How to configure PPPoE?

PPPoE provides an emulated (and optionally authenticated) point-to-point link across a shared medium, typically a broadband aggregation network such as those found in DSL service providers. In fact, a very common scenario is to run a PPPoE client on the customer side (commonly on a SOHO Linksys or similar brand router), which connects to and obtains its configuration from the PPPoE server (head-end router) at the ISP side. Note that ATM is typically run between the customer's modem and the DSLAM, though it will be transparent in this lab since our PPPoE client exists on a separate device.
Server Configuration
The first task at the ISP end is to configure a Broadband Aggregation (BBA) group which will handle incoming PPPoE connection attempts. We'll name this **MyGroup**, and bind it to a virtual template to be created shortly.
ISP(config)# **bba-group pppoe MyGroup**
ISP(config-bba-group)# **virtual-template 1**

Here we can also apply PPPoE session limits. For example, we can limit the number of sessions established per client MAC address (setting this limit to 2 allows a new session to be established immediately if the prior session was orphaned and is waiting to expire). This is an optional step.
ISP(config-bba-group)# **sessions per-mac limit 2**

Next we'll create the virtual template for the customer-facing interface. When a PPPoE client initiates a session with this router, the router automatically spawns a virtual interface to represent that point-to-point connection.
ISP(config)# **interface virtual-template 1**

At a minimum, we'll need to configure two items on our virtual template: an IP address, and a pool of IP addresses from which clients are assigned a negotiated address (similar in operation to DHCP).
ISP(config-if)# **ip address 10.0.0.1 255.255.255.0**
ISP(config-if)# **peer default ip address pool MyPool**

You may be wondering where the IP pool is defined. Well, it isn't; that's what we have to do next. Back in global configuration mode, we define a local IP pool named **MyPool** with the starting and ending addresses of an IP range. If you've configured DHCP on IOS before, you should find this task very familiar.
ISP(config)# **ip local pool MyPool 10.0.0.2 10.0.0.254**

Last, we need to enable our PPPoE group on the interface facing the customer network.
ISP(config)# **interface f0/0**
ISP(config-if)# **no ip address**
ISP(config-if)# **pppoe enable group MyGroup**
ISP(config-if)# **no shutdown**

Note that this interface should not have an IP address; the addressing is provided by our virtual template.
Client Configuration
Client configuration is relatively simple. We create a dialer interface to handle the PPPoE connection, and tie it to a physical interface which provides the transport.

Creating our PPPoE dialer interface:
CPE(config)# **interface dialer1**
CPE(config-if)# **dialer pool 1**
CPE(config-if)# **encapsulation ppp**
CPE(config-if)# **ip address negotiated**

The line ip address negotiated instructs the client to use an IP address provided by the PPPoE server.

The PPP header adds 8 bytes of overhead to each frame. Assuming the default Ethernet MTU of 1500 bytes, we'll want to lower our MTU on the dialer interface to 1492 to avoid unnecessary fragmentation.
CPE(config-if)# **mtu 1492**

Lastly we assign our ISP-facing interface to our newly created PPPoE dial group:
CPE(config)# **interface f0/0**
CPE(config-if)# **no ip address**
CPE(config-if)# **pppoe-client dial-pool-number 1**
CPE(config-if)# **no shutdown**

If all is well, you should see a notification indicating the PPPoE session has successfully formed:
%DIALER-6-BIND: Interface Vi1 bound to profile Di1
%LINK-3-UPDOWN: Interface Virtual-Access1, changed state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface Virtual-Access1, changed
state to up

We can verify that interface Dialer1 has negotiated an IP address from the ISP router:
CPE# **show ip interface brief**

```
Interface              IP-Address      OK? Method Status                Protocol
FastEthernet0/0        unassigned      YES manual up                    up
[...]
Virtual-Access1        unassigned      YES unset  up                    up
Dialer1                10.0.0.2        YES IPCP   up                    up
```

show pppoe session shows our PPPoE session with the ISP router terminated on Dialer0, via FastEthernet0/0:
CPE# **show pppoe session**

```
    1 client session

Uniq ID  PPPoE  RemMAC         Port               Source   VA       State
         SID  LocMAC                              VA-st
   N/A    16  ca00.4843.0008  Fa0/0              Di1      Vi1      UP
             ca01.4843.0008                      UP
```

Authentication
Of course, at this point anyone can connect via PPPoE. Generally we only want to provide service to trusted (e.g. paying) customers, so adding some low-layer authentication would be a prudent step. PPP can use PAP or CHAP to authenticate clients, with the later heavily preferred.

On our ISP router, we'll create a local user account name **CPE** and the password **MyPassword**. (In real practice, account creation is typically performed on a back-end server and referenced via RADIUS or TACACS+ rather than being stored locally.)
ISP(config)# **username CPE password MyPassword**

Next we enforce CHAP authentication on our virtual template:
ISP(config)# **interface virtual-template 1**
ISP(config-if)# **ppp authentication chap callin**

This will terminate our client session, as we can see from the logs on CPE:
%DIALER-6-UNBIND: Interface Vi1 unbound from profile Di1
%LINK-3-UPDOWN: Interface Virtual-Access1, changed state to down

%LINEPROTO-5-UPDOWN: Line protocol on Interface Virtual-Access1, changed state to down

To reestablish the connection from CPE, we'll need to enter the proper credentials:
CPE(config)# **interface dialer 1**
CPE(config-if)# **ppp chap password MyPassword**

We should see the PPPoE session come back up a few seconds later after successfully authenticating. debug ppp authentication can be used on the ISP router to monitor the CHAP exchange:
ppp50 PPP: Using vpn set call direction
ppp50 PPP: Treating connection as a callin
ppp50 PPP: Session handle[E800003A] Session id[50]
ppp50 PPP: Authorization required
ppp50 CHAP: O CHALLENGE id 1 len 24 from "ISP"
ppp50 CHAP: I RESPONSE id 1 len 24 from "CPE"
ppp50 PPP: Sent CHAP LOGIN Request
ppp50 PPP: Received LOGIN Response PASS
Vi1.1 PPP: Sent LCP AUTHOR Request
Vi1.1 PPP: Sent IPCP AUTHOR Request
Vi1.1 LCP: Received AAA AUTHOR Response PASS
Vi1.1 IPCP: Received AAA AUTHOR Response PASS
Vi1.1 CHAP: O SUCCESS id 1 len 4
By stretch