Technical Guide

Interfaces

Feature Overview and Configuration Guide

Introduction

This guide describes some specific features that are applied to some interfaces of AlliedWare Plus™ products.

The features are:

- Virtual Tunnel Interfaces (VTI)
- Loopback interfaces
- 802.1Q encapsulation and how it is applied to Ethernet and tunnel interfaces
- Maximum Segment Size (MSS) clamping
- Ethernet management interfaces (NET MGMT ports)
- ByPass ports
- The ifindex numbering scheme
- Converting Stacking ports into network ports

Products and software version that apply to this guide

This guide applies to AlliedWare Plus products running version **5.4.5** or later.

However, feature support and implementation varies between products. 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. Feature support may change in later software versions. For the latest information, see the above documents.

Contents

Switch Ports

The switch ports in the x-series switches support a number of features:

- Enabling and disabling of ports
- Auto negotiation of port speed and duplex mode, where supported by the port type
- Manual setting of port speed and duplex mode, where supported by the port type
- Link up and link down triggers
- Packet storm protection
- Port mirroring

Port numbering

A unique port number identifies each switch port on a device.

Ports are numbered using a 3 digit format **x.y.z** where:

- **x** is the **device number** (1 for a standalone device, or from 1 to 8 for a device in a VCStack™)
- **y** is a **module number** (for devices that have plugin line cards or other modules) or 0 for ports on the base device chassis.
- **z** is the **port number** within the module or on the base device chassis.

In an unstacked (standalone) configuration all device numbers are **1**. For example, port**1.0.37** represents device **1**, port **37** on the device chassis (i.e. not in a plug-in card), and **port1.2.6** represents device **1**, card **2**, port **6**.

In a VCStack, port2.0.8 represents device 2, port 8 on the switch chassis.

Adding a description to an interface

You can add a description to an **interface** to help identify its purpose or position. For example, to add the description “connected to Nerv” to **port1.0.3**, use the commands:

```
awplus(config)#interface port1.0.3
awplus(config-if)#description connected to Nerv
```

Port ranges

Continuous

To configure properties of a continuous range of ports at the same time, enter the range in the format:

```
portx.y.z-portx.y.z
```

For example, to configure the same interface setting on **port1.0.1** to **port1.0.2**, enter the Global Configuration mode command:

```
awplus(config)#interface port1.0.1-port1.0.2
```

Non-continuous

To configure a non-continuous set of ports at the same time, enter a comma-separated list:

```
portx.y.z,portx.y.z
```

For example, to configure the same interface setting on **port1.0.1** and **port1.0.5**, enter the Global Configuration mode command:

```
awplus(config)#interface port1.0.1,port1.0.5
```

You can combine a hyphen-separated range and a comma-separated list. To configure the same setting on **port1.0.1** to **port1.0.3** and **port1.0.5**, enter the Global Configuration mode command:

```
awplus(config)#interface port1.0.1-port1.0.3,port1.0.5
```

Activating and deactivating switch ports

An active switch port is one that is available for packet reception and transmission. By default ports and VLANs are activated.

To deactivate (shutdown) a port or VLAN use the **shutdown** command. Use the **no** variant of this command to reactivate it.

Autonegotiation

Autonegotiation lets the port adjust its speed and duplex mode to accommodate the device connected to it. When the port connects to another autonegotiating device, they negotiate the highest possible speed and duplex mode for both of them.

By default, all ports autonegotiate. Setting the port to a fixed speed and duplex mode may be necessary when connecting to a device that cannot autonegotiate.

We recommend having autonegotiation enabled for link speeds of 1000 Mbps and above. For example, to apply a fixed speed of 1000 Mbps use the command, **speed auto 1000**.

Duplex mode

Ports can operate in full duplex or half duplex mode depending on the type of port it is. When in full duplex mode, a port transmits and receives data simultaneously. When in half duplex mode, the port transmits or receives but not both at the same time.

You can set a port to use either of these options, or allow it to autonegotiate the duplex mode with the device at the other end of the link. To configure the duplex mode, use these commands:

```
awplus#configure terminal
awplus(config)#interface port1.0.1
awplus(config-if)#duplex {auto|full|half}
```

Make sure that the configuration of the switch matches the configuration of the device at the far end of the link. In particular, avoid having one end autonegotiate duplex mode while the other end is fixed. For example, if you set one end of a link to autonegotiate and fix the other end at full duplex, the autonegotiating end cannot determine that the fixed end is full duplex capable. Therefore, the autonegotiating end selects half-duplex operation. This results in a duplex mismatch and packet loss. To avoid this, either fix the mode at both ends, or use autonegotiation at both ends.

Speed options

Before configuring a port's speed, check the hardware limit for the particular port type.

For the latest list of approved SFP transceivers either contact your authorized distributor or reseller, or visit alliedtelesis.com.

You can set a port to use one of multiple speed options, or allow it to autonegotiate the speed with the device at the other end of the link.

We recommend having autonegotiation enabled for link speeds of 1000 Mbps and above.

Configuring the port speed

To set port1.0.1 to auto-negotiate its speed at 1000 Mbps only, which will fix this port speed to 1000 Mbps, enter the following commands:

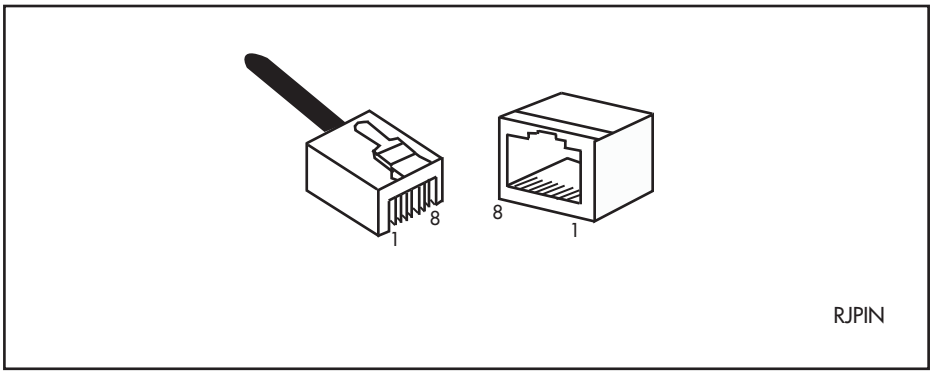
```
awplus#configure terminal
awplus(config)#interface port1.0.1
awplus(config-if)#speed auto 1000
```

MDI/MDIX connection modes

By default, copper 10Base-T, 100Base-T, and 1000Base-T ports on the switch automatically set the Media Dependant Interface mode to MDI or MDIX for successful physical connections. We recommend using this default setting. However, you can configure them to have either fixed MDI mode or fixed MDIX mode by using the polarity command. MDI/MDIX mode polarity does not apply to fiber ports.

Connections to 10BASE-T, 100BASE-T, and 1000BASE-T networks may either be straight though (MDI) or crossover (MDIX). The crossover connection can be achieved by using either a crossover cable or by integrating the crossover function within the device. In the latter situation, the connector is referred to as an MDIX connection. Refer to your switch's Installation Guide for more detailed information on physical connections cabling.

The IEEE 802.3 standard defines a series of Media Dependant Interface types and their physical connections. For twisted pair networking, the standard defines connectors that conform to the IEC 60603-7 standard. The following figure shows a connector of this type:



Virtual Tunnel Interface

The AR-Series firewall products support a number of tunneling protocols that transport data encapsulated within a point-to-point flow of some form.

For ease of configuration, it is useful to treat the tunnel as though it were a physical interface via which the tunnel traffic enters and leaves the device. The entity that embodies this imagined interface is called a Virtual Tunnel Interface (VTI).

A Virtual Tunnel Interface (VTI) is given similar characteristics to any other interface on the device. It is virtual because it does not directly map to any of the physical interfaces on the device. VTIs are presented as Layer 3 interfaces and can have IP configuration applied directly to them and are compatible with Layer 3 routing protocols.

You can use the **interface tunnel*<0-255>*** command to create a virtual tunnel interface.

```
awplus(config)#interface tunnel200
awplus(config-if)#
```

You can specify the type (mode) of tunnel by using the **tunnel mode** command. Tunnel mode types are listed in the following table:

Tunnel mode types

MODE	TRANSPORT	PAYLOAD	DESCRIPTION	RFC
gre	IPv4	IPv6, IPv4	IPv4/IPv6 over IPv4 using Generic Routing Encapsulation	RFC2784
gre ipv6	IPv6	IPv6, IPv4	IPv4/IPv6 over IPv6 using Generic Routing Encapsulation	Not applicable
ipsec ipv4	IPv4	IPv4	Internet Protocol Security in IPv4 tunnel mode	RFC4302, RFC4303
ipsec ipv6	IPv6	IPv6	Internet Protocol Security in IPv6 tunnel mode	RFC4302, RFC4303
l2tp	IPv4	PPP, Eth etc.	Layer 2 protocols over IPv4 using Layer 2 Tunneling Protocol	RFC2661
l2tp ipv6	IPv6	PPP, Eth etc.	Layer 2 protocols over IPv6 using Layer 2 Tunneling Protocol	RFC3931
openvpn tun	IPv4, IPv6	IPv6, IPv4	OpenVPN TUN mode	Not applicable
openvpn tap	IPv4, IPv6	Ethernet frames	OpenVPN TAP mode	Not applicable

For example, you can specify GRE as the tunnel mode for transporting IPv4 or IPv6 packets over IPv4 protocol for **tunnel200** by using the following commands:

```
awplus(config)#interface tunnel200
awplus(config-if)#tunnel mode gre
```

Tunnel endpoints are specified using the **tunnel source** and **tunnel destination** commands. The tunnel remains inactive if no valid tunnel source or tunnel destination is configured. These can be specified using:

- IPv4 address
- IPv6 address
- Interface
- Hostname

```
awplus(config)#interface tunnel200
awplus(config-if)#tunnel source eth1
awplus(config-if)#tunnel destination 192.168.1.2
```

You may find the following Feature Overview and Configuration Guides useful:

- Generic Routing Encapsulation
- IPSec
- OpenVPN

Loopback Interface

A local loopback (lo) interface is one that is always available for higher layer protocols to use and advertise to the network. Although a local loopback interface is assigned an IP address, it does not have the usual requirement of connecting to a lower layer physical entity. This lack of physical attachment creates the perception of a local loopback interface always being accessible via the network.

Local loopback interfaces can be used by a number of protocols for various purposes. Their key benefit is reliability, as they are always available.

One example of this increased reliability is for BGP to use a local loopback interface as a peer address. This peer remains accessible irrespective of the physical links that may be “up” or “down” at the time. This provides a higher probability that the BGP peering will remain active.

You can configure the local loopback interface in Global Configuration mode by using the **interface lo** command as follows:

```
awplus(config)#interface lo
awplus(config-if)#
```

Eth WAN Ports

AR-Series Firewalls contain one or two Ethernet interfaces for connectivity. These ports are found on the front panel and are labeled ETH.

On the AR3050S/AR4050S the ETH interfaces are dual twisted pair/SFP combo ports. The SFP ports can be used instead of the twisted pair ports, but not at the same time.

In configuration commands, the ETH ports are referred to as eth1 and eth2.

802.1Q Encapsulation

802.1Q is the networking standard that defines Virtual LANs (VLANs) on an Ethernet network. This section describes 802.1Q encapsulation on Ethernet and tunnel interfaces and its configuration.

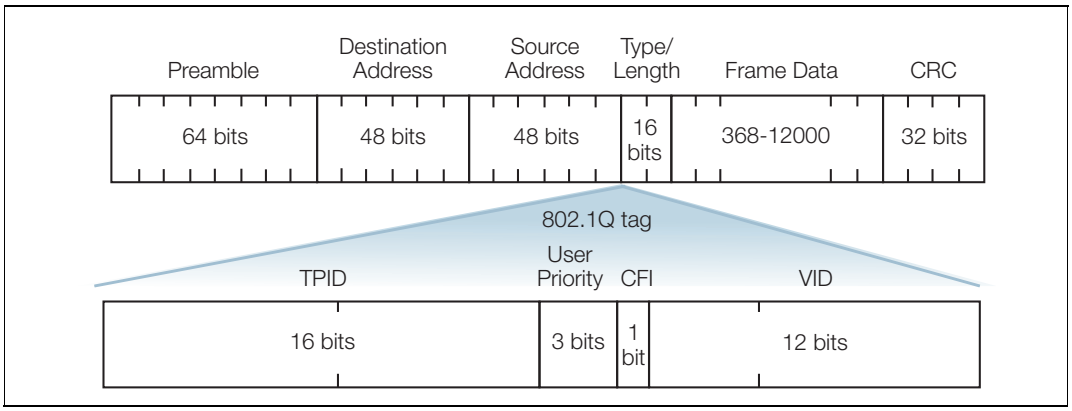
On the AR-Series Firewalls, the 802.1Q standard is implemented in two quite distinct ways.

- On the switching ports, the standard is configured in the same was as it is on switch products. VLANs are created as virtual Layer 3 interfaces, and switch ports are configured with tagged or untagged membership of those VLANs. Network Layer information (IP addresses, routes, and similar), are configured on the VLAN interface.
- On the Ethernet and tunnel interfaces, you create sub-interfaces which represent the logical entities that send and receive packets tagged with various VLAN IDs.

What is 802.1Q encapsulation?

802.1Q is the networking standard that defines virtual LANs (VLANs) on an Ethernet network. VLANs are logical networks that share a single physical connection using 802.1Q tagged frames. An Ethernet frame can contain an 802.1Q tag, with fields that specify VLAN membership and user priority. The VLAN tag is inserted between the source MAC address and the Type/Length fields in the Ethernet frame.

The following figure shows how the VLAN tag is inserted in the frame:



802.1Q-2011 specifies the insertion and removal of the VLAN tag.

Configuring 801.1Q encapsulation on Ethernet and tunnel interfaces

An Ethernet or tunnel interface can participate in multiple VLANs, that is, the interface can send and receive packets tagged with multiple different VLAN IDs. In this way, the interface can operate as the point at which packets are routed into and out of the VLANs it is connected to. Therefore, the interface can have IP interfaces into multiple VLANs. For example, an eth1 interface could have IP address 192.168.78.2 in VLAN 78, and IP address 221.82.128.96 in VLAN 102, and IP address 87.69.52.183 in VLAN 3183 and so on.

To support different VLAN encapsulations on a given interface, the Ethernet interface is divided into logical interfaces—one for each VLAN. These logical interfaces are called sub-interfaces. The sub-interfaces represent the fact that each of these IP addresses is attached to the eth1 interface, but each is associated with a different VLAN encapsulation occurring on eth1. Your router can be configured to perform either:

- Layer 2 forwarding of traffic (bridging) of Ethernet frames between the Ethernet sub-interfaces or to other interfaces.
- Layer3 routing of IP packets between the IP subnets associated with each Ethernet sub-interface or to other interfaces.

The syntax for representing a sub-interface is: ***<interface-name>.<vlan-id>***.

For example, the sub-interface that represents the connection of VLAN 78 to eth1 is written as eth1.78. The relationship of the IP addresses listed above to eth1 is presented as the sub-interface eth1.78 (has IP address 192.168.78.2). The sub-interface eth1.102 (has IP address 221.82.128.96). And the sub-interface eth1.3183 (has IP address 87.69.52.183). Additionally, the parent Ethernet interface could be configured with IP address 192.168.1.1.

802.1Q tags are inserted before transmission out the interface and removed from frames received on the interface. Frames received on the interface are deemed to have arrived on the sub-interface corresponding to the VLAN ID in the frame. Similarly, frames sent out with a certain VLAN tag in place are deemed to be transmitted from the sub-interface associated with the ID in the tag.

Configuration example

```
!
interface eth1
 ip address 192.168.1.1/24
 encapsulation dot1q 78
 encapsulation dot1q 102
 encapsulation dot1q 3183
!
interface eth1.78
 ip address 192.168.78.2/24
!
interface eth1.102
 ip address 221.82.128.96/24
!
interface eth1.3183
 ip address 87.69.52.183/24
!
```

In this configuration example above, the following happens:

- IP packets L3 routed to IP subnet 192.168.78.0/24 via eth1.78 sub interface will be 802.1q vlan tagged with vlanID 78 when egressing the sub-interface.
- IP packets L3 routed to IP subnet 221.82.128.0/24 via eth1.102 sub interface will be 802.1q vlan tagged with vlanID 102 when egressing the sub-interface.
- IP packets L3 routed to IP subnet 87.69.52.0/24 via eth1.3183 sub interface will be 802.1q vlan tagged with vlanID 3183 when egressing the sub-interface.
- IP packets L3 routed to IP subnet 192.168.1.0/24 via parent Ethernet interface have no 802.q tag applied.

Error if you set dot1q encapsulation on L3 tunnels

If you create a tunnel with a mode that only supports L3 traffic, for example (GRE, GRE IPv6, IPSec IPv4 or OpenVPN TUN), and then you try to set dot1q encapsulation on that tunnel, the following error displays:

- `% Failed to create 802.1Q interface, incompatible with an L3 mode`

If you create a tunnel with dot1q encapsulation, and then try to set a mode that only supports L3 traffic, for example (GRE, GRE IPv6, IPSec IPv4 or OpenVPN TUN), the following error displays:

- `% Encapsulation invalid for this mode`

Layer 2 forwarding between different sub-interfaces

Using the same VID on two different interfaces to enable 802.1Q encapsulation does not automatically create a bridge between those interfaces. For example, having eth1.1 and eth2.1 configured on two physical interfaces of the same device will not bridge frames between the interfaces. This is in contrast to the case of a switch, where adding a particular VLAN onto two ports immediately enables Layer 2 forwarding of that VLAN

between those ports. The fact is that different Ethernet and tunnel sub-interfaces are discrete connection points into their VLANs, and the default behavior is not to Layer 2 forward packets between these interfaces.

You must explicitly configure a bridge between the interfaces. In this way, you can pass the frames received on one interface on to the other interface.

Configuration example

This example shows how to configure 802 1.Q encapsulation on Ethernet interfaces. By default, 802.1Q encapsulation is disabled on any Ethernet interfaces or Layer 2 tunnel interfaces, and you need to explicitly enable it.

`awplus#` `configure terminal`	Enter Global Configuration mode.
`awplus(config)#` `interface eth1`	Enter Interface Configuration mode for eth1 interface.
`awplus(config-if)#` `encapsulation dot1q 1`	Configure the interface to perform 802.1Q encapsulation for the VLANs to which it connects. This configuration will create a sub-interface on eth1, called eth1.1.
`awplus(config-if)#` `encapsulation dot1q 2`	This configuration will create a sub-interface on eth1: called eth1.2.
`awplus(config-if)#` `encapsulation dot1q 3`	This configuration will create a sub-interface on eth1: called eth1.3.
`awplus(config)#` `bridge 2`	Enter your bridge ID. Sub-interfaces can be added to bridge instances.
`awplus(config)#` `interface eth1.2`	Enter the bridge group for eth1.2.
`awplus(config-if)#` `bridge-group 2`	Add sub-interface eth1.2 to bridge group 2.
`awplus(config)#` `interface tunnel1.2`	Enter the bridge group for tunnel1.2.
`awplus(config-if)#` `bridge-group 2`	Add sub-interface tunnel1.2 to bridge group 2.
`awplus(config-if)#` `interface eth1.3`	Separately, you can apply Layer 3 information like IP addresses to the sub-interfaces.
`awplus(config-if)#` `ip address 219.72.183.2/24`	The sub-interface can be used in configuration similarly to physical interfaces. For example, IP addresses can be applied to the sub-interface.
`awplus(config-if)#` `exit`	Return to Global Configuration mode.

MSS Clamping

Maximum Segment Size (MSS) clamping functionality allows you to prevent unnecessary fragmentation of packets being forwarded by or transmitted from the AR-Series Firewalls.

Data transmission and MSS clamping

Not all network links are created equal. Some network links can transmit larger packets than others. The largest packet size that can be sent on a given link is referred to as the Maximum Transmission Unit (MTU) for that link. In the path that a packet traverses to get from its source to its destination, it may pass through numerous links which may have differing MTUs.

Various protocols are applied to data when it passes through a network. Each of these protocols adds its own header, which encapsulates the information. This encapsulation increases the size of the packet being transmitted, potentially exceeding the MTU of devices along the path of the data flow. For example, the first link that the packet enters may have an MTU of 1500 bytes, but somewhere along the way, it might have to enter a link that has an MTU of only 1350 bytes. If the packet is sent with a size of 1500 bytes which fits into the initial link along the path, then the packet will have to be fragmented into two pieces when it hits the link that has an MTU of only 1350 bytes.

It is best to avoid fragmentation because it has drawbacks like:

- The receiver has to reassemble the packet, which takes up some computing power on the receiver.
- The extra per-fragment forwarding overhead reduces the throughput of the network.
- If one fragment gets lost, all the fragments have to be resent.
- The interface into the low-MTU link may be one that does not allow fragmentation, and the packet will have to be dropped.

There are techniques for finding the lowest MTU of all the links that a packet might pass through, so that the size of the packets sent will be less than the MTU of the lowest-MTU link along the path.

Path MTU Discovery (PMTUD) is a standardized technique to discover the maximum packet size that can be transmitted across a network. However, PMTUD relies on ICMP which is often blocked for security reasons, which means that PMTUD is not always effective.

An alternative solution is to change the Maximum Segment Size (MSS) of all TCP connections passing through links with MTU lower than the smallest-MTU link along the path. This is known as MSS clamping.

Setting the MSS clamping value at an appropriate limit prevents fragmentation by reserving a set amount of data payload space within a TCP packet.

When the initiation of a TCP session is being negotiated, the MSS is part of the negotiation. One end of a TCP session can say to the other “please don't send segments of more than X Bytes”.

Although the sender of the SYN packet, in the absence of PMTU information, does not actually know the MTU of the smallest-MTU link in the path, all the forwarding devices along the way do know the MTU of the link they are forwarding the SYN onto. If the MSS value in the SYN packet is too large for the link that the SYN is being forwarded onto, the forwarding device can modify the MSS value in the SYN to a value that would enable packets to fit within the link's MTU. If all the devices along the way were set up to do this, then when the SYN arrives at its destination, the MSS value in the SYN would be correct for the smallest-MTU link along the path.

The process of “*forwarding devices altering the MSS value in TCP SYN packets they are forwarding”* is called **MSS clamping**.

The MSS value is a field in the TCP options part of a SYN packet. MSS clamping can be performed on TCP SYN packets that are being forwarded by the device, or on TCP SYN packets that are being originated by the device.

The value that a device puts into the MSS field of an IPv4 SYN packet should be less than the MTU of the link into which the SYN packet is being forwarded. This is because the MSS value is the maximum size for the data segment inside the TCP packet. But the IP and TCP headers need to be inserted on top of the data segment to create the full IP packet.

- For IPv4, the sum of the sizes of the IP and TCP headers is 40 bytes
  For example, if the MTU is 1500 bytes, the MSS for an IPV4 packet should be 1460 bytes.
- For IPv6, the sum of the sizes of the IP and TCP headers is 60 bytes.
  For example, if the MTU is 1500 bytes, the MSS for an IPv6 packet should be 1440 bytes.

The two ends of the TCP session agree on the MSS value. The initiator puts an MSS value in the SYN packet it sends (this MSS value may be changed by devices along the way that are performing MSS clamping) and the other party in the session sends back a SYN-ACK packet which mirrors back the MSS value it found in the SYN packet it received.

Each TCP device uses its MSS value to let other devices know what is the highest allowable amount of data it can receive in a single packet. Although devices in a TCP/IP connection calculate the amount of data to sent in a packet based on variables, such as the current window size and various algorithms, the amount of actual data in a single packet can never exceed the MSS of the device the packet is being sent to.

Once this agreement is achieved, the two ends must adhere to the MSS; TCP will only deliver segments to the IP process that result in packets small enough to traverse the path without fragmentation. UDP does not have a similar mechanism. Instead, what’s usually done is limiting the size of UDP segments in the application itself to a size that is guaranteed to fit any MTU.

MTU and MSS

The Maximum Transmission Unit (MTU) is the maximum number of bytes per packet that may be transmitted by the interface. If a single packet exceeds the MTU, the packet is divided into smaller packets before transmission.

For a TCP packet, the packet size is the header size plus the MSS, where the header size is the size of the packet header and the MSS is the largest amount of TCP data (in bytes) that the device can transmit or receive in one single data packet. To avoid fragmentation, the MSS must be less than the MTU by at least an amount equal to the packet header size.

The MTU size for an interface is set manually with the **mtu** command. See the **mtu** command for details about setting the MTU manually.

If MSS clamping is configured on an interface, then the MSS value in TCP SYN packets egressing the interface will be examined. If the MSS value in a given TCP SYN is too large for the interface, then the MSS is reduced to a suitable value. The definition of “too large” depends on the way in which MSS clamping has been configured on the interface.

- MSS clamping has been configured with a specific MSS value. TCP SYNs egressing the interface have an MSS value greater than the configured MSS value, then their MSS is reduced to the configured value.
- MSS clamping has been configured to calculate the maximum MSS value from the interface's MTU. If the MSS value in an egressing TCP SYN is larger than the value of MTU-***<header size>***, then the packet's MSS is reduced to the value of MTU-***<header size>***.

The MSS value in the TCP SYN is rewritten, and the header checksum is recalculated and rewritten into the packet which is then sent on its way.

Configuring MSS clamping

MSS clamping is configured in Interface Configuration mode on the interface out which the packets will be egressing.

You can configure MSS clamping for IPv4 by using the **ip tcp adjust-mss** command. You can also configure MSS clamping for IPv6 by using the **ipv6 tcp adjust-mss** command.

If a particular value is specified in the command, then the MSS will be set to that value. If the keyword **mtu** is specified, then the MSS will be calculated from the interface MTU, using the following formula:

- for IPv4: MSS = MTU-40
- for IPv6: MSS = MTU-60

In the case of a PPPoE connection, the optimum value for the MSS parameter is 1452 bytes. This value plus the 20-byte IP header, the 20-byte TCP header, and the 8-byte PPPoE header add up to a 1500-byte packet that matches the MTU size for the Ethernet link.

For example, if you are configuring the **mtu** command on the same PPP interface as the **ip tcp adjust-mss** command, it is recommended that you use the following commands and values, as part of the PPPoE interface configuration:

```
awplus#configure terminal
awplus(config)#interface ppp0
awplus(config-if)#ip tcp adjust-mss 1452
awplus(config-if)#mtu 1492
```

NET MGMT Port

Some switches include an eth0 (NET MGMT) management port. The NET MGMT port is an Ethernet-based port and you can use this port to manage the device vis SSH, SNMP, or Telnet. NET MGMT port has an RJ-45 style (8P8C) connector. The following example shows you how to set an IP address on the NET MGMT port:

```
awplus#configure terminal
awplus(config)#interface eth0
awplus(config-if)#ip address 172.28.8.210/16
```

Bypass Port

Some of the AR-Series Firewalls provide bypass ports.The AR3050S and AR4050S firewalls have two bypass ports and the AR2050V has one bypass port. You can use the bypass port to connect the master router to a backup router to maintain the incoming WAN link.

You may connect the ETH port of the master router to the incoming WAN link to the building and connect the bypass port of the master router to the ETH port of an identical backup router. The bypass port of the master router is only active and in the bypass mode if:

- The master router has no power supply or suffers an internal fault, then the incoming WAN link is bypassed to the ETH port of the backup router.
- The master router is active but unable to boot because of some critical issues, then the incoming WAN link is bypassed to the backup router.
- The router is under software control because of some trigger events. For example, if the master router is active and a manual override command is given via the CLI, then the incoming WAN link is bypassed to the backup router.

Note: If the master router is active and is able to boot normally, then the master router connects to the WAN link, the bypass port is inactive and the backup router will not see a link to the WAN on its ETH port.

The ifIndex Numbering Scheme on AlliedWare Plus Devices

The if-MIB is a standard SNMP MIB, implemented by most IP networking devices, that conveys information about devices' interfaces.

A key element of the if-MIB is the ID number that is allocated to each interface on a device. The ID number allocated to a given interface is called its ifIndex. Every networking interface on a device running AlliedWare Plus has an ifIndex associated with it.

This ifIndex is used in a number of contexts:

- it is used to identify the interface in SNMP packets
- it appears in some log messages referring to events that have occurred on the interface
- it is part of the data embedded into Option 82 information in DHCP packets.
- it appears in the output of many **show** commands that are outputting interface-related information

So, it is valuable to have an understanding of the mapping between ifIndex numbers and the interfaces they represent.

The ifIndex values are allocated in ranges, as laid out in the following table:

ifIndex range values, use and interface naming convention

IFINDEX RANGE	USE	INTERFACE NAMING CONVENTION
1	Loopback interface in the Global VRF instance.	lo
2-64	Loopback interfaces in the other (non-Global) VRF instances. The index allocated to the lo interface in the VRF instance with ID VRF_ID is 1 + VRF_ID. So, VRF 1 has loopback interface "lo1" with ifindex of 2, VRF 2 has loopback interface "lo2" with ifindex of 3, etc.	loX
3-300	For products that **do** support VRF, the range starts at 65. For products that don't support VRF, the range starts at 3. This range is allocated to interfaces that are mapped onto interface controllers in the Linux kernel. This includes: Ethernet management interface: eth0 Eth interfaces on AR-Series Firewalls: eth1, eth2 Virtual tunnel interfaces: tunnel1, tunnel2,.... 802.1Q sub-interfaces: eth2.3, tunnel5.20.... The indices are allocated to the interfaces in the order they are created, and so will vary from config to config.	ethX tunnelX ethX.Y tunnelX.Y

ifIndex range values, use and interface naming convention (continued)

IFINDEX RANGE	USE	INTERFACE NAMING CONVENTION
301-4395	VLANS. The numbering scheme is ifIndex = 300 + VLAN_ID	vlanX
4500-4599	Static link aggregations. The numbering scheme is ifIndex = 4500 + static aggregator ID.	saX
4600-4699	Dynamic link aggregations The numbering scheme is 4600 + dynamic aggregator ID.	poX
16778240-16778495	PPP interfaces The numbering scheme is ifIndex = 16778240 + ppp index	pppX
33555968-33556223	Bridge instances The numbering scheme is ifIndex = 33555968+ bridge index	brX
33556224-35556479	3G/4G cellular interfaces The numbering scheme is ifIndex = 33556224 + cellular index	cellularX

Converting Stacking Ports to Network Ports

x610 Series

Reconfiguring x6EM/XS2 stacking module ports as network ports

The Stack module x6EM/XS2 contains two 10 GbE SFP+ ports. By default, these ports are configured for stacking. However, you can reconfigure them as network switch ports. To do this, reconfigure the switch as a non-stacked switch, using the command **no stack *<stack-ID>* enable**, then reboot to apply this configuration.

In the non-stacked mode these ports will appear as configurable switch ports, even without the SFP+ Stack module being inserted within the switch. When configuring these ports, you can identify them by their value 1 in the middle address component of the port number triplet. For example the port number 1.1.2 has the components:
1 (stack member identifier; always a 1 in non-stack mode) **.1** (the board identifier; always a 1 for the Stack module) **.2** (the port number on the Stack module: can have the value 1 or 2).

x930 Series

Reconfiguring StackQS stacking module ports as network ports

On x930 Series switches, you can configure each port on the StackQS card as:

- a stacking port, or
- one 40 Gbps port, or
- four 10 Gbps ports (28-port models only).

The ports are configured as stacking ports by default. When converted to network switch ports, they operate as 40 Gbps ports by default.

To convert the ports to network switch ports, you need to disable VCStack on the ports. There are two options for doing this:

- make the switch into a standalone switch, by running the command:
  no stack *<stack-id>* enable, or
- use the 10 Gbps front-panel SFP+ ports for stacking, by running the command:
  stack enable builtin-ports

Operating the ports as four 10 Gbps ports

To use an StackQS port as four 10 Gbps ports, you need a QSFP-4SFP10G-3CU or QSFP-4SFP10G-5CU breakout cable. Then use the command **platform portmode interface** to change the port or ports to 10 Gbps.

For example, to change both the stacking ports into 10 Gbps ports, use the commands:

```
awplus#configure terminal
awplus(config)#no stack 1 enable
awplus(config)#platform portmode int port1.1.1,port1.1.5 10gx4
```

In 10 Gbps mode, the ports are numbered as follows:

SLOT NUMBER	PORT NUMBER	BECOMES
1	1.1.1	1.1.1, 1.1.2, 1.1.3, 1.1.4
2	1.1.5	1.1.5, 1.1.6, 1.1.7, 1.1.8

When changing the portmode setting, you must also remove any interface and channel-group configuration from the specified ports, and then reboot the switch. Note that the StackQS ports can only operate as four 10 Gbps network switch ports on 28-port switch models, not on 52-port switch models.

DC2552XS/L3 Series

Reconfiguring the front QSFP+ ports as network ports

On DC2552XS/L3 series switches, the front 4 QSFP+ ports can be configured as stacking ports or network ports. By **default**, the ports are configured as **stacking** ports. Note that when stacking is enabled, all four QSFP+ ports become stacking ports, even if you only use one or two ports for stacking.

To convert stacking ports to **network** ports, VCStack needs to be disabled using the following command:

```
no stack <stack-id> enable
```

When operating as network ports the four QSFP+ ports are numbered as follows:

```
Stack 1
Port 1.0.49, 1.0.53, 1.0.57, 1.0.61
```

Operating the QSFP+ ports as four 10 Gbps ports

To use a QSFP+ port as four 10 Gbps ports:

- a QSFP-4SFP10G-3CU or QSFP-4SFP10G-5CU breakout cable is required.
- the QSFP+ port must be configured for 10 Gbps mode using the following command:

```
platform portmode interface <port> 10gx4
```

In 10 Gbps mode, the ports are numbered as follows:

PORT	BECOMES
1.0.49	1.0.49, 1.0.50, 1.0.51, 1.0.52
1.0.53	1.0.53, 1.0.54, 1.0.55, 1.0.56
1.0.57	1.0.57, 1.0.58, 1.0.59, 1.0.60
1.0.61	1.0.61, 1.0.62, 1.0.63, 1.0.64

Show commands

The following show commands display interface configuration and status:

- show interface
- show interface brief
- show interface memory
- show interface status

For detailed show output examples, see your product Command Reference.

Allied Telesis NETWORK SMARTER

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

alliedtelesis.com