

Cisco 6400 NRP Configuration and Troubleshooting

Document ID: 9249

Introduction

Before You Begin

Conventions

Prerequisites

Components Used

RFC1483 Bridging

Broadcasts, Security, and Subscriber Bridging

RFC1483 IP Routing

Configuring PPP

Configuring the Router to Use AAA

Debugging PPPoA

Configuring VPDN

Configuring LAC

Configuring LNS

Debugging LAC

Debugging LNS

Related Information

Introduction

Current digital subscriber line (DSL) deployments consist of RFC1483 bridging or point-to-point routing. After service providers build their expertise, it's likely that higher value-added deployment scenarios will be undertaken. Some possible models include:

- Point-to-point routing and bridging with or without Integrated Routing and Bridging (IRB)
- RFC1483 Bridging with or without IRB
- RFC1483 IP Routing with Subnetwork Access Protocol (SNAP) encapsulation
- Layer 2 Tunneling Protocol (L2TP)
- Interoperability with a Service Selection Gateway (SSG)

The Asynchronous Transfer Mode (ATM) command-line syntax has been significantly enhanced as of Cisco IOS® Software Release 11.3(2)T. Some new features, most notably IETF Point-to-Point Protocol (PPP) over Asynchronous Transfer Mode (ATM) (PPPoA), can only be configured using the new ATM command-line syntax. You can configure the permanent virtual circuit (PVC) directly in the virtual circuit (VC) configuration or using vc-class inheritance.

To use the direct VC configuration method, enter interface or subinterface configuration mode on the ATM interface and type the following commands:

```
router(config-if)#pvc 0/40
router(config-if-atm-vc)#encapsulation aal5mux ppp virtual-Template 1
router(config-if-atm-vc)#ubr 384
router(config-if-atm-vc)#exit
```

These commands configure the following:

- PVC with VPI=0, VCI=40

- Traffic class of unspecified bit rate (UBR)
- Peak bandwidth of 384 Kbps
- Use of virtual-template 1 as the default PPP interface configuration

Note that the **pvc 0/40** command puts you in `config-if-atm-vc` configuration mode. The direct PVC configuration method is useful but forces the operator into repetitive typing if there are many users with similar characteristics. For example, in a large-scale 384 K service deployment, the operator would have to type the same lines once for each user.

An alternate method of VC configuration is to use VC classes. VC classes allow the operator to define a template for a particular VC, apply the class to a subinterface, and have all VCs in the subinterface inherit the configuration of the VC class. To use VC class inheritance, issue the following commands in global configuration mode:

```
router(config)#vc-class atm ppp-atm
router(config-vc-class)#encapsulation aal5mux ppp virtual-Template 1
router(config-vc-class)#ubr 384
router(config-vc-class)#exit
router(config)#interface ATM 4/0
router(config-if)#class ppp-atm
router(config-if)#pvc 0/40
router(config-if)#exit
router(config-if)#pvc 0/41
router(config-if)#exit
```

In the configuration above, all PVCs in the main interface will inherit the properties of VC class `ppp-atm`, unless it is specifically overridden by another VC class in a subinterface or by direct VC configuration. Note that typing the **vc-class** command puts you in `config-vc-class` mode.

By using the PVC Discovery feature, the VC and its traffic parameters may be discovered from the switch, further reducing the router configuration. Note that in the following configuration no VCs or traffic parameters are specified since they are discovered via the Interim Local Management Interface (ILMI) from the switch. It is sufficient to specify the encapsulation type for all VCs discovered from the switch. Per-user configuration is obtained via Remote Dial-In User Service (RADIUS) when the PPP session is established.

```
router(config)#vc-class atm ppp-atm
router(config-vc-class)#encapsulation aal5mux ppp virtual-Template 1
router(config-vc-class)#exit
router(config)#interface ATM 4/0
router(config-if)#class ppp-atm
router(config-if)#atm ilmi-enable
router(config-if)#atm ilmi-pvc-discovery
router(config-if)#exit
```

If you want to separate users on a per-VP basis, you may append the **subinterface** keyword to the ILMI discovery command. For example, a university may partition its constituents into classes *general* class for students and other users, *administration*, and *faculty* and give varying traffic classes and perhaps networks to each.

```
vc-class atm ppp-General
  encapsulation aal5mux ppp virtual-Template 1
  ubr 256
!
vc-class atm ppp-Admin
  encapsulation aal5mux ppp virtual-Template 2
  ubr 512
!
vc-class atm ppp-Faculty
  encapsulation aal5mux ppp virtual-Template 3
```

```

ubr+ 384 512
!
interface ATM 4/0
  atm ilmi-enable
  atm ilmi-pvc-discovery subinterface
  class ppp-atm-General
!
interface ATM 4/0.1 multipoint
  class ppp-atm-Admin
!
interface ATM 4/0.2 multipoint
  class ppp-atm-Faculty
!
interface ATM 4/0.3 multipoint
?
interface Virtual-Template 1
  ip address 10.1.2.3 255.255.255.0
!
interface Virtual-Template 2
  ip address 10.2.3.4 255.255.255.0
!
interface Virtual-Template 3
  ip address 10.3.4.5 255.255.255.0

```

This configuration causes all VCs with non-zero VPI values learned via ILMI to be placed into the subinterface of the corresponding number. For example, if VC 2/123 is reported by the switch, the VC will be placed in subinterface ATM 4/0.2 and will use VC class `ppp-atm-Faculty` and virtual-template 2 as a basic configuration.

For more information please see Enhanced ATM VC Configuration and Management.

Before You Begin

Conventions

For more information on document conventions, see the Cisco Technical Tips Conventions.

Prerequisites

There are no specific prerequisites for this document.

Components Used

This document is not restricted to specific software and hardware versions.

RFC1483 Bridging

RFC1483 bridging is the simplest model to deploy. It is also the most widely adopted because most customer premises equipment (CPE) currently in the market supports only bridging. RFC1483 bridging uses multiprotocol Logical Link Control Subnetwork Access Protocol (LLC-SNAP) encapsulation. To configure RFC1483 bridging, first create a VC with AAL5SNAP encapsulation and then place the subinterface containing the VC in a bridge-group.

```

router(config)#int atm 2/0.2 point-to-point
router(config-if)#pvc 1 32
router(config-if-atm-vc)#encapsulation aal5snap
router(config-if-atm-vc)#protocol bridge broadcast

```

```
router(config-if-atm-vc)#exit
router(config-if)#bridge-group 1
```

Now configure the parameters for bridge-group 1. Typically, a single command is sufficient:

```
router(config)#bridge 1 protocol ieee
```

Bridging remains one of the simplest ways of configuring a Layer 2 multiprotocol VPN. It is useful for nailed-up corporate access where the DSL access concentrator acts as a bridge, forwarding Media Access Control (MAC) frames between the client and the corporate gateway. In this type of configuration, each corporation is given its own bridge group, which prevents traffic from intermingling.

In some instances, usually for general Internet access, bridging may be combined with the Integrated Routing and Bridging (IRB) feature to terminate the bridged traffic and route the traffic to an IP or IPX network. Issuing the **bridge irb** command enables the IRB feature in IOS, but you still need to specify the protocol to be used from the bridged domain to the routed domain.

```
router(config)#bridge irb
router(config)#bridge 1 route ip
```

In the above configuration we have enabled the IRB feature and specified that we want to route IP from bridge-group 1 to the routed domain. When we configure `bridge 1 route ip`, a bridge virtual interface (BVI) is created. The BVI number corresponds to the bridge-group. All other protocols will be bridged. You can also route multiple protocols over a BVI.

```
router(config)#interface bvi 1
router(config-if)#ip address <address>
```

The BVI accepts all commands that you can apply to an interface; for example, **ip helper-address** and **access-groups**.

In the following example there are 2 bridge groups, bridge-group 1 and bridge-group 2. BVI 1 is associated with bridge 1, BVI 2 is associated with bridge 2, and so forth. In the images that have the subscriber bridging feature (Cisco IOS Software Releases 1.3(1)AA, 11.3(1)T, 12.0, 12.0T), 255 bridge groups can be supported.

```
interface atm 2/0
  no ip address
  !
interface atm 2/0.32 point-to-point
  no ip address
  pvc 1/32
    encapsulation aal5snap
  exit
  !
  bridge-group 1
  !
interface atm 2/0.33 point-to-point
  no ip address
  pvc 1/33
    encapsulation aal5snap
  exit
  !
  bridge-group 1
  !
interface atm 2/0.56 point-to-point
  no ip address
  pvc 2/56
    encapsulation aal5snap
  exit
  !
```

```

bridge-group 2
!
interface atm 2/0.57 point-to-point
no ip address
pvc 2/57
encapsulation aal5snap
!
bridge-group 2
!
interface atm 2/0.58 multipoint
no ip address
pvc 2/58
encapsulation aal5snap
exit
!
bridge-group 2
!
interface atm 2/0.58 multipoint
no ip address
pvc 2/59
encapsulation aal5snap
exit
!
bridge-group 2
!
interface BVI1
ip address 10.1.1.1 255.255.255.0 secondary
ip address 10.1.2.1 255.255.255.248 secondary
ip address 10.1.3.1 255.255.255.248 secondary
! you can use secondary addresses... for example some of your customers
! may be having static subnets, and you can describe them using the secondary
! addresses
ip address 10.1.0.1 255.255.255.0
ip helper-address 10.4.5.6
! and for the others shove them off to a DHCP server to pick up
! and address using the ip helper-address command
no ip directed-broadcast
no ip proxy-arp
no ip mroute-cache
arp timeout 3600
!
interface BVI2
ip address 192.168.1.1 255.255.255.0
ipx network F00
!
ip classless
ip route 0.0.0.0 0.0.0.0 10.4.4.1
bridge irb
bridge 1 protocol ieee
bridge 1 route ip
bridge 1 aging-time 3602
bridge 2 protocol ieee
bridge 2 route ip
bridge 2 route ipx
bridge 2 aging-time 3602

```

One important configuration in large IRB networks is the Address Resolution Protocol (ARP) and Bridge timeouts. Always ensure that both the ARP and Bridge entries are aged almost simultaneously or you will see unnecessary flooding of traffic in your links. For example:

ARP entries are maintained for four hours (the default) and bridge MAC entries are kept for ten minutes. If a downstream client is idle for more than ten minutes, the entry will be purged from the bridge table but not from the ARP table. When traffic is sent from an upstream device to a downstream device, we check our ARP tables and find a valid entry pointing to a given MAC address. We then check our bridge tables for this MAC

address but fail to find it, causing us to flood the traffic out of every VC in the bridge–group. This causes a potential security hazard and the transmission of unnecessary amounts of traffic downstream. On the other hand, if both timers expired, we check our ARP table, find no entries, and instead of sending the data traffic down every VC, we send an ARP request packet for the recipient. Upon receipt of the ARP response, we continue data transmission.

Always run either the IEEE or DEC spanning tree protocol on your bridged networks and ensure that all your equipment is running the same spanning–tree protocol. You can choose to disable spanning–tree on a port–by–port basis if you are sure that the remote equipment has no other way into your bridged network. In this case disabling spanning–tree may be good as you can reduce the amount of calculation the router has to do to build a loop–free topology of your network.

Troubleshooting bridging can be a fairly complex task. The best tools are the following **show** and **debug** commands.

```
show spanning-tree X
show bridge X
show arp
debug arp
debug atm packet {interface atm <x/y> {vc / vcd}}
```

Use the **show spanning–tree** command to see the port states. We want to see that a port is in forwarding state, which indicates that it is forwarding data out of the link. A bridge port normally goes through the following states:

- Listening (when it first comes up)
- Learning
- Forwarding or blocked (blocked by spanning–tree to prevent a loop in the bridged network)

```
DSL_7200#show spanning-tree 1
```

```
Bridge group 1 is executing the IEEE compatible Spanning Tree protocol
Bridge Identifier has priority 32768, address 0000.0ce0.70ec
Configured hello time 10, max age 20, forward delay 15
We are the root of the spanning tree
Topology change flag not set, detected flag not set
Times: hold 1, topology change 35, notification 10
      hello 10, max age 20, forward delay 15
Timers: hello 9, topology change 0, notification 0
bridge aging time 10
```

```
Port 18 (ATM2/0.2 RFC1483) of Bridge group 1 is forwarding
Port path cost 6, Port priority 128
Designated root has priority 32768, address 0000.0ce0.70ec
Designated bridge has priority 32768, address 0000.0ce0.70ec
Designated port is 18, path cost 0
Timers: message age 0, forward delay 0, hold 0
BPDU: sent 6778, received 0
```

```
Port 19 (ATM2/0.3 RFC1483) of Bridge group 1 is forwarding
Port path cost 6, Port priority 128
Designated root has priority 32768, address 0000.0ce0.70ec
Designated bridge has priority 32768, address 0000.0ce0.70ec
Designated port is 19, path cost 0
Timers: message age 0, forward delay 0, hold 0
BPDU: sent 6574, received 0
```

The values of the different timers are also of interest to us. Its important that the bridges in your networks have similar timers. Ensure that all devices are pointing to the same designated root when you are troubleshooting spanning–tree issues. If you have devices pointing to different roots in the network, either

some Bridge Protocol Data Units (BPDUs) are being missed or the device with the incorrect root entry is still discovering the topology of the network. If the device is up with spanning-tree stabilized and still showing incorrect values, use the LAN analyzer.

```
DSL_7200#show bridge
```

```
Total of 300 station blocks, 299 free
Codes: P - permanent, S - self
```

```
Bridge Group 1:
```

Address	Action	Interface	Age	RX count	TX count
0010.073c.e100	forward	ATM2/0.2	0	5	0

Issuing the **show bridge** command is useful for determining if we are seeing packets from the remote device. This lets us know if we at least have connectivity to the host. In a large network, the bridge table is likely to be several hundred to a few thousand entries. In this case the easiest way to find an entry using the **show bridge** command is to do the following:

1. Issue the **set term len 0** command.
2. Execute the **show bridge** command.
3. Capture the output in a file.
4. Issue the **grep** command or search for the appropriate MAC address.

The **show arp** command is another useful troubleshooting command. It can be used to verify an IP address to MAC address mapping and lets us know that we have some form of IP connectivity to the device (that is, we can understand the ARP request or at least reply). In a large network, you may want to capture the output to a file and issue the **grep** command for the results.

```
DSL_7200#show arp
```

Protocol	Address	Age (min)	Hardware Addr	Type	Interface
Internet	172.16.25.33	23	00e0.f79c.55c1	ARPA	Ethernet3/0
Internet	172.16.25.34	-	0010.79fb.8054	ARPA	Ethernet3/0
Internet	172.16.25.1	-	0000.0c90.4b7c	ARPA	BVI1
Internet	172.16.25.2	117	0010.073c.e100	ARPA	BVI1

If there are no ARP entries, one first step in troubleshooting connectivity is to issue the **enable debug arp** command.

```
DSL_7200#ping 172.16.25.2
```

```
Type escape sequence to abort.
```

```
Sending 5, 100-byte ICMP Echos to 172.16.25.2, timeout is 2 seconds:
```

```
Sep 18 08:00:18.645: IP ARP: creating incomplete entry for IP address: 172.16.25.2
```

```
Sep 18 08:00:18.645: IP ARP: sent req src 172.16.25.1 0000.0c90.4b7c,
dst 172.16.25.2 0000.0000.0000 BVI1.
```

```
Sep 18 08:00:20.645: IP ARP: sent req src 172.16.25.1 0000.0c90.4b7c,
dst 172.16.25.2 0000.0000.0000 BVI1
```

```
Sep 18 08:00:20.769: IP ARP: rcvd rep src 172.16.25.2 0010.073c.e100, dst 172.16.25.1 BVI1
Success rate is 60 percent (3/5), round-trip min/avg/max = 4/5/8 ms
```

We want to see the router send an ARP request from the correct interface. On the remote side we should monitor for the incoming ARP request. What we should expect back is an ARP reply with a MAC address for the IP address for which we sent the ARP request.

If the output from the **debug arp** command is inconclusive, we need to turn on ATM packet debugging to ensure that we are sending the traffic and to see if we get traffic in response.

The format of a bridged (802.3-specific) RFC1483 protocol data unit (PDU) is shown below. Cisco IOS Software does not usually transmit (but is able to receive) the LAN FCS, and certain vendors include the LAN FCS. Cisco Broadband Operating System (CBOS) does transmit the LAN FCS.

```
LLC (bytes 1 - 3) - [0xAA-AA-03]
OUI (bytes 4 - 6) - [0x00-08-C2]
PID (bytes 7 - 8) - [0x00-07] or [0x00-01] if LAN FCS is needed / included
PAD (bytes 9-10) - [0x00 - 00]
Standard MAC header
Payload
LAN FCS (optional)
```

Sample ATM packet debug output is shown below. Note that turning on ATM packet debugging can stop a router so it is best to execute it on a VC by VC basis.

```
DSL_7200#debug atm packet interface atm 2/0.2 vc 1/32
```

```
DSL_7200#ping 172.16.25.2
```

```
Type escape sequence to abort.
```

```
Sending 5, 100-byte ICMP Echos to 172.16.25.2, timeout is 2 seconds:
```

```
!!!!
```

```
Success rate is 100 percent (5/5), round-trip min/avg/max = 8/32/132 ms
```

```
DSL_7200#
```

```
Sep 18 08:03:32.157: ATM2/0.2(O):
```

```
VCD:0x21 VPI:0x1 VCI:0x21 DM:0x100 SAP:AAAA CTL:03 OUI:0080C2 TYPE:0007 Length:0x80
```

```
Sep 18 08:03:32.157: 0000 0010 073C E100 0000 0C90 4B7C 0800 4500 0064 002D 0000 FF01  
3148 AC10
```

```
Sep 18 08:03:32.157: 1901 AC10 1902 0800 7A54 081C 1D28 0000 0000 04E1 D9D0 ABCD ABCD  
ABCD ABCD
```

```
Sep 18 08:03:32.157: ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD  
ABCD ABCD
```

```
Sep 18 08:03:32.157: ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD
```

```
Sep 18 08:03:32.157:
```

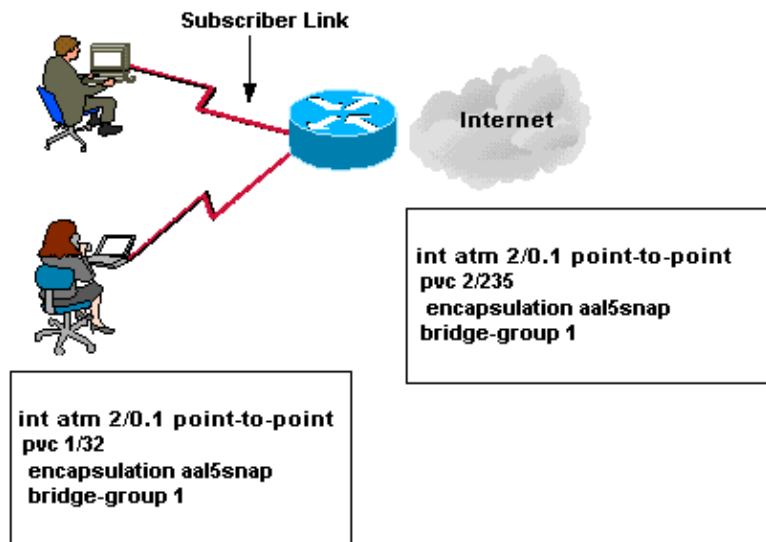
Broadcasts, Security, and Subscriber Bridging

Bridging, a broadcast-based and self-learning protocol, is typically suited only to trusted environments; for example, within a corporation or home. When its deployed in a mass consumer environment such as DSL access, security becomes a concern.

One way to enhance security is to prevent flooding of broadcast, unknown, or ARP packets to each subscriber. In this manner, a user is unable to listen to another user's ARP or unknown packets, nor will one user's broadcasts cause excessive bandwidth usage on another user's ADSL loop. The subscriber bridging feature introduced in Cisco IOS Software Release 11.3(1)T provides these tight policy controls.

The two types of links in subscriber bridging are an implicit *subscriber* link (normally used to describe an ADSL subscriber link) and a *trunk* link (for example, a back-haul link to the Internet). Via the configuration of subscriber policies, we can selectively block ARPs, broadcasts, multicasts, spanning-tree BPDUs, and so forth.

The most restrictive form of subscriber bridging (when all traffic between VCs is denied) is equivalent to placing all VCs on a single multipoint interface. This form of subscriber bridging also minimally uses the IDBs.



A point-to-point interface can have only a single VC; a multipoint interface can have multiple VCs. A general rule of thumb is that a multipoint subinterface models a broadcast LAN, so it is expected that the VCs under a multipoint subinterface are fully meshed. This rule is occasionally broken for simplicity or scalability reasons (for example, in hub-and-spoke IP networks by using the **ip route-cache same-interface** command) but it is important not to break this rule without understanding the implications.

For DSL deployment, if subscriber-to-subscriber communication is unnecessary, it is possible to place all subscriber VCs in a single multipoint subinterface. This type of configuration is possible because of standard bridging rules which specify that data never be forwarded out the same port on which it was received, and that ARP replies should not be sent to the same interface on which the target host (the one queried in the original ARP request) resides. In this configuration, the trunk VC must be separated into another subinterface if IRB is not used.

For more information about Bridging and Subscriber Bridging, please see [Configuring Transparent Bridging and x Digital Subscriber Line Bridge Support](#).

RFC1483 IP Routing

Another possible DSL deployment model is RFC1483 IP routing. CBOS 2.1 will support RFC1483 encapsulation of IP traffic. Both multipoint and point-to-point interfaces are supported with RFC1483 Routing. However, when being used with our CBOS CPE equipment, point-to-point interfaces should be used, as IOS only permits point-to-point interfaces to be unnumbered.

```

int atm 2/0.2 multipoint
ip address 1.1.1.1 255.255.255.0
pvc 1/32
encapsulation aal5snap
protocol ip inarp broadcast
!
pvc 1/33
encapsulation aal5snap
protocol ip inarp broadcast
!
.
.
.
!
int atm 2/0.3 point-to-point
ip address 3.1.1.1 255.255.255.0
pvc 1 32
  
```

```

    encapsulation aal5snap
    protocol ip inarp broadcast
    !
    !
int atm 2/0.4 point-to-point
ip unnumbered ethernet 3/0
pvc 2 32
    encapsulation aal5snap
    protocol ip inarp broadcast
    !
    !
int atm 2/0.5 point-to-point
ip unnumbered ethernet 3/0
pvc 3 32
    encapsulation aal5snap
    !

```

Some of the advantages of using multipoint interfaces include lower IDB and IP address consumption. The disadvantages include routing issues caused by split-horizon and decreased flexibility. Point-to-point interfaces eliminate most routing issues and make configuration changes easy, but the cost is inefficient address usage (unless unnumbered), and high rates of IDB consumption. Since the target audience is primarily residential subscribers, its unlikely that the CPE equipment will need to run a routing protocol. In this scenario, multipoint interfaces might be more efficient for a service provider, but when using CBOS-based CPE equipment, point-to-point interfaces should be used. However, we can regain some level of efficiency by not running routing protocols over the interfaces and not allowing broadcasts.

With multipoint interfaces, you will need to explicitly configure either static maps for the remote addresses on a VC or configure for inverse ARP. Point-to-point interfaces do not need either as there is only one way to send data down a pipe. However, it might be better to configure it for inverse ARP in case the remote changes.

Useful commands when debugging RFC1483 routing include **show atm vc**, **show atm map**, **debug ip packet detail**, **debug atm packet**, and **debug atm error**.

```

DSL7200#show atm vc

```

Interface	VCD / Name	VPI	VCI	Type	Encaps	Peak Kbps	Avg/Min Kbps	Burst Cells	Sts
2/0.3	12	0	41	PVC	CISCOPPP	155000			UP
2/0.1	1	1	32	PVC	MUX	384			UP
2/0.2	10	1	35	PVC	MUX	384			UP
2/0.4	13	1	36	PVC	SNAP	155000			UP
2/0.5	14	1	37	PVC	SNAP	155000			UP

Issuing the **show atm vc** command helps us to verify the interface, VPI/VCI, Virtual-Circuit Descriptor (VCD), and encapsulation used on a particular interface. For an initial check, this is a very helpful command.

```

DSL7200#show atm vc 13
ATM2/0.4: VCD: 13, VPI: 1, VCI: 36
UBR, PeakRate: 155000
AAL5-LLC/SNAP, etype:0x0, Flags: 0xC20, VCmode: 0x0
OAM frequency: 0 second(s)
InARP frequency: 15 minutes(s)
InPkts: 0, OutPkts: 38, InBytes: 0, OutBytes: 3064
InPRoc: 0, OutPRoc: 25
InFast: 0, OutFast: 0, InAS: 0, OutAS: 13
OAM cells received: 0
OAM cells sent: 0
Status: UP

```

We can then look at a particular VC. We can reference it by using its VCD value or by specifying both the interface and the VPI/VCI values. Here we want to verify that the correct encapsulation and Ethertype are

being used.

One indication of traffic on the link is incrementing counters for In and Out. If these counters are not incrementing, you may have a configuration or connection problem. The **show atm map** command is useful only with multipoint interfaces. It lets us know the static and dynamic ATM maps that have been applied on a particular VC. In the example below, the ATM map is a static map that defines IP address 6.6.6.1 to be reachable via 3/32 on interface 2/0.7. It is also permitting broadcasts.

```
DSL7200#show atm map
Map list ATM2/0.7pvc15 : PERMANENT
ip 6.6.6.1 maps to VC 15, VPI 3, VCI 32, ATM2/0.7, broadcast
```

The need for a valid map is very important. If we are not able to map a Layer 3 address (for example an IP address) to a corresponding Layer 2 address, we will mark an encapsulation failure on the packet and drop it. So, on a multipoint interface we need a static map to the next hop or a dynamic map (learned via inverse-arp) to forward packets. By issuing the **show atm map** command, we can avoid issuing the **debug ip packet detail** command except when its needed. Always combine the **debug ip packet detail** command with an access list to prevent stopping the router.

```
DSL7200#show ip access-li 187
Extended IP access list 187
    permit ip 5.5.5.0 0.0.0.255 5.5.5.0 0.0.0.255 (9 matches)

DSL7200#debug ip pack det 187
IP packet debugging is on (detailed) for access list 187

*Oct  4 20:42:39.617: IP: s=5.5.5.5 (local), d=5.5.5.6 (ATM2/0.4), len 100, sending
*Oct  4 20:42:39.617:      ICMP type=8, code=0
```

The configuration above shows an example of how to enable IP packet debugging and how to correctly send a packet. If we did not have a map for the location, the following error would occur:

```
DSL7200#ping 6.6.6.6
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 6.6.6.6, timeout is 2 seconds:
*Oct  4 20:47:57.781: IP: s=6.6.6.5 (local), d=6.6.6.6 (ATM2/0.8), len 100,
sending
*Oct  4 20:47:57.781:      ICMP type=8, code=0
*Oct  4 20:47:57.781: IP: s=6.6.6.5 (local), d=6.6.6.6 (ATM2/0.8), len 100,
encapsulation failed
*Oct  4 20:47:57.785:      ICMP type=8, code=0.
*Oct  4 20:47:59.781: IP: s=6.6.6.5 (local), d=6.6.6.6 (ATM2/0.8), len 100,
sending
*Oct  4 20:47:59.781:      ICMP type=8, code=0
*Oct  4 20:47:59.781: IP: s=6.6.6.5 (local), d=6.6.6.6 (ATM2/0.8), len 100,
encapsulation failed
```

We did not have a valid ATM static or dynamic map to the remote address so we encountered encapsulation failures. This would not occur with point-to-point interfaces as all addresses in the given net are automatically sent down the VC that is applied to the interface (provided the VC is up). We could have also seen the encapsulation errors without resorting to IP packet debugging.

```
DSL7200#debug atm error
ATM errors debugging is on
DSL7200#ping 5.5.5.6
*Oct  4 20:56:03.581: ATM(ATM2/0): Encapsulation error1, link=7, host=5050506
```

If everything looks correct but data is still failing, we can use the **debug atm packet** command to see what is happening.

```

*Oct  4 20:42:39.617: ATM2/0.5(I):
VCD:0xE VPI:0x1 VCI:0x25 Type:0x0 SAP:AAAA CTL:03 OUI:000000 TYPE:0800 Length:0x70
*Oct  4 20:42:39.617: 4500 0064 006E 0000 FF01 A716 0505 0505 0505 0506 0800 8FD9 18F9
1A75 0000
*Oct  4 20:42:39.617: 0000 3236 88CC ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD
ABCD ABCD
*Oct  4 20:42:39.617: ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD
ABCD ABCD
*Oct  4 20:42:39.617: ABCD ABCD ABCD ABCD ABCD

```

The format of a RFC1483 routed IP frame is shown below:

```

LLC (bytes 1 - 3) - [0xAA-AA-03]
OUI (bytes 4 - 6) - [0x00-00-00]
EtherType (bytes 7 - 8) - [0x08-00]
Payload

```

Per the format of the AAL5SNAP routed IP PDU, we can see that the incoming packet shown in the **debug** output is an AAL5SNAP encapsulated IP packet.

For more information about ATM configurations, please see the Wide–Area Networking Configuration Guide.

Configuring PPP

Large–scale deployment of PPP user services requires the use of a central database such as TACACS+ or RADIUS to ease the configuration burden.

The use of RADIUS or TACACS+ servers, collectively known as Authentication, Authorization, and Accounting (AAA) servers, for PPPoA and other media in Cisco IOS Software was introduced with the Virtual Template and Virtual Profile features in Cisco IOS Software Release 11.2(4)F. A virtual template is a basic configuration which is applied to each user as they log in. As each PPP session comes online, a virtual access is *cloned* from the virtual template. This virtual access inherits the full configuration of the virtual template and when the virtual template is changed, the changes are automatically propagated to all virtual accesses cloned from that particular virtual template.

The second step in per–user configuration occurs when the user is identified. At that point user–specific information such as IP address authorization and user–specific route–maps are downloaded from the AAA server and applied to the virtual access belonging to the user.

To configure a virtual template, issue the following commands:

```

router(config)#interface Virtual-Template 1
router(config-if)#ip unnumbered ethernet 1/0
router(config-if)#peer default ip address pool telecommuters
router(config-if)#ppp authentication chap
?
router(config)#ip local pool telecommuters 10.36.1.1 10.36.1.255

```

The configuration above assumes that all PPPoA VCs (users) cloned from virtual template 1 will be using CHAP authentication and will be allocated an IP address from the pool named telecommuters configured on the router. In addition, the local end of the PPPoA connection will be running without an IP address (recommended), and will use the IP address of the e1/0 interface for addressability.

To configure a different class of users on the same router, it is possible to provision a separate virtual template interface. For example, to use a DHCP server rather than a local pool, and to use PAP authentication rather than CHAP, you would configure the following:

```

router(config)#ip dhcp-server <server-name-or-ip>

router(config)#interface Virtual-Template 2
router(config-if)#ip unnumbered ethernet 1/0
router(config-if)#peer default ip address dhcp
router(config-if)#ppp authentication pap

```

Up to 25 virtual templates may be configured.

Configuring the Router to Use AAA

The final step in user configuration is to point the router at an AAA server. The AAA server holds the per-user configuration database, including password authentication and authorization information. To configure the router to use AAA for PPP authentication only, enter the following configuration commands:

```

router(config)#aaa new-model
router(config)#aaa authentication ppp <name> <method>

```

In this example, <name> is the name of the method list (or default, if it is the default list). This is a list of methods, which for our purposes is either RADIUS or TACACS+. For example, to configure virtual template 1 to use TACACS+ before RADIUS, and virtual template 2 to use RADIUS before TACACS+, the following configuration commands would be issued.

```

router(config)#aaa new-model
router(config)#aaa authentication ppp list1 tacacs+ radius
router(config)#aaa authentication ppp list2 radius tacacs+
?
router(config)#interface Virtual-Template 1
router(config-if)#ip unnumbered ethernet 1/0
router(config-if)#ppp authentication chap list1
?
router(config)#interface Virtual-Template 2
router(config-if)#ip unnumbered ethernet 1/0
router(config-if)#ppp authentication chap list2

```

Occasionally, it is useful to keep a local database of usernames and passwords. If a local database is used, the router should be configured to check that local database before checking the AAA servers. Enable the following command:

```

router(config)#aaa authentication local-override

```

For either TACACS+ or RADIUS, the router must be configured with the name or IP address of the server:

```

router(config)#radius-server host <server-name-or-ip> auth-port 1645 acct-port 1646
router(config)#radius-server key <radius-password>

```

The authentication and accounting ports do not have to be specified. The authentication port will default to 1645 and the accounting port will default to 1646. Following is a sample router configuration and RADIUS profile for AAA authentication and network authorization.

```

aaa new-model
aaa authentication login default radius
aaa authentication login no_radius enable

!--- You want to put that on the console/aux port, so that
!--- we use enable passwords for authentication / authorization.

aaa authentication ppp default radius

```

```

aaa authorization network radius
radius-server host 192.168.1.1 auth-port 1645 acct-port 1646

!--- The authentication and accounting ports for the radius daemon.

radius-server timeout 20
radius-server key root

!--- Shared key between NAS and radius server.

#sample radius profile
service-type = framed
framed-protocol = ppp
framed-ip-address = 192.168.54.100
framed-route = 192.168.54.0/24

!--- Sets users IP address to 192.168.54.100 and installs a route
!--- for 192.168.54.0/24 pointing to the remote 192.168.54.100.

```

Debugging PPPoA

One of the common questions asked about the new ATM CLI is, "What happened to the VCD?" The answer is that the VCD still exists but it is hidden from the user. Instead of the VCD, you can refer to a VC either by VCI alone, by VPI/VCI, or by a text-string name. For example:

```

router(config)#int a4/0
router(config-if)#pvc ?
    <0-255>   Enter VPI/VCI value(slash required)
    <1-32767> Enter VCI value
    WORD      Optional handle to refer to this connection

router(config-if)#^Z
router#

```

Note that if a number is typed after the **pvc** command, it is taken to be a VCI instead of a VCD. Therefore, the command **pvc 122** creates a VC with VPI=0, VCI=122, and not a VC with VCD=122. The VCD field has not disappeared; the router still uses the VCD internally:

```

router(config-if)#pvc 0/33
router(config-if)#exit
router(config-if)#pvc foo 0/34
router(config-if)#exit

```

If the command **show atm pvc** is issued, the following output will be generated.

VCD /		Peak Avg/Min Burst							
Interface	Name	VPI	VCI	Type	Encaps	Kbps	Kbps	Cells	Sts
4/0	1	0	33	PVC	MUX	384			UP
4/0	foo	0	34	PVC	MUX	384			UP

Therefore, VC 0/33 is given the VCD 1, but VC 0/34 retains the name "foo." As you will see below, under certain circumstances the VCD is still visible.

For PPPoA the special **show atm pvc ppp** command shows the PPPoA characteristics of a VC.

```

router#show atm pvc ppp
      VCD /
ATM Int.  Name      VPI  VCI  Type  VCSt  VA  VASt  IP Addr
4/0       1         0    33   PVC   UP    1   DOWN  10.123.1.1
4/0       foo         0    34   PVC   UP    2   DOWN  10.123.1.1

```

The "VA" column shows the virtual access used for this particular PPPoA session. A subsequent **show** command on the virtual-access interface will show the PPP specific characteristics of the session:

```
darkshadows#show int virtual-access 2
Virtual-Access2 is up, line protocol is up
  Hardware is Virtual Access interface
  Internet address is 10.123.1.1/24
  MTU 1500 bytes, BW 100000 Kbit, DLY 100000 usec, rely 255/255, load 1/255
  Encapsulation PPP, loopback not set, keepalive not set
  DTR is pulsed for 5 seconds on reset
  LCP Open
  Open: IPCP
  Bound to ATM4/0 VCD: 2, VPI: 0, VCI: 34
  Cloned from virtual-template: 1
  Last input 01:04:26, output never, output hang never
  Last clearing of "show interface" counters 5d02h
  Queueing strategy: fifo
  Output queue 0/40, 0 drops; input queue 0/75, 0 drops
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
    782 packets input, 30414 bytes, 0 no buffer
  Received 3 broadcasts, 0 runts, 0 giants, 0 throttles
  0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
  395 packets output, 5540 bytes, 0 underruns
  0 output errors, 0 collisions, 0 interface resets
  0 output buffer failures, 0 output buffers swapped out
  0 carrier transitions
```

Note the bold text, which shows the Layer 3 protocols enabled on this interface, the interface, the VPI and VCI, and the master virtual template from which this virtual access interface was cloned.

When debugging data traffic, the following **debug** commands may be useful.

```
debug ppp negotiation
debug ppp authentication
debug ppp error
debug ppp packet
```

For communication to occur, negotiation must take place. Peers need a common set of options to reach consensus about how the link should be configured. This includes both LCP layer configurations such as maximum receive unit (MRU) as well as Network Control Protocol (NCP) layer negotiation such as IP address. Authentication refers to the mutual password verification exchanges. Error level debug displays abnormal events, while packet level debug (not recommended) shows all PPP packets received by the router. As with all packet level debugs, issuing the **debug ppp packet** command will show only packets that are not fast-switched.

Following is an example of output from the **debug ppp negotiation** and **debug ppp authentication** commands.

```
*Sep 17 22:06:40: Vi1 LCP: O CONFREQ [ACKsent] id 168 len 15
*Sep 17 22:06:40: Vi1 LCP:   AuthProto CHAP (0x0305C22305)
*Sep 17 22:06:40: Vi1 LCP:   MagicNumber 0x1CB4B6D1 (0x05061CB4B6D1)
```

```
!--- We want to authenticate using CHAP and to send a magic number to
!--- detect a loop in the circuit.
```

```
*Sep 17 22:06:40: Vi1 LCP: I CONFACK [ACKsent] id 168 len 15
*Sep 17 22:06:40: Vi1 LCP:   AuthProto CHAP (0x0305C22305)
*Sep 17 22:06:40: Vi1 LCP:   MagicNumber 0x1CB4B6D1 (0x05061CB4B6D1)
```

*!--- The remote side acknowledges our authentication type and also
!--- sends us its magic number.*

***Sep 17 22:06:40: Vi1 LCP: State is Open**

!--- LCP opens.

*Sep 17 22:06:40: Vi1 PPP: Phase is AUTHENTICATING, by this end
*Sep 17 22:06:40: Vi1 CHAP: O CHALLENGE id 160 len 32 from "darkshadows"
*Sep 17 22:06:40: Vi1 CHAP: I RESPONSE id 160 len 27 from "vtempl"

!--- The remote side is called "darkshadows" and our CHAP name is "vtempl."

*Sep 17 22:06:40: Vi1 PPP: Phase is FORWARDING
*Sep 17 22:06:40: Vi1 PPP: Phase is AUTHENTICATING
*Sep 17 22:06:40: Vi1 CHAP: O SUCCESS id 160 len 4

!--- CHAP authentication passed.

***Sep 17 22:06:40: Vi1 PPP: Phase is UP**

!--- We now attempt IPCP negotiation.

*Sep 17 22:06:40: Vi1 IPCP: O CONFREQ [Not negotiated] id 1 len 10
*Sep 17 22:06:40: Vi1 IPCP: Address 10.1.123.129 (0x03060A017B81)

!--- We want our address to be 10.1.123.129.

*Sep 17 22:06:40: Vi1 IPCP: I CONFREQ [REQsent] id 21 len 10
*Sep 17 22:06:40: Vi1 IPCP: Address 0.0.0.0 (0x030600000000)

!--- The remote side wants an address from us.

*Sep 17 22:06:40: Vi1 IPCP: Using pool 'default'
*Sep 17 22:06:40: Vi1 IPCP: Pool returned 10.1.123.130
*Sep 17 22:06:40: Vi1 IPCP: O CONFNAK [REQsent] id 21 len 10
*Sep 17 22:06:40: Vi1 IPCP: Address 10.1.123.130 (0x03060A017B82)

*!--- We issue an address from our pool. We used the pool called
!--- default to issue an address.*

*Sep 17 22:06:40: Vi1 IPCP: I CONFACK [REQsent] id 1 len 10
*Sep 17 22:06:40: Vi1 IPCP: Address 10.1.123.129 (0x03060A017B81)
*Sep 17 22:06:40: Vi1 IPCP: I CONFREQ [ACKrcvd] id 22 len 10
*Sep 17 22:06:40: Vi1 IPCP: Address 10.1.123.130 (0x03060A017B82)
*Sep 17 22:06:40: Vi1 IPCP: O CONFACK [ACKrcvd] id 22 len 10
*Sep 17 22:06:40: Vi1 IPCP: Address 10.1.123.130 (0x03060A017B82)
*Sep 17 22:06:40: Vi1 IPCP: State is Open
*Sep 17 22:06:40: Vi1 IPCP: Install route to 10.1.123.130
*Sep 17 22:06:41: %LINEPROTO-5-UPDOWN: Line protocol on Interface Virtual-Access1,

changed state to up

```
!--- IPCP was successfully negotiated and we install a host  
!--- route to the remote IP address.
```

One of the new features in Cisco IOS Software Releases 11.3 AA and 12.0 T allows for per-user debugs in PPP. This means that when a particular PPP debug is enabled such as **debug ppp packet**, only the PPP packets coming on the interface belonging to the designated user will be shown. This feature dramatically improves the debugging scalability of PPP.

Per-user debugs are enabled by issuing the **debug condition** command.

```
darkshadows#debug condition ?  
called      called number  
calling     calling  
interface   interface  
username    username
```

Of these four options, **debug condition interface** *<interface #>* and **debug condition username** *<username>* are the most useful for PPPoA.

Multiple debug conditions may be set, in which case the debugs are output if any condition is matched. When a debug condition is set, it is assigned a condition number from 1 to 1000. This condition number may be used to disable the condition using the **no** form of the command. For example:

```
darkshadows#debug condition interface virtual-access 1  
Condition 1 set  
darkshadows#no debug condition  
darkshadows#no debug condition 1
```

This condition is set last. Removing all conditions may cause a flood of debugging messages unless specific debugging flags were previously removed.

```
Proceed with removal? [yes/no]: yes  
Condition 1 has been removed
```

In the previous example, **no debug condition 1** or **no debug condition interface virtual-access 1** are equivalent. To serve as a reminder that the user should first issue a **no debug all** command prior to removing the final debug condition, a warning is printed if the last debug condition is removed. Finally, the shortcut command to removing all debug conditions is **no debug condition all**. Debugs may also be enabled at the ATM layer, though these are generally less useful and should be enabled only if PPP packet debugs yield no information. Perhaps the more useful ATM debug command is the **debug atm packet** command set, which can be enabled globally for all packets or on a per interface/VPI/VCI basis.

```
darkshadows#debug atm packet ?  
interface   ATM Interface  
  
darkshadows#debug atm packet interface  
darkshadows#debug atm packet interface ?  
ATM         ATM interface  
  
darkshadows#debug atm packet interface a4/0 ?  
vc          ATM VPI/VCI Value  
vcd         ATM VCD Number  
  
darkshadows#debug atm packet interface a4/0 vc ?  
<0-255>     VPI/VCI value(slash required)  
<0-65535>   VCI
```

WORD Connection Name

```
darkshadows#debug atm packet interface a4/0 vc 0/33
ATM packets debugging is on
```



Caution: Due to the implementation nature of PPPoA, only outbound PPPoA packets will be displayed

when you issue the debug atm packet command. Inbound packets will never be shown. Useful debugs for AAA include:

- **debug aaa authentication**
- **debug aaa authorization**
- **debug aaa accounting**
- **debug radius**
- **debug tacacs**



Caution: You can also do per-user debugs using the **debug aaa per-user** command.

```
c7200#show debug
General OS:
  AAA Authentication debugging is on
  AAA Authorization debugging is on
PPP:
  PPP authentication debugging is on
  PPP protocol negotiation debugging is on
VTEMPLATE:
  Virtual Template debugging is on
Radius protocol debugging is on
c7200#
*May  8 11:48:03.777: Vi1 LCP: O CONFREQ [Open] id 86 len 14
*May  8 11:48:03.777: Vi1 LCP:   AuthProto PAP (0x0304C023)
*May  8 11:48:03.777: Vi1 LCP:   MagicNumber 0x15ED47E5(0x050615ED47E5)
```

```
!---We send out configuration requests with the options we want,
!--- PAP and MagicNumber, to detect a link loop.
```

```
*May  8 11:48:03.777: Vi1 LCP: O CONFACK [Open] id 1 len 10
*May  8 11:48:03.777: Vi1 LCP:   MagicNumber 0xA60C0000(0x0506A60C0000)
```

```
!--- We acknowledge the remote side's magic number.
```

```
*May  8 11:48:03.785: Vi1 LCP: I CONFACK [ACKsent] id 86 len 14
*May  8 11:48:03.785: Vi1 LCP:   AuthProto PAP (0x0304C023)
*May  8 11:48:03.785: Vi1 LCP:   MagicNumber 0x15ED47E5(0x050615ED47E5)
```

```
!--- The remote side acknowledges our options.
```

```
*May  8 11:48:03.785: Vi1 LCP: State is Open
```

```
!--- LCP now goes to an open state.
```

```
*May  8 11:48:03.785: Vi1 PPP: Phase is AUTHENTICATING, by this end
*May  8 11:48:03.785: Vi1 PAP: I AUTH-REQ id 2 len 19 from "cisco"
*May  8 11:48:03.785: Vi1 PAP: Authenticating peer cisco
```

!--- We attempt PAP authentication. The remote claims to be user "cisco."

```
*May 8 11:48:03.785: AAA/AUTHEN: create_user (0x6104F4A8) user='cisco' ruser=''
port='Virtual-Access1' rem_addr='' authen_type=PAP service=PPP priv=1
*May 8 11:48:03.785: AAA/AUTHEN/START (231528981):port='Virtual-Access1' list=''
action=LOGIN service=PPP
*May 8 11:48:03.785: AAA/AUTHEN/START (231528981): using "default" list
*May 8 11:48:03.785: AAA/AUTHEN/START (231528981): Method=RADIUS
*May 8 11:48:03.785: RADIUS: Initial Transmit Virtual-Access1 id 10 192.168.1.1:1645,
Access-Request, len 75
*May 8 11:48:03.785: Attribute 4 6 C0A80164
*May 8 11:48:03.785: Attribute 5 6 00000001
*May 8 11:48:03.785: Attribute 61 6 00000005
*May 8 11:48:03.785: Attribute 1 7 63697363
*May 8 11:48:03.789: Attribute 2 18 7F5AA6C4
*May 8 11:48:03.789: Attribute 6 6 00000002
*May 8 11:48:03.789: Attribute 7 6 00000001
```

!--- We transmit the user information to the RADIUS server.

```
*May 8 11:48:04.201: RADIUS: Received from id 10 192.168.1.1:1645,Access-Accept, len 55
*May 8 11:48:04.201: Attribute 6 6 00000002
*May 8 11:48:04.201: Attribute 7 6 00000001
*May 8 11:48:04.201: Attribute 8 6 C0A83664
*May 8 11:48:04.201: Attribute 22 17 3139322E
*May 8 11:48:04.205: RADIUS: saved authorization data for user 6104F4A8 at 6105B6EC
*May 8 11:48:04.205: AAA/AUTHEN (231528981): status = PASS
```

!--- We passed authentication and now need to do network authorization.

```
*May 8 11:48:04.205: Vi1 AAA/AUTHOR/LCP: Authorize LCP
*May 8 11:48:04.205: AAA/AUTHOR/LCP: Virtual-Access1: (3215441474):user='cisco'
*May 8 11:48:04.205: AAA/AUTHOR/LCP: Virtual-Access1: (3215441474):send AV service=ppp
*May 8 11:48:04.205: AAA/AUTHOR/LCP: Virtual-Access1: (3215441474):send AV protocol=lcp
*May 8 11:48:04.205: AAA/AUTHOR/LCP: Virtual-Access1: (3215441474): Method=RADIUS
*May 8 11:48:04.205: AAA/AUTHOR (3215441474): Post authorization status= PASS_REPL
```

!--- The user is authorized for PPP services.

```
*May 8 11:48:04.205: Vi1 AAA/AUTHOR/LCP: Processing AV service=ppp
*May 8 11:48:04.205: Vi1 PAP: O AUTH-ACK id 2 len 5
*May 8 11:48:04.205: Vi1 PPP: Phase is UP
*May 8 11:48:04.205: Vi1 AAA/AUTHOR/FSM: (0): Can we start IPCP?
```

!--- Start the authorization process for IPCP.

```
*May 8 11:48:04.205: AAA/AUTHOR/FSM: Virtual-Access1: (270044845):user='cisco'
*May 8 11:48:04.205: AAA/AUTHOR/FSM: Virtual-Access1: (270044845): send AV service=ppp
*May 8 11:48:04.205: AAA/AUTHOR/FSM: Virtual-Access1: (270044845): send AV protocol=ip
*May 8 11:48:04.205: AAA/AUTHOR/FSM: Virtual-Access1: (270044845):Method=RADIUS
*May 8 11:48:04.205: RADIUS: Authorize IP address 192.168.54.100
*May 8 11:48:04.205: RADIUS: route for 192.168.54.0 255.255.255.0
```

!--- We can do IPCP and we are also instructed to install a network

```
!--- route to 192.168.54.0/24.
```

```
*May 8 11:48:04.205: AAA/AUTHOR (270044845): Post authorization status= PASS_REPL
*May 8 11:48:04.205: Vi1 AAA/AUTHOR/FSM: We can start IPCP
*May 8 11:48:04.205: Vi1 IPCP: O CONFREQ [Closed] id 6 len 10
*May 8 11:48:04.205: Vi1 IPCP: Address 192.168.1.100(0x0306C0A80164)
```

```
!--- We want our address to be 192.168.1.100.
```

```
*May 8 11:48:04.209: Vi1 IPCP: I CONFREQ [REQsent] id 3 len 4
*May 8 11:48:04.209: Vi1 AAA/AUTHOR/PCP: Start.
```

```
!--- The address is 0.0.0.0 and we want it to be 192.168.54.100.
```

```
*May 8 11:48:04.209: Vi1 AAA/AUTHOR/PCP: Processing AV service=ppp
*May 8 11:48:04.209: Vi1 AAA/AUTHOR/PCP: Processing AV addr=192.168.54.100
*May 8 11:48:04.209: Vi1 AAA/AUTHOR/PCP: Processing AV route=192.168.54.0 255.255.255.0
*May 8 11:48:04.209: Vi1 AAA/AUTHOR/PCP: Authorization succeeded
*May 8 11:48:04.209: Vi1 AAA/AUTHOR/PCP: Done.
```

```
!--- The address is 0.0.0.0 and we want it to be 192.168.54.100.
```

```
*May 8 11:48:04.209: Vi1 IPCP: O CONFACK [REQsent] id 3 len 4
*May 8 11:48:04.209: Vi1 IPCP: I CONFACK [ACKsent] id 6 len 10
*May 8 11:48:04.209: Vi1 IPCP: Address 192.168.1.100(0x0306C0A80164)
*May 8 11:48:04.209: Vi1 IPCP: State is Open
```

```
!--- IPCP completes negotiation, and we install a host route
!--- to 192.168.54.100.
```

```
*May 8 11:48:04.217: Vi1 IPCP: Install route to 192.168.54.100
```

```
c7200#show ip route
```

```
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
```

```
.
```

```
.
```

```
U - per-user static route, o - ODR
```

```
192.168.54.0/24 is variably subnetted, 2 subnets, 2 masks
```

```
C 192.168.54.100/32 is directly connected, Virtual-Access1
```

```
U 192.168.54.0/24 [1/0] via 192.168.54.100
```

```
C 192.168.1.0/24 is directly connected, Ethernet3/0
```

Configuring VPDN

Virtual Private Dial-up Networking (VPDN) is a powerful way of aggregating many users into a single VC. In DSL deployment, this greatly eases the configuration burden of the ISP when compared to end-to-end PVCs, and speeds the deployment of xDSL services without requiring ubiquitous SVC capability in the WAN.

VPDN uses the Layer 2 Forwarding (L2F) protocol to extend or tunnel a PPP session across the wide area. L2F is a proprietary protocol of Cisco Systems but is widely supported. The next generation protocol is Layer 2 Tunneling Protocol (L2TP), which is a jointly developed IETF standard. At the time of this writing, L2TP is in *Proposed Standard* state, which means that all major hurdles have been identified and solved, and several multivendor interoperability events have been carried out. Cisco has been one of the leaders in developing

L2TP and has been active in the interoperability events.

In the following sections home gateway (HG) is used synonymously with L2TP Network Server (LNS), and network access server (NAS) is used synonymously with L2TP Access Concentrator (LAC). The configuration is specific to L2TP but is readily transferable to L2F.

Configuring LAC

The NAS, or LAC, is the aggregation point of PPP sessions into a L2F or L2TP tunnel. Although both protocols are meant to operate over a variety of media, including IP, ATM, and Frame Relay, currently only L2F and L2TP over IP are supported within Cisco IOS Software.

In the example below, the tunnel configuration is shown in bold type. Each end of the tunnel is assigned a tunnel name, which is used to authenticate the LAC to the LNS and the LNS to the LAC.

As part of the NAS configuration, the tunnel is provisioned to look for the structure ID "dsl.net" in the username. When a user logs in using that structure ID (for example, joe@dsl.net), a tunnel to the IP address 10.1.2.3 is created, if one does not already exist. The tunnel to "dsl.net" will use the LAC router name (dsl-server) when authenticating with the LNS, and will respond to authentication challenges using the password configured in the local database. Note that no users are explicitly identified as users of the L2TP tunnel. The user must log in as belonging to the dsl.net domain. This allows a user to dynamically change destinations by logging into different domains without operator intervention. The configuration below also supports non-tunneled (local) users. If a user logs in without a structure ID (for example, joe) and passes authentication, that user will be allocated an IP address from the *default* IP address pool.

L2TP is transported over IP, and the IP destination 10.1.2.3 for the LNS is reached via an ATM connection, in this case over ATM VC 2/100 under subinterface ATM 4/0.100. Although this example assumes a point-to-point ATM VC between the LAC and LNS, possibly with a WAN ATM network in between, in reality the L2TP tunnel may be routed over an arbitrary IP network such as ATM or Ethernet, or any other media. The other subinterface, ATM 4/0.1, contains two PPPoA PVCs. Both use the default VC class *ppp-atm* configured on the main ATM interface.

Finally, RADIUS is used as the security server (shown in plain italics). Note that RADIUS will only be used for authenticating users who are not tunneled. Tunneled users will be authenticated by the LNS; the LAC plays no part unless configured to do so.

```
!  
version 11.3  
no service password-encryption  
service udp-small-servers  
service tcp-small-servers  
!  
hostname LAC  
!  
aaa new-model  
aaa authentication ppp default radius  
enable secret 5 xxxxxx  
enable password xxxxxx  
!  
username dsl-server password 0 shared-password  
ip multicast rpf-check-interval 0  
ip address-pool local  
vpdn enable  
vpdn-group 1  
  request dialin  
    protocol l2tp  
    domain cisco.com  
    initiate-to ip 10.1.2.3
```

```

local-name dsl-server
no l2tp tunnel authentication
!
!
interface Ethernet1/0
  ip address 10.207.1.2 255.255.0.0
  no ip mroute-cache
  no cdp enable
!
interface Ethernet1/1
  no ip address
  no ip mroute-cache
  shutdown
  no cdp enable
!
interface Ethernet1/2
  no ip address
  no ip mroute-cache
  shutdown
  no cdp enable
!
interface Ethernet1/3
  no ip address
  no ip mroute-cache
  shutdown
  no cdp enable
!
interface ATM4/0
  no ip address
  no ip mroute-cache
  vc-class ppp-atm
!

interface ATM4/0.1 multipoint
  pvc 0/33
    exit
  !
  pvc 0/34
    class ppp-atm
    exit

!
interface ATM4/0.100 point-to-point
  ip unnumbered Ethernet1/0
  pvc 2/100
    protocol ip 10.1.2.3 broadcast
    ubr 3000
    encapsulation aal5snap
    exit
  !
!

interface Virtual-Template1
  ip unnumbered Ethernet 1/0
  no ip mroute-cache
  ppp authentication chap pap
  peer default ip address pool default

!
router igrp 99
  redistribute static
  network 10.0.0.0
!

ip local pool default 10.34.0.1 10.34.0.16

```

```

ip default-gateway 10.1.0.1
no ip classless
ip route 0.0.0.0 0.0.0.0 Ethernet1/0
no cdp run
!

vc-class ppp-atm
ubr 384
encapsulation aal5mux ppp virtual-template 1

!
radius-server host server-name-or-ip auth-port 1645 acct-port 1646
radius-server key radius-password
!
line con 0
line aux 0
line vty 0 4
password xxxxxx
!
end

```

Configuring LNS

LNS configuration is very similar to LAC configuration, with the following differences.

- The virtual templates are used for the PPP sessions coming from the tunnel and not from the PPPoA sessions.
- To request a tunnel for a dialin user, instead of using the **request dialin** command, the **accept dialin** command is used to accept incoming tunnels from dial users.
- The LAC name is used to identify the tunnel for authentication and manipulation purposes.
- Unlike the LAC, which must be configured to use either L2F or L2TP when originating tunnels, the LNS may auto-detect between L2TP or L2F tunnels if configured with **accept dialin** any as opposed to **accept dialin l2tp** or **accept dialin l2f**.

The VPDN command in the LNS specifies that if a tunnel setup request comes in from an LAC named `dsl-server`, the LNS router name "corporate-gateway" should be the name sent back to the LAC. This allows the LAC to authenticate the LNS. When the LNS receives a tunnel setup request, it looks up the name given by the LAC in the local user name database (since no security server has been configured in this example). Similarly, in replying to the LAC setup request, the LNS also passes authentication using the password information for corporate-gateway. This information is then passed to the RADIUS server (used by the LAC) for authentication. Note that the names used when authenticating between the LNS and LAC are the router names by default; however, the default names can be overridden by issuing the **local name** command in `vpdn-group config` mode. Note also that the password is shared and is thus identical between the LAC and LNS.

As in the LAC configuration, it's assumed that the tunnel is carried over an IP over ATM connection. Thus, the other end of the ATM connection is configured in subinterface ATM 0.123. Note that the PVC does not match that of the LAC, since there is an ATM switched network between the LAC and the LNS.

Unlike a true PPP interface, each tunnel can carry multiple PPP sessions. In a manner identical to PPPoA, the PPP user sessions coming in from the LAC are virtual access interfaces cloned from a virtual template.

In the example below, the blue text highlights the user configuration and the bold text shows the LNS configuration. Two users are configured, user1 and user2. Both belong to the dsl.net domain. Since there is only one tunnel configured, it is assumed that both users will come in via the same tunnel. Since the tunnel from the NAS is configured to use virtual-template 1, both users will be allocated an IP address from the *telecommuters* address pool, and both users are expected to authenticate using CHAP (preferred) or PAP.

```

!
version 11.3
no service password-encryption
!
hostname corporate-gateway
!
enable secret 5 xxxxxx
enable password xxxxxx
!

username user1@dsl.net password 0 user1-password
username user2@dsl.net password 0 user2-password

username corporate-gateway password 0 shared-password
ip domain-name cisco.com
ip name-server 171.69.2.132
ip name-server 198.92.30.32
ip multicast rpf-check-interval 0
vpdn enable
vpdn-group 1
  accept dialin
  protocol l2tp
  virtual-template 1
  terminate-from remote dsl-server
  no l2tp tunnel authentication
!
!
interface Ethernet0
ip address 10.2.3.26 255.255.0.0
no ip mroute-cache
media-type 10BaseT
no cdp enable
!

interface Virtual-Template1
ip unnumbered Ethernet0
no ip route-cache
no ip mroute-cache
peer default ip address pool telecommuters
ppp authentication chap pap

!
interface ATM0
no ip address
!
interface ATM0.123
ip address 10.1.2.3 255.255.255.0
pvc 0/144
  protocol ip 10.207.1.2 broadcast
  exit
!
!
router igrp 99
redistribute connected
network 10.0.0.0
!

ip local pool telecommuters 10.36.1.1 10.36.1.255

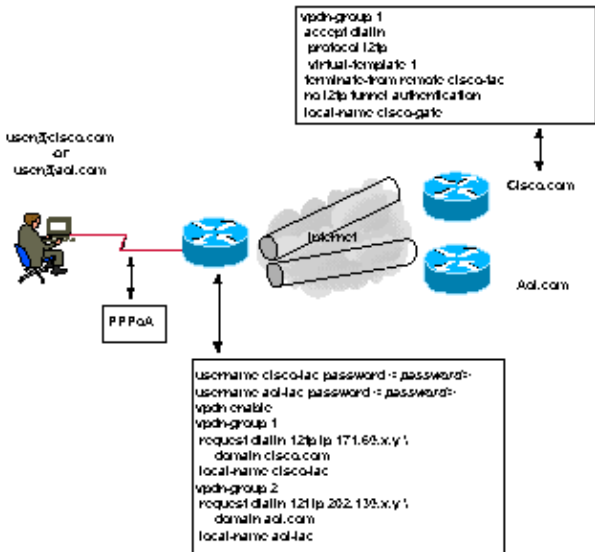
no ip classless
ip route 0.0.0.0 0.0.0.0 Ethernet0
no cdp run
!
!
line con 0
password xxxxxx

```

```

line aux 0
line vty 0 4
  password xxxxxx
  login
!
end

```



Debugging LAC

```

*Sep 16 17:00:43: %LINK-3-UPDOWN: Interface Virtual-Access1, changed state to up
*Sep 16 17:00:43: Tnl/Cl 102/1 L2TP: Session FS enabled
*Sep 16 17:00:43: Tnl/Cl 102/1 L2TP: Session state change from idle to wait-for-tunnel
*Sep 16 17:00:43: Vi1 102/1 L2TP: Create session
*Sep 16 17:00:43: Tnl 102 L2TP: SM State idle
*Sep 16 17:00:43: Tnl 102 L2TP: Create SCCRQ
*Sep 16 17:00:43: Tnl 102 L2TP: Tunnel state change from idle to wait-ctl-reply
*Sep 16 17:00:43: Tnl 102 L2TP: SM State wait-ctl-reply
*Sep 16 17:00:44: %LINEPROTO-5-UPDOWN: Line protocol on Interface Virtual-Access1,
changed state to up
*Sep 16 17:00:45: Tnl 102 L2TP: Process SCCRQ from knob-hill
*Sep 16 17:00:45: Tnl 102 L2TP: Tunnel state change from wait-ctl-reply to established
*Sep 16 17:00:45: Tnl 102 L2TP: Got a challenge from remote peer, knob-hill
*Sep 16 17:00:45: %VPDN-6-AUTHENERR: L2F NAS darkshadows cannot locate a AAA server for
tunnel darkshadows
*Sep 16 17:00:45: Tnl 102 L2TP: Got a response from remote peer, knob-hill
*Sep 16 17:00:45: %VPDN-6-AUTHENERR: L2F NAS darkshadows cannot locate a AAA server for
tunnel darkshadows
*Sep 16 17:00:45: Tnl 102 L2TP: Tunnel Authentication success
*Sep 16 17:00:45: Tnl 102 L2TP: Create SCCCN to knob-hill tnlid 16

```

Start Control Connection Connected (SCCCN) indicates that a tunnel is established. Incoming Call Connected (ICCN) indicates that the PPP session within the tunnel is established. Each additional session in the tunnel gets only the ICCN. The LAC side is similar.

```

*Sep 16 17:00:45: Tnl 102 L2TP: SM State established
*Sep 16 17:00:45: Vi1 102/1 L2TP: Create ICRQ to knob-hill tnlid/clid 16/0
*Sep 16 17:00:45: Vi1 102/1 L2TP: Session state change from wait-for-tunnel to wait-reply
*Sep 16 17:00:45: Vi1 102/1 L2TP: Create ICCN to knob-hill tnlid/clid 16/1
*Sep 16 17:00:45: Vi1 102/1 L2TP: Session state change from wait-reply to established

```

Debugging LNS

```
*Jul 26 03:55:33: L2TP: Process SCCRQ from darkshadows tnlid 102
*Jul 26 03:55:33: Tnl 16 L2TP: New tunnel created for remote darkshadows,
address 10.1.121.1
*Jul 26 03:55:33: Tnl 16 L2TP: Got a challenge in SCCRQ, darkshadows
*Jul 26 03:55:33: %VPDN-6-AUTHENERR: L2F HGW cannot locate a AAA server for
tunnel knob-hill
*Jul 26 03:55:33: Tnl 16 L2TP: Create SCCRP to darkshadows tnlid 102
*Jul 26 03:55:33: Tnl 16 L2TP: Tunnel state change from idle to wait-ctl-reply
*Jul 26 03:55:33: Tnl 16 L2TP: Process SCCCN from darkshadows tnlid 102
*Jul 26 03:55:33: Tnl 16 L2TP: Tunnel state change from wait-ctl-reply to established
*Jul 26 03:55:33: Tnl 16 L2TP: Got a Challenge Response in SCCCN from darkshadows
*Jul 26 03:55:33: %VPDN-6-AUTHENERR: L2F HGW cannot locate a AAA server for
tunnel knob-hill
*Jul 26 03:55:33: Tnl 16 L2TP: Tunnel Authentication success
*Jul 26 03:55:33: Tnl 16 L2TP: SM State established
*Jul 26 03:55:33: Tnl 16 L2TP: Process ICRQ from darkshadows tnlid 102
*Jul 26 03:55:33: Tnl/Cl 16/1 L2TP: Session FS enabled
*Jul 26 03:55:33: Tnl/Cl 16/1 L2TP: Session state change from idle to wait-for-tunnel
*Jul 26 03:55:33: Tnl/Cl 16/1 L2TP: New session created
*Jul 26 03:55:33: Tnl/Cl 16/1 L2TP: Create ICRP to darkshadows tnlid/clid 102/1
*Jul 26 03:55:33: Tnl/Cl 16/1 L2TP: Session state change from wait-for-tunnel
to wait-connect
*Jul 26 03:55:33: Tnl/Cl 16/1 L2TP: Process ICCN from darkshadows tnlid/clid 102/1
*Jul 26 03:55:33: Vi1 PPP: Phase is DOWN, Setup
*Jul 26 03:55:33: %LINK-3-UPDOWN: Interface Virtual-Access1, changed state to up
*Jul 26 03:55:33: Vi1 PPP: Treating connection as a dedicated line
*Jul 26 03:55:33: Vi1 PPP: Phase is ESTABLISHING, Active Open
*Jul 26 03:55:33: Vi1 LCP: O CONFREQ [Closed] id 1 len 10
*Jul 26 03:55:33: Vi1 LCP: MagicNumber 0x27B4B354 (0x050627B4B354)
*Jul 26 03:55:33: Vi1 PPP: Treating connection as a dedicated line
*Jul 26 03:55:33: Vi1 LCP: I FORCED CONFREQ len 11
*Jul 26 03:55:33: Vi1 LCP: AuthProto CHAP (0x0305C22305)
*Jul 26 03:55:33: Vi1 LCP: MagicNumber 0x16764D6D (0x050616764D6D)
```

FORCED indicates that the link control protocol (LCP) isn't negotiated. This occurs between the LAC and the client; the LNS receives only an indication of what is negotiated. From this point on, PPP NCP negotiation occurs as normal. The LAC side will show LCP negotiation prior to tunnel and session negotiation.

```
*Jul 26 03:55:33: Vi1 PPP: Phase is AUTHENTICATING, by this end
*Jul 26 03:55:33: Vi1 CHAP: O CHALLENGE id 102 len 30 from "knob-hill"
*Jul 26 03:55:33: Vi1 L2X: Discarding packet because of no mid/session
*Jul 26 03:55:33: Vi1 CHAP: I RESPONSE id 102 len 35 from "vtempl@dsl.net"
*Jul 26 03:55:33: Vi1 CHAP: O SUCCESS id 102 len 4
*Jul 26 03:55:33: Vi1 PPP: Phase is UP
*Jul 26 03:55:33: Vi1 IPCP: O CONFREQ [Closed] id 1 len 10
*Jul 26 03:55:33: Vi1 IPCP: Address 10.1.121.2 (0x03060A017902)
*Jul 26 03:55:33: Vi1 IPCP: I CONFREQ [REQsent] id 12 len 10
*Jul 26 03:55:33: Vi1 IPCP: Address 0.0.0.0 (0x030600000000)
*Jul 26 03:55:33: Vi1 IPCP: Using pool 'default'
*Jul 26 03:55:33: Vi1 IPCP: Pool returned 10.1.131.1
*Jul 26 03:55:33: Vi1 IPCP: O CONFNAK [REQsent] id 12 len 10
*Jul 26 03:55:33: Vi1 IPCP: Address 10.1.131.1 (0x03060A018301)
*Jul 26 03:55:33: Vi1 IPCP: I CONFACK [REQsent] id 1 len 10
*Jul 26 03:55:33: Vi1 IPCP: Address 10.1.121.2 (0x03060A017902)
*Jul 26 03:55:33: Vi1 IPCP: I CONFREQ [ACKrcvd] id 13 len 10
*Jul 26 03:55:33: Vi1 IPCP: Address 10.1.131.1 (0x03060A018301)
*Jul 26 03:55:33: Vi1 IPCP: O CONFACK [ACKrcvd] id 13 len 10
*Jul 26 03:55:33: Vi1 IPCP: Address 10.1.131.1 (0x03060A018301)
*Jul 26 03:55:33: Vi1 IPCP: State is Open
*Jul 26 03:55:33: Vi1 IPCP: Install route to 10.1.131.1
*Jul 26 03:55:34: %LINEPROTO-5-UPDOWN: Line protocol on Interface Virtual-Access1,
changed state to up
```

To learn more about configuring VPDN networks, please see [Virtual Private Dialup Networks](#).

Related Information

- [Cisco DSL Product Support Information](#)
 - [Technical Support – Cisco Systems](#)
-

[Contacts & Feedback](#) | [Help](#) | [Site Map](#)

© 2007 – 2008 Cisco Systems, Inc. All rights reserved. [Terms & Conditions](#) | [Privacy Statement](#) | [Cookie Policy](#) | [Trademarks of Cisco Systems, Inc.](#)

Updated: Feb 26, 2008

Document ID: 9249
