FLOW ROUTE REFLECTORS

UNDER PROTOCOL BGP

FULL MESH RR GROUP

This peering group is used for peering will all the other route reflectors.

```
group FLOWSPEC_FULLMESH_RR{
    type internal;
    local-address 10.1.1.4;
    family inet {
        flow {
            prefix-limit {
                maximum 500;
            }
            no-validate ACCEPT_FLOWSPECIFICATION;
        }
    }
    neighbor 10.1.1.3 {
        import ACCEPT_FLOWSPECIFICATION;
    }
}
```

CLIENTS GROUP

Under protocol BGP we will have a group to peer with all the FLOW clients in this scenario peers & transits:

```
group FLOWSPECIFICATION_CLIENTS {
    type internal;
    traceoptions {
        file inetflow.log;
        flag all;
    }
    family inet {
        flow {
            prefix-limit {
                maximum 500;
            }
            no-validate ACCEPT_FLOWSPECIFICATION;
        }
    }
    Cluster 10.1.1.1;
    neighbor 10.1.2.12{
        description CLIENT_ROUTER;
}
```

This is the flow specification BGP group and currently there is only one neighbor configured and is the peer-

01 router, this group will be used to add all the remaining peers and transits.

POLICY OPTIONS

POLICY STATEMENTS

Policy map used for validating the flow routes from BGP.

```
policy-statement ACCEPT_FLOWSPECIFICATION {
    term ACCEPT_FLOWSPECIFICATION_ROUTES {
        from protocol bgp;
        then accept;
    }
}
policy-statement ACCEPT_FLOWSPECIFICATION {
    term ACCEPT_FLOWSPECIFICATION_ROUTES {
        from {
            protocol bgp;
            community FLOWSPECIFICATION_FULL_MESH;
            route-filter 0.0.0.0/0 prefix-length-range /24-/32;
        }
        then accept;
    }
    then reject;
}

policy-statement NO_ROUTES {
    from protocol bgp;
    then reject;
}
```

PEERS & TRANSIT.

Peer and Transit routers is the door to the entrusted side this is where most of the attacks and alerts are coming from, so it will be the main router for mitigation and so each Peer and transit will peer with the route reflectors to obtain FLOW SPECIFICATION ROUTES.

PROTOCOL BGP
BGP group used to peer with the route reflectors

```
group INET_FLOW_BLACKHOLE {
    type internal;
    traceoptions {
        file inetflow.log size 50m;
        flag all;
    }
    family inet {
```

```
        flow {
            prefix-limit {
                maximum 500;
            }
            no-validate ACCEPT_FLOWSPECIFICATION;
        }
    }
    neighbor 10.1.199.103 {
        description FLOWSPECIFICATION_RR-3;
        import ACCEPT_FLOWSPECIFICATION;
        export NO_ROUTES;
    }
    neighbor 10.1.199.104 {
        description FLOWSPECIFICATION_RR-4;
        import ACCEPT_FLOWSPECIFICATION;
        export NO_ROUTES;
    }
}
```

POLICY OPTIONS

POLICY STATEMENT

Policy statement used for importing the routes from the route reflectors, it only allows to import the FLOWSPEC unique community for the router in question and a more general community.

There is also a route filter that will only import prefixes from a /32 up to a /24.

```
policy-statement ACCEPT_FLOWSPECIFICATION
term ACCEPT_FLOWSPECIFICATION_ROUTES {
    from {
        protocol bgp;
        community [ FLOWSPEC FLOWSPEC_SAME_ROLE];
        route-filter 0.0.0.0/0 prefix-length-range /24-/32;
    }
    then accept;
}
then reject;
```

COMMUNITIES (GRANULARITY)

DEVICE ROLE
CITY CODE
ROUTING SOURCE
DEVICE NUMBER

```
community FLOWSPEC members [ 65000:666 65001:99];
community FLOWSPEC_SAME_ROLE members [ 65000:0 65101:99];
```

Note: communities for each device on the network are specified on a separate document.

FLOW SPECIFICATION ROUTER

This router is the interface for adding and removing flow specification routes the purpose of this router is to have an interface for adding such routes with out affecting the route reflectors and their clients. Currently the design is based on a single homed FLOW SPECIFICATION ROUTER but in the future the idea is to have redundant servers.

Application

ARBOR & DDOS MITIGATION

FUNCTIONALITY
This application uses ARBOR as the tool to identify and attack or anomaly with a certain profile such as a source and destination address and also port number and such. Once it gets identified by ARBOR it will facilitate an option to mitigate such anomaly.

The mitigation can be performed with ARBOR using flow specification to mitigate such flow. Such mitigation is advertised to the FLOW SPECIFICATION ROUTE REFECTORS and they will advertise such mitigation to the PEERS & TRANSITS with the destination COMMUNITY that will be imported by the device that will identified that community.

DEVICE CONFIGURATION

FLOW SPECIFICATION ROUTE REFLECTORS

PROTOCOL BGP

The route reflectors will have a peering group that will be required for peering with ARBOR.

```
group ARBOR_FLOWSPECICIFATION {
    type internal;
    local-address 10.1.199.104;
    family inet {
        flow {
            prefix-limit {
                maximum 500;
            }
            no-validate ACCEPT_FLOWSPECIFICATION;
        }
    }
    neighbor 10.1.198.173;
}
```

POLICY OPTIONS
Policy option to accept all advertisements from ARBOR.

```
policy-statement ACCEPT_FLOWSPECIFICATION {
    term ACCEPT_FLOWSPECIFICATION_ROUTES {
        from protocol bgp;
        then accept;
    }
}
```

Routing Options FLOW.
With out using ARBOR here are some configuration examples creating flow routes with arbor

```
validation {
    traceoptions {
        file flowroute.log;
        flag all;
    }
}
route test1020 {
    match {
        destination 10.200.251.254/32;
        destination-port 80;
    }
    then {
        community COMMUNITY_ROLE_XX;
```

```
        discard;
        sample;
    }
}
route test1021 {
    match dscp 1;
    then {
        community COMMUNITY_ROLE_XX;
        discard;
    }
}
route test1022 {
    match dscp 2;
    then {
        community COMMUNITY_ROLE_XX;
        discard;
    }
}
route test1023 {
    match destination 10.200.251.254/32;
    then {
        community COMMUNITY_ROLE_XX;
        routing-instance target:1000:1000;
    }
}


route test1030 {
    match {
        destination 10.200.251.254/32;
        source 10.200.250.254/32;
        source-port [ 70 80 ];
    }
    then {
        community COMMUNITY_ROLE_XX;
        rate-limit 100k;
        routing-instance target:1000:1000;
    }
}
```