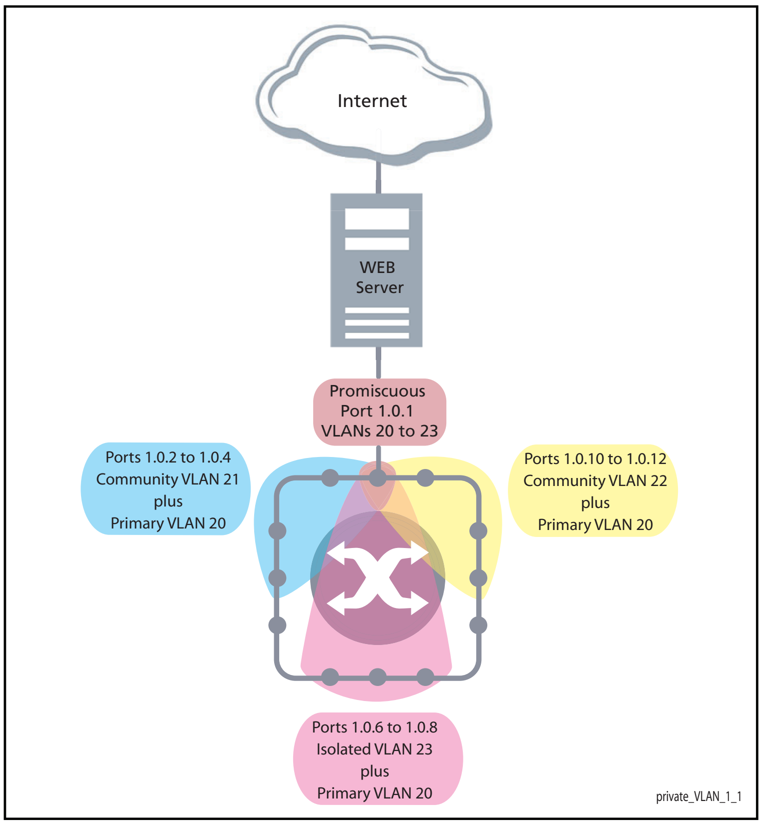
2. Host ports within a community VLAN

These ports are able to communicate with both the promiscuous port and the other ports within the community VLAN that they are associated with. They receive their data from the promiscuous port via the primary VLAN, and transmit their data to both the promiscuous port and the other host ports (within their community VLAN) via their common secondary VLAN. However, the only external path from a community VLAN is from its promiscuous port.

Private VLAN operation with ports in access mode

A basic private VLAN operation is shown in the following figure. It comprises a primary VLAN 20 plus community and isolated VLANS.

Figure 16: Private VLAN


Internet
WEB Server
Promiscuous Port 1.0.1 VLANs 20 to 23
Ports 1.0.2 to 1.0.4 Community VLAN 21 plus Primary VLAN 20
Ports 1.0.10 to 1.0.12 Community VLAN 22 plus Primary VLAN 20
Ports 1.0.6 to 1.0.8 Isolated VLAN 23 plus Primary VLAN 20
private_VLAN_1_1

The ports on this switch have the following configuration:

- Port 1.0.1 is the promiscuous port and is a member of the primary VLAN 20 and all its associated secondary VLANs.
- Ports 1.0.2 to 1.0.4 are members of the community VLAN 21 and are able to communicate with both the promiscuous port and all other ports in VLAN 21.
- Ports 1.0.10 to 1.0.12 are members of the community VLAN 22 and are able to communicate with both the promiscuous port and all other ports in VLAN 22.
- Ports 1.0.6 to 1.0.8 are members of the isolated VLAN 23. Each of these ports can only communicate with the promiscuous port.

Table 3: Private VLANs - Port Tagging

PORT	MODE	UNTAGGED VLAN MEMBERSHIP	PVID
1.0.1	Promiscuous	20, 21, 22, 23	20
1.0.2 to 1.0.4	Host	20, 21	21
1.0.10 to 1.0.12	Host	20, 22	22
1.0.6 to 1.0.8	Host	20, 23	23
1.0.5	Not members of the private VLAN		-
1.0.9	Not members of the private VLAN		-

Private VLANs operate within a single switch and comprise one primary VLAN plus a number of secondary VLANS. All data enters the private VLAN ports untagged.

Using the example of Figure 16 on page 36, data enters the switch via the promiscuous port1.0.1 and is forwarded to the host ports using VLAN 20, the primary VLAN. Data returning from the host ports to the promiscuous port (and exiting the switch) use the secondary VLAN associated with its particular host port, VLAN 21, 22, or 23 in the example.

Thus the data flows into the switch via the primary VLAN and out of the switch via the secondary VLANs. This situation is not detected outside of the switch, because all its private ports are untagged. Note however, that data flowing between ports within the same community VLAN will do so using the VID of the community VLAN.

Portfast on private VLANS

Within private VLANs, we recommend that you place all host ports into spanning-tree portfast mode and enable BPDU guard. Portfast assumes that because host ports will also be edge ports, they will have no alternative paths (loops) via other bridges. These ports are therefore allowed to move directly from the spanning-tree blocking state into the forwarding state, thus bypassing the intermediate states.

Applying BPDU guard is an extra precaution. This feature disables an edge port if it receives a BPDU frame, because receiving such a frame would indicate that the port has a connection to another network bridge.

For more information on BPDU guard and portfast, refer to the following commands in the CLI reference documentation:

- **spanning-tree portfast bpdu-guard**
- **spanning-tree portfast (STP)**

Configuration restrictions

- you cannot configure the default VLAN (vlan1) as a private VLAN.
- there can only be one Isolated VLAN associated with the Primary VLAN.
- there can be multiple Community VLANs associated with the Primary VLAN.

Access mode private VLAN configuration example

Step 1: Create the VLANs

```
awplus# configure terminal
awplus(config)# vlan database
awplus(config-vlan)# vlan 20-23
```

Step 2: Create the private VLANs and set the type

Create primary VLAN 20.

```
awplus(config-vlan)# private-vlan 20 primary
```

Create community VLAN 21.

```
awplus(config-vlan)# private-vlan 21 community
```

Create community VLAN 22.

```
awplus(config-vlan)# private-vlan 22 community
```

Create isolated VLAN 23.

```
awplus(config-vlan)# private-vlan 23 isolated
```

Step 3: Associate the secondary VLANs with the primary VLAN

Associate secondary VLAN 21 with the primary VLAN 20.

```
awplus(config-vlan)# private-vlan 20 association add 21
```

Associate secondary VLAN 22 with the primary VLAN 20.

```
awplus(config-vlan)# private-vlan 20 association add 22
```

Associate secondary VLAN 23 with the primary VLAN 20.

```
awplus(config-vlan)# private-vlan 20 association add 23
awplus(config-vlan)# exit
```

Step 4: Set port 1.0.1 to be the promiscuous port

```
awplus(config)# interface port1.0.1
awplus(config-if)# switchport mode private-vlan promiscuous
awplus(config-if)# exit
```

Step 5: Set the other ports to be host ports

```
awplus(config)# interface port1.0.2-1.0.4, port1.0.6
awplus(config-if)# switchport mode private-vlan host
awplus(config-if)# exit
```

Step 6: On the promiscuous port, map the primary VLAN to each secondary VLAN

Associate primary VLAN 20 and the secondary VLANs 21 to 23 to the promiscuous port.

```
awplus(config)# interface port1.0.1
awplus(config-if)# switchport private-vlan mapping 20 add 21-23
awplus(config-if)# exit
```

Step 7: Associate the community host ports with the community VLANs

Associate primary VLAN 20 and secondary VLAN 21 to the host ports 1.0.2 to 1.0.4.

```
awplus(config)# interface port1.0.2-1.0.4
awplus(config-if)# switchport private-vlan host-association 20 add 21
awplus(config-if)# exit
```

Associate primary VLAN 20 and secondary VLAN 22 to the host ports 1.0.10 to 1.0.12.

```
awplus(config)# interface port1.0.10-1.0.12
awplus(config-if)# switchport private-vlan host-association 20 add 22
awplus(config-if)# exit
```

Step 8: Associate the isolated host ports with the isolated VLAN 23

Associate primary VLAN 20 and secondary VLAN 23 to the host ports 1.0.6 - 1.0.8.

```
awplus(config)# interface port1.0.6-1.0.8
awplus(config-if)# switchport private-vlan host-association 20 add 23
awplus(config-if)# exit
```

Private VLANs for ports in trunked mode

A private VLAN for trunked ports consists of a single isolated VLAN. There is no concept of primary and secondary VLANs with this type of private VLAN.

The promiscuous port is in this Isolated VLAN.

- Private VLANs for trunked ports do not support community VLANs.
- Both the promiscuous and host ports must be in trunk mode

Port roles in private VLANs for trunked ports

All of the ports in this type of private VLAN are contained within the same isolated VLAN.

These private VLANs have just two types of port—the promiscuous port, and host ports.

Promiscuous port

The promiscuous port is the only port in the isolated VLAN which can send and receive frames from any other port that is a member of this VLAN.

Command

```
awplus(config-if)# switchport mode private-VLAN trunk promiscuous group
<group-id>
```

Promiscuous ports can contain multiple trunked VLANs, but these VLANs may only be either isolated, or non-private. AlliedWare Plus does not support community VLANs for this port type.

Host ports

Command

```
awplus(config-if)# switchport mode private-VLAN trunk secondary group
<group-id>
```

By default, the switch removes host ports from the default VLAN. You can add the host port to the default VLAN only if it is an isolated VLAN, and if it exists on the associated promiscuous port.

The promiscuous port can be a member of multiple Isolated VLANs.

If a host port is a member of multiple isolated VLANs, then the host port's promiscuous port must be the promiscuous port for all of these isolated VLANs

A promiscuous port in trunk mode allows you to combine multiple isolated VLANs on a single trunk port. A port in trunk mode enabled as a secondary port with the **switchport mode private-vlan trunk secondary** command can combine traffic for multiple isolated VLANs over a trunk.

A private VLAN group for trunked ports comprises the following components:

- a single promiscuous port.
- one or more isolated secondary (host) ports: These can only communicate with the associated promiscuous port.

The following membership rules apply when creating and operating private VLANs for trunked ports.

A promiscuous trunk port:

- must be in trunk mode.
- can be a member of both private VLANs and non-private VLANs.
- has a group ID that is solely used to associate the promiscuous port with secondary ports.

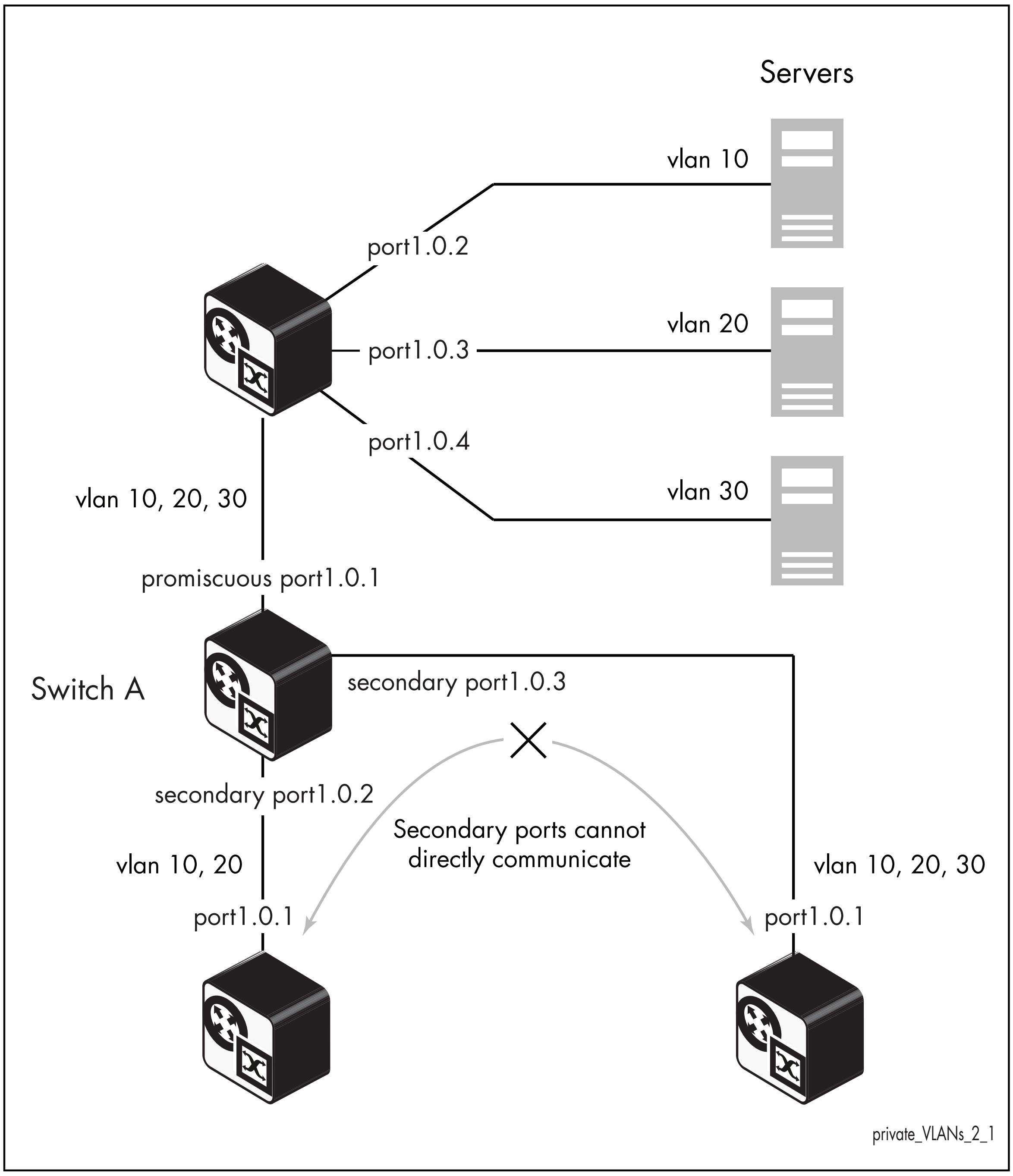
A secondary trunk port:

- must be in trunk mode.
- can only be a member of private VLANs.
- cannot be a promiscuous port in another VLAN.
- has a group ID that is solely used to associate the secondary port with its promiscuous port.

Trunked port private VLAN configuration example

A basic trunked port private VLAN operation is shown in Figure 17.

Figure 17: Trunked port private VLAN



The ports on **Switch A** have the following configuration:

- Port 1.0.1 is the promiscuous port, and has a group ID of 1.
- Port 1.0.2 is a secondary port for isolated private VLANs 10 and 20, and has a group ID of 1.
- Port 1.0.3 is a secondary port for isolated private VLANs 10, 20 and 30, and has a group ID of 1.

The configuration procedure below shows the steps to configure **Switch A**.

Step 1: Create the VLANs

```
awplus# configure terminal
awplus(config)# vlan database
awplus(config-vlan)# vlan 10,20,30
```

Step 2: Create the private VLANs and set the type

Create isolated VLAN 10.

```
awplus(config-vlan)# private-vlan 10 isolated
```

Create isolated VLAN 20.

```
awplus(config-vlan)# private-vlan 20 isolated
```

Create isolated VLAN 30.

```
awplus(config-vlan)# private-vlan 30 isolated
```

Step 3: Set port 1.0.1 to trunk mode and add the VLANs to be trunked over the port

```
awplus(config-vlan)# interface port1.0.1
awplus(config-if)# switchport mode trunk
awplus(config-if)# switchport trunk allowed vlan add 10,20,30
```

Step 4: Set port 1.0.2 to trunk mode and add the VLANs to be trunked over the port

```
awplus(config-if)# exit
awplus(config)# interface port1.0.2
awplus(config-if)# switchport mode trunk
awplus(config-if)# switchport trunk allowed vlan add 10,20
```

Step 5: Set port 1.0.3 to trunk mode and add the VLANs to be trunked over the port

```
awplus(config-if)# exit
awplus(config)# interface port1.0.3
awplus(config-if)# switchport mode trunk
awplus(config-if)# switchport trunk allowed vlan add 10,20,30
```

Step 6: Set port 1.0.1 to be the promiscuous port

Enable port 1.0.1 in trunk mode to be promiscuous port for isolated VLANs 10, 20 and 30 with a group ID of 1.

```
awplus(config-if)# exit
awplus(config)# interface port1.0.1
awplus(config-if)# switchport mode private-vlan trunk promiscuous group 1
```

Step 7: Set port 1.0.2 to be a secondary port

Enable port 1.0.1 in trunk mode to be a secondary port for isolated VLANs 10 and 20 with a group ID of 1.

```
awplus(config-if)# exit
awplus(config)# interface port1.0.2
awplus(config-if)# switchport mode private-vlan trunk secondary group 1
```

Step 8: Set port 1.0.3 to be a secondary port

Enable port 1.0.1 in trunk mode to be a secondary port for isolated VLANs 10, 20 and 30 with a group ID of 1.

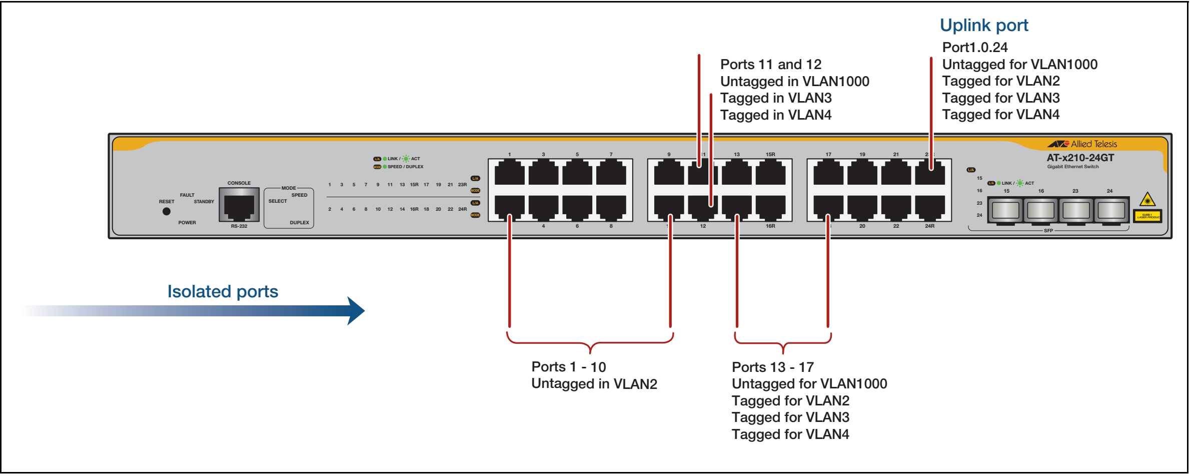
```
awplus(config-if)# exit
awplus(config)# interface port1.0.3
awplus(config-if)# switchport mode private-vlan trunk secondary group 1
```

Mixed tagged and untagged private VLANs example

A real-world customer scenario has the following requirements:

- The switch has one uplink port
- All other ports can communicate with the uplink port
- None of the non-uplink ports can communicate with any port other than the uplink port
- There are four VLANs on the switch: Data (VLAN2), SetTopBoxData(STB) (VLAN3), Multicast (VLAN4), Management (VLAN1000)
- The uplink port is untagged in VLAN1000, and tagged in VLAN2, VLAN3, VLAN4
- Ports 1- 10 are untagged in VLAN2 (just PC users connected to those ports)
- Ports 11 and 12 are untagged in VLAN100 and tagged in VLAN3 and VLAN4 (connected to STBs)
- Ports 13 - 17 are untagged in VLAN100 and tagged in VLAN2, VLAN3, VLAN4 (connected to CPEs)

Figure 18: Mixed tagged and untagged private VLAN example



The configuration that satisfies these requirements is:

```
no spanning-tree rstp enable
!
vlan database
 vlan 2-4,1000 state enable
 private-vlan 2 isolated
 private-vlan 4 isolated
 private-vlan 3 isolated
 private-vlan 1000 isolated
!
#Ports attached to the individual PCs
interface port1.0.1-1.0.10
 switchport mode trunk
 switchport trunk native vlan 2
 switchport mode private-vlan trunk secondary group 1
!

[continued ...]
```

```
#Ports attached to the STBs - management on vlan1000
interface port1.0.11-1.0.12
 switchport mode trunk
 switchport trunk allowed vlan add 3,4
 switchport trunk native vlan 1000
 switchport mode private-vlan trunk secondary group 1
!
#Ports attachd to the CPEs
interface port1.0.13-1.0.17
 switchport mode trunk
 switchport trunk allowed vlan add 2-4
 switchport trunk native vlan 1000
 switchport mode private-vlan trunk secondary group 1
!
#Uplink port
interface port1.0.24
 switchport mode trunk
 switchport trunk allowed vlan add 2-4
 switchport trunk native vlan 1000
 switchport mode private-vlan trunk promiscuous group 1
```

Private VLANs for switchports in Upstream Forwarding Only (UFO) mode

Introduction

Upstream Forwarding Only (UFO) is a private VLAN feature. It is configured on individual VLANs and blocks or isolates traffic at Layer 2. It blocks the forwarding of Ethernet frames between certain ports of a UFO VLAN while allowing forwarding of others.

All data from ports associated with a UFO VLAN must only be forwarded to the **upstream** port, which is why it is called Upstream Forwarding Only.

Each UFO VLAN has isolated and non-isolated ports. Isolated ports are called **downstream** ports. Non-isolated ports are called **upstream** ports.

As such, traffic arriving from downstream ports must be forwarded only to upstream ports. Traffic arriving from an upstream port is allowed to be forwarded to downstream ports or to other upstream ports.

How is UFO useful?

Some services need to control connections between the port and upstream device. For example, in applications such as Triple-Play networks, VLANs are often shared across subscribers and provide a specific service or set of services. Such VLANs are called service VLANs. An Internet service VLAN that is shared amongst subscribers needs to block subscribers from sending to other subscribers, while a shared voice service VLAN needs to let subscribers forward voice traffic directly with each other.

Because these two VLANs often have the same port memberships, there is a need to allow isolated VLANs to co-exist with regular VLANs on the same port(s).

Enabling private VLAN UFO on the Internet VLAN will provide isolation, while allowing the voice VLAN to remain operating as a standard or regular VLAN.

UFO features

Private VLANs for UFO have the following features:

- UFO is configured on a per VLAN basis.
- UFO VLANs can coexist with regular (non-private) VLANs in any mix on any of the same or different ports.
- Upstream traffic can be statically or dynamically configured based on topology changes detected by topology protocols.
- Multiple upstream UFO VLAN members are allowed at the same time.
- You cannot group VLANs (no primary, no secondary).

- You cannot designate a port as upstream or downstream at the port level (no community, no isolated, no promiscuous, no secondary group). Designation is only at the UFO VLAN port membership level.
- A UFO VLAN can have member interfaces that are considered as downstream or upstream.
- The UFO VLAN isolates traffic downstream to downstream.
- The UFO VLAN permits traffic downstream to upstream and upstream to downstream.

UFO advantages

UFO removes many of the Private VLAN trunk restrictions, this means that:

- Regular VLANs can now coexist with UFO VLANs on the same ports.
- VLANs can belong to the different port groups.
- Ports don't all have to be trunk ports.
- A UFO VLAN can change its upstream based on topology changes. This allows an aggregation network to be in rings, mesh, etc. and continue to isolate (i.e. block forwarding of) downstream edge ports across the aggregation network. This means that protocols such as EPSR can coexist on Private VLAN ports.

How does UFO work?

UFO works on a per VLAN basis and implements forwarding and blocking rules on traffic based on the VLANs member port designation of upstream or downstream.

The UFO forwarding/blocking rules are simple:

- Block forwarding of downstream to downstream member ports.
- Allow forwarding of downstream to upstream member ports.
- Allow forwarding of upstream to downstream member ports.
- Allow forwarding of upstream to upstream member ports.

Determining upstream and downstream

Private VLAN UFO determines the role of the member port as upstream or downstream. By default, the member port is downstream. So if the member port is not upstream it is downstream.

The upstream member port is otherwise determined based on static, dynamic, or a mix of the two configurations:

	STATICALLY	DYNAMICALLY
Determining Upstream	A private VLAN UFO can have its member ports configured statically as upstream. All other member interfaces need not be specially configured for downstream, as they are downstream by default.	A private VLAN UFO can have its member ports configured as ‘candidates’ for upstream. Whether the candidates actually become upstream or remain as downstream is based on different applications and protocols such as: ■ 1:1 Protection ■ STP/MSTP ■ EPSR ■ G.8032 (not currently supported)

Protection and redundancy

There are applications where 1:1 redundancy is used in routers or application platforms. Redundancy in these applications can be at the node or module level within a node.

There are a couple of platform classes to consider:

- Interface availability coupled with redundancy.
- Interface availability **de**coupled with redundancy.

Interface availability coupled with redundancy

Some classes of platform couple their interface availability with the availability of the nodes (or modules within a node) that are providing the redundancy. This means the nodes are either the Primary point of processing or they are the Secondary or backup points of processing, and these designations of Primary and Secondary do not change.

Under normal conditions, the Primary node (or Primary module within a node) is in the Active state, while the Secondary node (or Secondary module within a node) is in the Standby state. When the Primary node fails, the Secondary node's processors take over and become Active.

For platforms classes that tie their interfaces to these roles and states, their interfaces can also be in the Active or Standby state. That is, when a Primary node (or module) is Active, then its interfaces are also Active, and the Secondary node (or module) is in Standby then its interfaces are also in Standby. Similarly, when a Secondary node (or module) becomes Active, its interfaces also become Active.

For example, consider a Voice over IP (VoIP) Gateway with one Primary module for voice processing and protocol handling (e.g. MGCP) and one interface connected to it. A Secondary module provides backup for voice processing and protocol handling, and it too has one interface connected to it.

The two interfaces are connected together via a Layer 2 network and provide **heartbeat** messaging with one another to ensure they are running. Using UFO, the VoIP Gateway is at the upstream point of a Layer 2 network, so the interface performing the Active role is considered as upstream, and the interface performing the Standby role is considered as downstream.

Private VLAN UFO supports this by allowing UFO VLAN member ports to be configured as Primary or as Secondary. The member ports configured as Primary are always upstream, as long as one or more of these Primary ports are available for carrying traffic. Technically for UFO, a member port is considered as available if it is ‘Link Up’ and not blocked at the port level by protocols such as STP, or at the UFO VLAN level by protocols such as MSTP.

- If at least one Primary member port of the UFO VLAN is available, then the Secondary member port(s) are downstream.
- If all the Primary member ports are not available, then the Secondary member ports take over as upstream.
- When at least one Primary member port becomes available again, the roles switch and the Primary member port(s) become the upstream once again, causing the Secondary member ports to change back to downstream.

Interface availability decoupled with redundancy

Most classes of platforms do not couple their interface availability with the availability of the nodes (or modules within a node) that are providing the redundancy. A good example of this is VRRP (Virtual Router Redundancy Protocol) setup. In a VRRP setup, interfaces do not go ‘Link Up’ or ‘Link Down’ based on which VRRP node is Master.

Typically, in a VRRP setup, you only statically configure UFO member interfaces as ‘Primary’ interfaces (no ‘Secondary’ is needed) as this will be sufficient for providing multiple upstream interfaces to the redundant VRRP nodes. The two VRRP nodes can send VRRP advertisements between each other over the upstream interfaces of a UFO VLAN, and UFO will forward these packets along.

STP/RSTP/MSTP

Private VLAN UFO can interact with STP/RSTP/MSTP to designate as upstream, the UFO VLAN member that can reach the Root Bridge. The port that can reach the Root Bridge is called the Root Port. This allows arbitrary Layer 2 topologies to be constructed and have all upstream traffic make its way to the STP/RSTP/MSTP Root Bridge and all traffic make its way downstream from the Root Bridge while blocking forwarding otherwise.

EPSR

Private VLAN UFO can interact with EPSR and designate as upstream, the UFO member interface that can reach the EPSR Master, assuming the UFO VLAN is one of EPSR's protected data VLANs. This allows Layer 2 ring topologies to be constructed and have all upstream traffic make its way to the EPSR Master and all traffic make its way downstream from the EPSR Master while blocking forwarding otherwise.

G.8032 (future)

In the future, private VLAN UFO can interact with G.8032 in a proprietary way and designate as upstream, the member interface that can reach a specified G.8032 ring node, assuming the UFO VLAN is one of G.8032's traffic data VLANs. This will allow Layer 2 ring topologies to be constructed and have all upstream traffic make its way to the designated G.8032 node and all traffic make its way downstream from the designated G.8032 node while blocking forwarding otherwise.

Limitations

- Private VLAN UFO is limited to 16 VLANs.
- Layer 3 Switching is not supported on UFO VLANs on 52-port x310 and x510 Series switches, or VCStack configurations.

How to use UFO

Configuring the private VLAN

To configure a private VLAN to operate as UFO, use the **private-vlan** command with the **ufo** parameter:

```
awplus(config-vlan)# private-vlan <vid> {primary |community | isolated |
ufo}
```

Once the VLAN is configured to operate as UFO, its member ports will default to downstream. If the VLAN has already been configured as another type of private VLAN, this command will be rejected.

Configuring upstream interfaces

To configure the UFO VLAN member ports for a role other than downstream, use the following command:

```
awplus(config-if)# switchport mode private-vlan ufo <vid-list> {primary-
upstream | secondary-upstream | stp | epsr | g8032 | ucp }
```

where:

Table 4: Parameter descriptions for the **switchport mode private-vlan ufo** command

PARAMETER	DESCRIPTION
<vid-list>	A comma-separated list, or a range of VLANs, all of which have previously been configured with the **private-vlan** setting of **ufo**.
primary-upstream	The primary interface port/LAG that the UFO VLAN will attempt to use as the upstream interface. More than one primary-upstream interface can be configured.
secondary-upstream	The interface that the UFO VLAN will revert to use as the upstream port if all the primary-upstream ports are unavailable due to being either operationally down (i.e. Link Down) or blocked from forwarding (e.g. STP). More than one secondary-upstream interface can be configured.
stp	STP will determine if the interface plays the role of an STP Root Port for the CIST, or for the MSTI that this UFO VLAN belongs to. If it is, then UFO will designate the interface as an upstream interface member for the UFO VLAN.
epsr	If the UFO VLAN is a protected data VLAN in an EPSR instance, and the interface is one of the transit node's interfaces, then the interface will determine if an EPSR Master is reachable from the interface. If it is, then UFO will designate the interface as an upstream interface member for the UFO VLAN. Otherwise it is downstream.
g8032 (Future)	In the future, g8032 can be used to determine whether the interface is upstream or downstream. If the UFO VLAN is one of the traffic data VLANs being protected in a G8032/ERP instance and the interface is one of the East or West interfaces, then the interface will determine if the 'Hello' G8032 node is reachable from the interface. If it is, then UFO will designate the interface as an upstream interface member for the UFO VLAN. Otherwise the interface is downstream.
ucp (Future)	In the future, an Allied Telesis protocol upstream Control Protocol (UCP) will be used to determine whether the interface can reach the upstream UCP beacon generator. If it is, then UFO will designate the interface as an upstream interface member for the UFO VLAN.

To change the role of the member interface back to downstream (or to ensure it remains as downstream), use the following command:

```
awplus(config-if)# no switchport mode private-vlan ufo <vid-list>
```

Configuring downstream interfaces

There is no particular configuration needed to configure a Private-VLAN UFO member interface as downstream. All Private-VLAN UFO members are by default downstream.

Note, it is possible for a UFO VLAN to be in a state where all members are downstream. This leaves the Private VLAN UFO unable to Layer 2 forward traffic. The **show** command will highlight this condition if it occurs. In addition, SNMP traps, and if available, facility alarms will also provide notifications of this condition.

UFO configuration examples

In this example, three VLANs are configured that belong to the same member ports:

- VLAN 10 is an Internet VLAN that needs to operate as UFO.
- VLAN 20 is a Voice Service VLAN that needs to operate as a standard VLAN
- VLAN 30 is an IPTV Service VLAN that needs to operate as UFO.

The two possible upstream ports are: port1.0.1, port1.0.2, the rest are downstream ports.

Example procedure for private-vlan UFO configuration

Step 1: **Create the VLANs**

Configure VLANs 10, 20 and 30.

```
awplus# configure terminal
awplus(config)# vlan database
awplus(config-vlan)# vlan 10,20,30
```

Configure VLAN 10 and VLAN 30 as UFO.

```
awplus(config-vlan)# private-vlan 10 ufo
awplus(config-vlan)# private-vlan 30 ufo
awplus(config-vlan)# exit
```

Step 2: **Configure the VLAN member interfaces**

Configure trunking on the port interfaces.

```
awplus(config)# interface port1.0.1,port1.0.2,port1.0.3,port1.0.4
awplus(config-if)# switchport mode trunk
```

Add VLANs 10, 20, and 30 to the trunk group.

```
awplus(config-if)# switchport trunk allowed vlan add 10,20,30
awplus(config-if)# exit
```

Step 3: **Configure the primary and secondary-upstream interfaces**

Configure port1.0.1 as the primary-upstream interface

```
awplus(config)# interface port1.0.1
awplus(config-if)# switchport mode private-vlan ufo 10, 30 primary-upstream
```

Configure port1.0.1 as the secondary-upstream interface.

```
awplus(config)# interface port1.0.2
awplus(config-if)# switchport mode private-vlan ufo 10,30 secondary-upstream
```

Note that Ports1.0.3 and port1.0.4 are not configured as downstream as they are downstream by default.

Monitoring UFO

Use the following command to show the configuration and status of a private VLAN UFO based VLAN(s) and their member interfaces:

```
show vlan private-vlan ufo {all|<vlan-id>}
```

```
awplus# show vlan private-vlan ufo {all|<vlan-id>}
VLAN   INTERFACES CONFIGURED          TYPE             PROTOCOL MISC.
--------------------------------------------------------------------------------
<vid> <port|lag> <config>           <us |ds>          <proto>  <miscellaneous>
....
<vid> <port|lag> <config>           <us |ds>          <proto>  <miscellaneous>
                                                               <miscellaneous>
```

Where:

Table 5: Parameter descriptions for the **show vlan private-vlan ufo** command

PARAMETER	DESCRIPTION
all | <vlan-id>	‘all’ shows all UFO VLANs. <vlan-id> filters on a specified VLAN.
<vid>	The VLAN ID configured for private-vlan UFO.
<port | lag>	The port and number, or static LAG ‘sa’ and number or dynamic LAG ‘po’ and number.
<config>	This is one of ‘primary-upstream’ or ‘secondary-upstream’ or ‘epsr’ or ‘stp’ or ‘g8032’.
<us | ds>	us (upstream) or ds (downstream) specifies that the UFO VLAN on this interface is designated as upstream or downstream respectively. If all the member ports are designated as downstream, then ‘downstream-fault’ will be shown instead of ‘downstream’. The fault is not displayed if an IP Address is attached to the UFO VLAN.
<proto>	The protocol used to determine whether the port is upstream or downstream. The protocol can be one of the following: ■ **Static** - the upstream or downstream designation is static through configuration and can not change. This is the designation by default. ■ **UFO** - the upstream or downstream designation is being determined by UFO itself. This is mainly when primary-upstream or secondary-upstream have been configured, and UFO dynamically determines based on Link Up/Down events. ■ **EPSR** - the EPSR protocol running on this interface for this UFO VLAN is dynamically determining whether the EPSR Master is reachable from this interface and if it is, then the interface is designated as upstream, otherwise its downstream. ■ **STP** - the STP protocol running on this interface for this UFO VLAN is dynamically determining whether this interface is playing the role of an STP Root for the CIST or MSTI that this UFO VLAN belongs to, and if it is, then the interface is designated as upstream, otherwise its downstream. ■ **G8032** - the G.8032 protocol running on this interface for this UFO VLAN is dynamically determining whether the G8032 ‘Hello’ node is reachable from this interface and if it is, then the interface is designated as upstream, otherwise its downstream. ■ **UCP** (FUTURE) - the upstream Control Protocol (UCP) running on this interface for this UFO VLAN is dynamically determining whether the Primary UCP beacon node or its backup Secondary UCP beacon node are reachable from this interface and if it is, then the interface is designated as upstream, otherwise its downstream.

Table 5: Parameter descriptions for the **show vlan private-vlan ufo** command (continued)

PARAMETER	DESCRIPTION
<miscellaneous>	This provides miscellaneous or supplemental information: ■ HHHH.HHHH.HHHH is the MAC address of the EPSR Master, STP Root Bridge, or G.8032 'Hello' node if known, otherwise '-'. ■ (FUTURE) this will also include the MAC addresses of the UCP Primary Node with the MAC address appended with '-P' and UCP Secondary Node appended with '-S', along with an indicator '<' as to which of the two UCP nodes is the 'active' upstream node.

Example:

```
VLAN  INTERFACES CONFIGURED         TYPE             PROTOCOL MISC.
-------------------------------------------------------------------------------
100   port1.0.1  primary-upstream   upstream         UFO      -
100   port1.0.2  secondary-upstream downstream       UFO      -
100   port1.0.3  -                  downstream       Static   -
200   port1.0.4  epsr               downstream       EPSR     -
200   sa1        epsr               upstream         EPSR     0030.846e.bac7
200   port1.0.5  -                  downstream       Static   -
300   port1.0.6  -                  downstream-fault Static   -
300   port1.0.7  -                  downstream-fault Static   -
300   port1.0.8  -                  downstream-fault Static   -
400   Port1.0.9  stp                upstream         STP      0030.846e.9bf4
400   Port1.0.10 stp                downstream       STP      -
400   Port1.0.11                    downstream       Static   -
500   Port1.0.12 ucp                downstream       UCP      -
500   Port1.0.13 ucp                upstream         UCP      00c0.2500.05B1-P<
                                                              00c0.2500.7889-S
500   Port1.0.14 -                  downstream       Static   -
600   Port1.0.12 g8032              upstream         G8032    0029.8432.1098
600   Port1.0.13 g8032              downstream       G8032    -
600   Port1.0.14 -                  downstream       Static   -
```

Important information for SBx8100 (CFC960 only)

For the SwitchBlade x8100 with CFC960 Controller, Private VLAN UFO is limited to a small number of VLANs. This is based on the need to conserve hardware resources and because typical applications often only need a small number of VLANs.

The following limitations apply:

- Private VLAN UFO is limited to 16 VLANs.
- UFO and VLAN translation are both implemented using Egress PCL rules, so only one of these features can be configured on a port to function properly. i.e. If a port is configured as a UFO port, then VLAN translation should not be configured on this port and vice versa.
- RSPAN and Port Mirroring are not supported on a UFO VLAN port.

Egress PCL rules

UFO uses Egress PCL rules as follows:

- 1 default drop rule per downstream port per UFO VLAN. If this downstream port is an aggregator (LAG), then each aggregator member port will be installed with one default rule.
- 1 downstream port per UFO VLAN per upstream port/LAG. If the upstream port is an aggregator (LAG), only one PERMIT rule that matches on this aggregator will be installed on each downstream port.
- The CPU port is added as one default upstream port per UFO VLAN. Each downstream port/LAG has a default permit rule that allows traffic coming from the CPU.

For example, using the configuration shown in the table below:

This Egress PCL Rule	Applied to
Upstream ports: sa1	port1.1.1-1.1.3 (3 member ports) and CPU port (default upstream port)
Downstream ports: sa2	port1.2.1-1.2.3 (3 member ports)
Trunk VLAN group	vlan10, vlan20

The following is true:

- One PERMIT rule that allows packets from sa1 will be installed on each downstream aggregator member port. The number of PERMIT rules = 2 (sa1+ ‘CPU port’) x 3 (port1.2.1-1.2.3) = 6
- One DENY rule that drops all packets is installed on each downstream aggregator member port. The number of DENY rules= 3 (port1.2.1-1.2.3)
- The number of PCL rules on each UFO VLAN = 6 + 3 = 9
- Total number of PCL rules on two VLANs = 9 x 2 = 18

Combining private VLANs with other features—limitations

There are some limitations when combining the private VLANs feature with other features:

Using private VLANs with EPSR

You cannot have Ethernet Protected Switching Ring (EPSR) configured on the switch at the same time as private VLANs. Private VLANs allow only one promiscuous port, whereas EPSR topologies require two interfaces to be part of the EPSR loop. However, Private-VLAN UFO supports EPSR.

Using private VLANs with ARP

Devices that are connected to an Isolated VLANs Host Port cannot use Address Resolution Protocol (ARP) to find the switch's IP address. All packets that the Host Port receives are sent directly out via the promiscuous port, without following any of the normal switching operations. This means that ARP requests received on a Host Port are sent to the promiscuous port instead of the CPU port, and therefore the CPU does not process them.

For this reason, you cannot ping a switch through a Host Port or use Simple Network Management Protocol (SNMP) or Telnet to manage the switch through a host port, unless static ARPs are added to the devices at both ends of the host port's link.

This functionality protects your network. Devices connected to Isolated VLANs' host ports are typically not trusted, and this prevents them from having management access to the switch.

This functionality also means that, without static ARPs, you can't process Layer 3 switching through a host port, since ARPs are required to resolve the next hops. Receiving packets on a host port and Layer 3 switching them to a non-private port on another VLAN is a solution, however, traffic coming the other way won't process, as ARPs cannot be dynamically learned on the host port.

Using private VLANs with LLDP

Link Layer Discovery Protocol (LLDP) does not work on trunked interfaces. Any private VLAN solution trunked interfaces cannot also have LLDP configured.

Using private VLANs with GVRP

Private VLAN trunk ports are not supported for GVRP. Private VLAN trunk ports and GVRP are mutually exclusive.

Using private VLANs with link aggregation

Once you have created a static channel or LACP, configure this aggregate interface as the promiscuous port, or even as a secondary host interface. Note that the promiscuous/secondary port status cannot be applied to the individual ports within the aggregation, it can only be applied to the aggregation itself. However, link aggregators can be upstream or downstream interfaces for UFO VLANs.

Using private VLANs with management utility servers

SNMP, sFlow, DHCP, SYSLOG and other management servers should not be connected to isolated ports.

Protocol-based VLANS

Up until now, we have been thinking just of port-based VLANs. However, there are other ways of defining VLAN membership. In this section, we will consider another type of VLAN - namely the Protocol-Based VLAN.

With this VLAN classification method, different protocol types are assigned to different VLANs. For example, IP defines one VLAN, IPX defines another VLAN, Netbeui yet another VLAN, etc.

It is possible that a question will come to your mind at this point, like:

"Isn't a VLAN a set of workstations? How does a protocol specify a workstation? Surely a given workstation can send out packets using different protocols (often at the same time), depending on which applications it is running?"

At this point, you may be starting to see that the description of a VLAN as a set of workstations is a bit of a simplification. So, let us look a bit deeper here and get to a better understanding of what VLAN membership means.

In fact, a given workstation can belong to multiple VLANs. It could belong to one protocol-based VLAN when sending IP packets, another protocol-based VLAN when sending IPX packets, and yet another different port-based VLAN when sending some other protocol.

So, certainly, when analyzing the VLAN setup on a network, it is a mistake to ask "*What VLAN does this workstation belong to?*" The more meaningful question to ask is "*If a packet of such-and-such a protocol arrived at port x of the switch, which VLAN would that packet be associated with?*"

It is important to really understand the change of mind-set that has just been introduced here. When initially learning about VLANs, it is usual to think of VLANs as sets of workstations. And, in practice, this is often all that a network administrator wants to achieve.

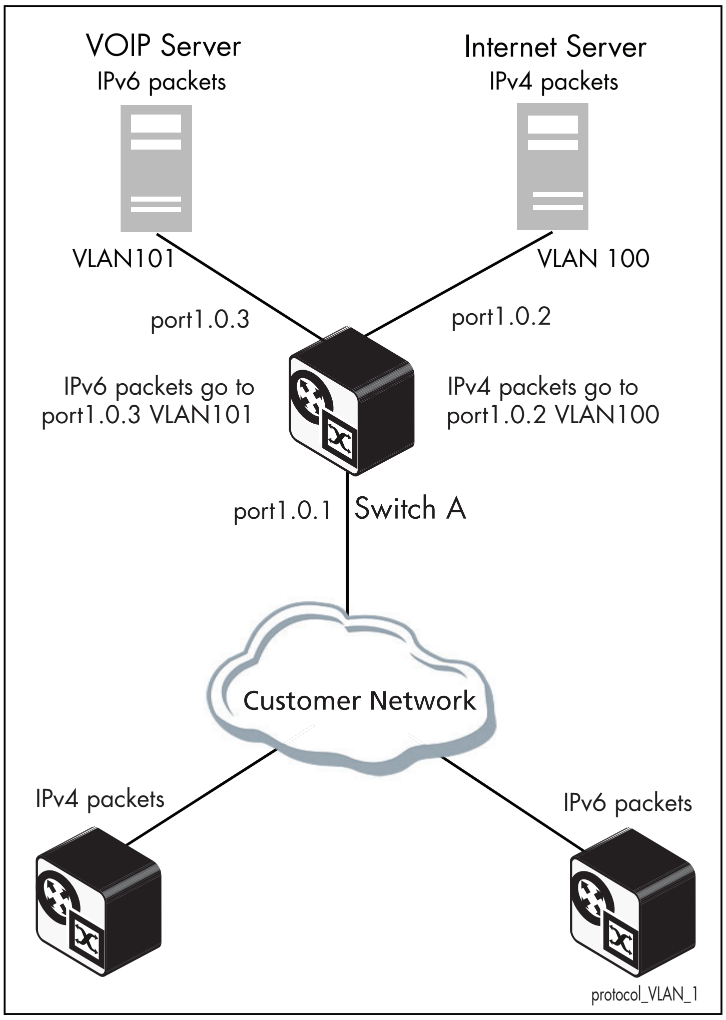
However, once the VLAN configuration on a switch becomes complex, with multiple VLANs of different types all configured on the same port, it is no longer possible to really think about the VLAN from the workstation point of view. It becomes necessary to think of it from the packet point of view.

Therefore, it really is vital to think of packets being associated to VLANs when trying to understand VLAN configurations. Any other approach just ends in confusion. The main point is that, when using protocol-based VLANs, it is data streams that are divided into VLANs, not necessarily whole workstations.

Protocol based VLAN configuration example

To configure **Switch A**, see the procedure following.

Figure 19: Protocol based VLAN configuration



Switch A has the following configuration to enable protocol based VLAN classification:

- VLAN 100 and VLAN 101 created and applied to port1.0.2 and port1.0.3 respectively
- IPv4 and IPv6 VLAN classifier rules created and mapped to VLAN 100 and VLAN 101
- VLAN classifier group created and mapped to port1.0.1
- VLAN 100 and VLAN 101 are trunked over port1.0.2 and port1.0.3 respectively
- IPv4 packets received on port1.0.1 go to port1.0.2 VLAN 100
- IPv6 packets received on port1.0.1 go to port1.0.3 VLAN 101

The configuration procedure below shows the steps to configure **Switch A**.

Configuration procedure for Switch A

```
awplus# configure terminal
awplus(config)# vlan database
awplus(config-vlan)# vlan 100,101
awplus(config-vlan)# exit
```

Step 4: **Create two protocol-type-based VLAN classifier rules for IPv4 and IPv6 mapped to VLAN 100 and 101**

Create a VLAN classifier rule 1 for IPv4 packets on VLAN 100.

```
awplus(config)# vlan classifier rule 1 proto ip encap ethv2 vlan 100
```

Create a VLAN classifier rule 2 for IPv6 packets on VLAN 101.

```
awplus(config)# vlan classifier rule 2 proto ipv6 encap ethv2 vlan 101
```

Step 5: **Create a group of VLAN classifier rules and map the defined VLAN classifier rules 1 and 2 to the group**

Add VLAN classifier rule 1 to VLAN classifier group 1.

```
awplus(config)# vlan classifier group 1 add rule 1
```

Add VLAN classifier rule 2 to VLAN classifier group 1.

```
awplus(config)# vlan classifier group 1 add rule 2
```

Step 6: **Associate the created VLAN classifier group 1 with port1.0.1**

Associate VLAN classifier group 1 with port1.0.1.

```
awplus(config)# interface port1.0.1
awplus(config-if)# vlan classifier activate 1
awplus(config-if)# exit
```

Step 7: Add VLAN 100 to be trunked over port1.0.2

Enable switchport trunking on port1.0.2.

```
awplus(config)# interface port1.0.2
awplus(config-if)# switchport mode trunk
```

Add VLAN 100 to be trunked over port1.0.2.

```
awplus(config-if)# switchport trunk allowed vlan add 100
awplus(config-if)# exit
```

Step 8: Add VLAN 101 to be trunked over port1.0.3

Enable switchport trunking on port1.0.3.

```
awplus(config)# interface port1.0.3
awplus(config-if)# switchport mode trunk
```

Add VLAN 101 to be trunked over port1.0.3.

```
awplus(config-if)# switchport trunk allowed vlan add 101
awplus(config-if)# exit
```

VLAN statistics

This feature provides a series of data counters each able to count both the number of received frames or the number of received bytes (octets) belonging to a particular VLAN. Data frames are counted as they enter the switch ports. By allocating VLANs to each customer, a service provider could use the VLAN counter output to provide the basis for a traffic based billing component.

Commands

Use these commands to create a VLAN packet counter instance named vlan2-data, and apply this to count incoming vlan2 tagged frames on ports 1.0.4 and 1.0.5:

```
awplus(config)# interface port1.0.4,port1.0.5
awplus (config-if)# vlan 2 statistics name vlan2-data
```

To view the counters, use the following command:

```
awplus# show vlan statistics [name <instance-name>]
```

Counter operation

In this section we detail two scenarios: in the first scenario the switch is being used at the edge of the network; in the second scenario it is directly connected to an edge switch. In each situation, separate counters are maintained for incoming traffic that is associated with a particular VLAN across a range of ports. This enables both incoming and outgoing traffic volumes to be measured.

A port may not be assigned to multiple counter instances so as to count frames (or bytes) within the same VLAN. The byte count includes frame headers, therefore the byte counter for a VLAN tagged frame will be 4 bytes longer than for an untagged frame.

Where a VLAN packet counter instance encompasses ports on a stacked member and the member is removed from the stack, these ports will automatically be removed from the counter instance. If this process removes all ports within a counter instance, then the instance will be deleted.

Edge switch scenario

This network is shown in Figure 20 on page 61. The total data count is the upload count plus the download packet count.

Customer A data count

The upload data count for customer A is determined by monitoring the inbound VLAN 10 packets on ports 1.0.2 and 1.0.8 (i.e. packets from Customer A's network). These ports must be untagged members of VLAN 10. Note that packets traveling between these ports will be included in the count.

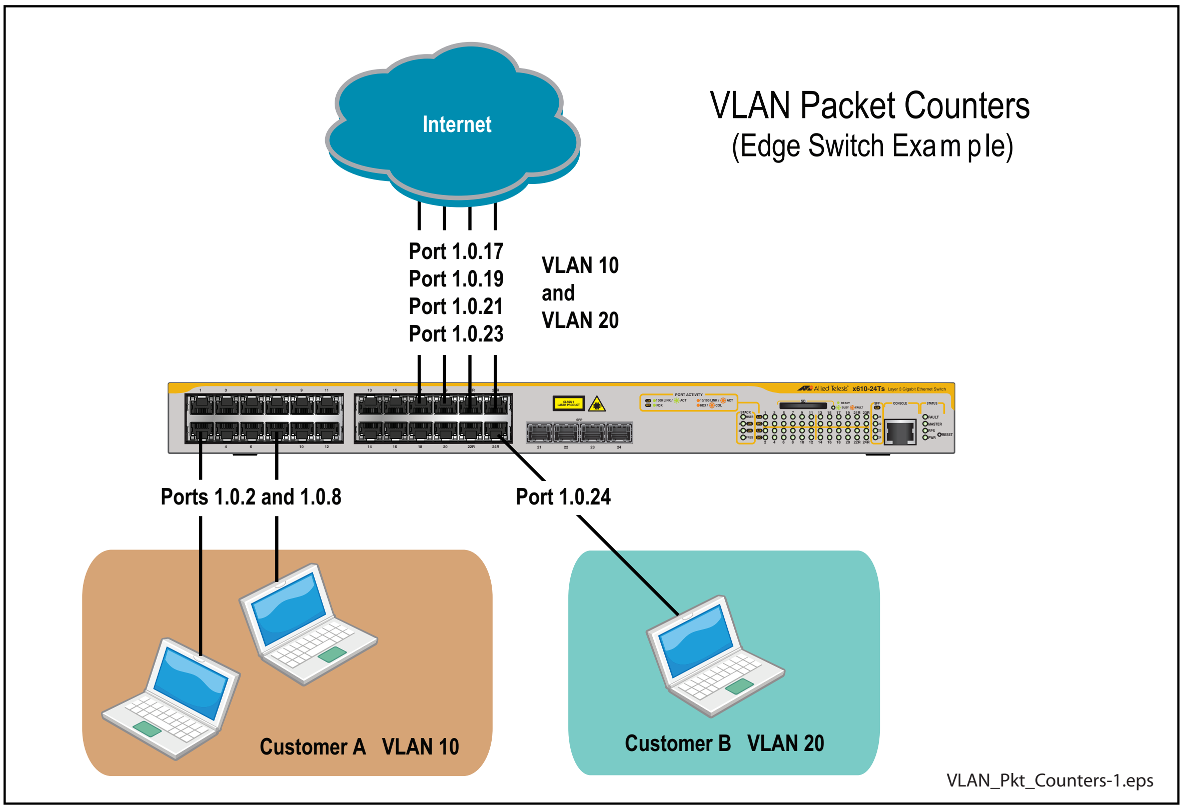
The download data count for customer A is determined by monitoring inbound VLAN 10 packets on ports 1.0.17, 1.0.19, 1.0.21, and 1.0.23 (i.e. packets destined for Customer A's network from the Internet).

Customer B data count

The upload data count for customer B is determined by monitoring the inbound VLAN 20 packets on port1.0.24 (i.e. packets from Customer B's network). This port must be an untagged member of VLAN 20.

The download data count for customer B is determined by monitoring inbound VLAN 20 packets on ports 1.0.17, 1.0.19, 1.0.21, and 1.0.23 (i.e. packets destined for Customer B's network from the Internet).

Figure 20: Packet counters—edge switch scenario



Non-edge switch scenario

This network is shown in Figure 21 on page 62. The total data count is the upload count plus the download packet count.

Customer A data count

The upload data count for customer A is determined by monitoring the inbound VLAN 10 packets on ports 1.0.18 and 1.0.20, 1.0.22, and 1.0.24 on switch Y (i.e. the traffic from customer A's network).

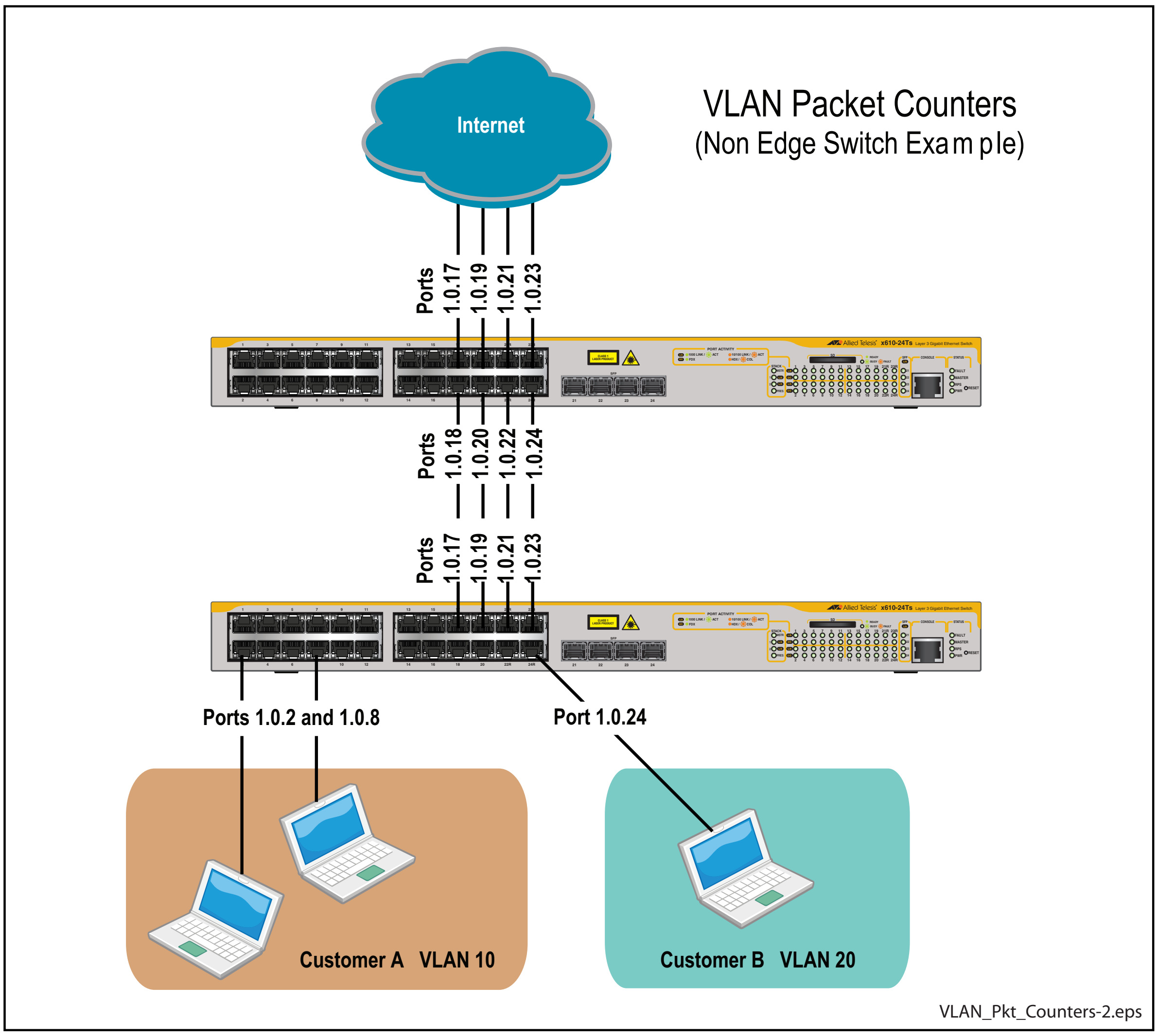
The download data count for customer A is determined by monitoring inbound VLAN 10 packets on ports 1.0.17, 1.0.19, 1.0.21, and 1.0.23.

Customer B data count

The upload data count for customer B is determined by monitoring the inbound VLAN 20 packets on ports 1.0.18 and 1.0.20, 1.0.22, and 1.0.24 on switch Y (i.e. the traffic from customer B's network).

The download data count for customer B is determined by monitoring inbound VLAN 20 packets on ports 1.0.17, 1.0.19, 1.0.21, and 1.0.23.

Figure 21: VLAN packet counters—non-edge switch scenario


Internet
VLAN Packet Counters
(Non Edge Switch Exam ple)
Ports 1.0.17 1.0.19 1.0.21 1.0.23
Ports 1.0.18 1.0.20 1.0.22 1.0.24
Ports 1.0.17 1.0.19 1.0.21 1.0.23
Ports 1.0.2 and 1.0.8
Port 1.0.24
Customer A VLAN 10
Customer B VLAN 20
VLAN_Pkt_Counters-2.eps