Setting up Genesys Multicloud CX Private Edition

Configure F5 SIP-ALG

8/1/2025

Contents

Learn how to configure F5 SIP-ALG as the final step in SBC integration with the private edition deployment on OpenShift.

Related documentation:

-
-

RSS:

- For private edition

Create SNATs and pools

1. Create the Source Network Address Translations (SNATs).

   ```
   create ltm snat-translation /f5-voice-sipproxy/10.30.3.2 address 10.30.3.2 traffic-
   group /Common/traffic-group-1

   create ltm snat-translation /f5-voice-sipproxy/10.10.2.11 address 10.30.3.2 traffic-
   group /Common/traffic-group-1
   ```

   When traffic is source network address translated to the cluster via the VXLAN tunnel or to the SBCs, translations are used. VXLAN tunnel host subnets provide the source NAT IP addresses used in Network Address Translation to the cluster.

2. Add the SNATs to a SNAT pool.

   ```
   create ltm snatpool f5-voice-sipproxy/snat-towards-sip-servers members add {
   10.30.3.2 10.10.2.11 }
   ```

   The SNAT pool is used when traffic is source network address translated to the OpenShift Cluster within the tunnel and also the SBCs on the external side. The F5 selects the appropriate SNAT translation when sending it out to the VXLAN/external VLAN.

Sip-media handler iRules

1. Create the two iRules required to handle SIP and media:
   media-handler

   ```
   #irule: media-handler

   when CLIENT_ACCEPTED {
           #log local0.debug "In Client Accepted"
           if { ![info exists read_table_flow_info] } {
               if { [set target_node_and_snat [table lookup sdp-l-[IP::remote_addr]-
   [UDP::remote_port]-[IP::local_addr]-[UDP::local_port]]] eq "" } {
   ```

```
                log local0.debug "Received udp flow for undefined definition
([IP::remote_addr]:[UDP::remote_port][IP::local_addr]:[UDP::local_port])"
                reject
                return
            }

            set read_table_flow_info yes

            set snat_ip   [getfield $target_node_and_snat "-" 1]
            set snat_port [getfield $target_node_and_snat "-" 2]
            set node_ip   [getfield $target_node_and_snat "-" 3]
            set node_port [getfield $target_node_and_snat "-" 4]

            log local0.debug "Allowing udp flow for
([IP::remote_addr]:[UDP::remote_port][IP::local_addr]:[UDP::local_port]) using node
($node_ip:$node_port) and snat ($snat_ip:$snat_port)"

            node $node_ip $node_port
            snat $snat_ip $snat_port
        }
    }
}
```

sip-sdp-handler

```
#irule: sip-sdp-handler

when RULE_INIT {
        set static::sip_sdp_handler_emit_debug_logs yes
        set static::sip_sdp_handler_flow_holddown_timeout 10
    }

    proc log_debug {msg} {
        if { $static::sip_sdp_handler_emit_debug_logs } {
            if { [clientside] } { set side clientside }
            else                { set side serverside }

            log local0.debug "($side
[IP::remote_addr]:[TCP::remote_port][IP::local_addr]:[TCP::local_port]) $msg"
        }
    }

    proc set_sdp_connection_field_for_ip {ip_address} {
        if { $ip_address contains ":" } {
            call log_debug " - setting c= to IN IP6 \[$ip_address\]"
            SDP::field c 0 "IN IP6 \[$ip_address\]"
        }
        else {
            call log_debug " - setting c= to IN IP4 $ip_address"
            SDP::field c 0 "IN IP4 $ip_address"
        }
    }

    # call create_media_listener_table_entry when_src_is 10.1.10.10-3000 and_dest_is
10.1.10.100-40000 then_snat_is 10.1.20.100-40000 and_node_is 10.1.20.20-5000
    #
    # The literals when_src_is, and_dest_is, then_snat_is and and_node_is are
syntactic sugar.  The args must be in the order
    # provided above.  Note the use of a dash (-) rather than colon (:) for ip/port
separator.  This facilitates easy use of IPv6.
    # This creates the table entry:
    #   $src-$dest -> $snat-$node
    proc create_media_listener_table_entry {l1 src l2 dest l3 snat l4 node} {
```

```
        call log_debug " - adding table entry with key (sdp-l-$src-$dest) and value
($snat-$node)"
        table set "sdp-l-$src-$dest" "$snat-$node"
$static::sip_sdp_handler_flow_holddown_timeout indef
    }

    when CLIENT_ACCEPTED {
        set reverse_flow_id ""
    }

    when SERVER_CONNECTED {
        set reverse_flow_id ""
    }

    when MR_INGRESS {
        if { [clientside] } {
            call log_debug " - storing message lasthop on clientside"
            set reverse_flow_id [MR::message lasthop]
            MR::store reverse_flow_id
        } else {
            if { $reverse_flow_id ne "" } {
                MR::message nexthop $reverse_flow_id
                call log_debug " - using stored reverse flow id = ($reverse_flow_id)"
            } else {
                log local0.warn "No reverse flow id, dropping message"
                MR::message drop "no reverse flow id"
            }
        }
    }

    when MR_EGRESS {
        if { [serverside] } {
            MR::restore reverse_flow_id
            call log_debug " - restoring flow_id = ($reverse_flow_id)"
        }
    }

    when SIP_REQUEST_SEND {
        if { [SIP::method] ne "INVITE" or [SIP::call_id] eq "" } {
            return
        }

        call log_debug "SIP::method = ([SIP::method]); Call-ID = ([SIP::call_id])"

        set clientside_media_ip [getfield [SDP::field connection_address] " " 3]
        call log_debug " - found ([SDP::media count]) media entries in SDP;
connection_address = ($clientside_media_ip); conn info = ([SDP::field
connection_address])"

        # Strictly speaking, this isn't correct, as there may be more than one c=,
but customer indicated
        # that there will only ever be one (and only one m= declaration, though we
can be more pedantic for that).
        call set_sdp_connection_field_for_ip [IP::local_addr]

        # 64512 == 65535 - 1024 + 1.  Reserve port after incr plus one more (for
RTCP).  However, because the
        # table entry cannot be set in RULE_INIT (table is not allowed in RULE_INIT)
and we must use
        # 'table incr', the first port returned will be 1026, not 1024.
        if { [set vs_listen_port [expr { [table incr sdp-xlat-port-incr 2] % 64512 +
1024 }]] == 1026 } {
            call log_debug "Setting timeout and lifetime for table (sdp-xlat-port-
```

```
incr) to indef"
            table timeout  sdp-xlat-port-incr indef
            table lifetime sdp-xlat-port-incr indef
        } elseif { $vs_listen_port == 5060 } {
            # SIP OPTIONS requests are made by the server to the same address as
we're using for snat, so we need to skip that port
            # We don't simply set this to 5062 statically and increment the table
because, while 'table incr' is atomic,
            # the work between the previous 'table incr' and this one is not.
            set listen_port [expr { [table incr sdp-xlat-port-incr 2] % 64512 + 1024
}]
        }

        call log_debug " - Ports reserved for media are ($vs_listen_port) and ([expr
{ $vs_listen_port + 1 }])"

        for { set i 0 } { $i
```

2. Create the SBC pool.

```
create ltm pool /f5-voice-sipproxy/sbc-pool { members add { :5080 { address
 }
```

3. Create the SIP Proxy pool.

```
create ltm pool /f5-voice-sipproxy/sipproxy-pool { members add { :5080 { address
 }
```

Create message routing components

1. Create the message-routing SIP profile.

```
 create ltm message-routing sip profile session sipsession-sdp-rewrite { app-service
none defaults-from /Common/sipsession insert-record-route-header enabled record-
route-mode double }
```

2. Create the message-routing transport configuration.

```
 create ltm message-routing sip transport-config /f5-voice-sipproxy/tc-sdp-rewrite1 {
profiles add { sipsession-sdp-rewrite { } /Common/f5-tcp-progressive { } } rules {
/f5-voice-sipproxy/sip-sdp-handler } source-address-translation { pool snat-towards-
sip-servers type snat } source-port 5080
```

3. Create the message-routing SIP peers.

```
create ltm message-routing sip peer /f5-voice-sipproxy/peer-sbc pool sbc-pool
transport-config /f5-voice-sipproxy/tc-sdp-rewrite

create ltm message-routing sip peer /f5-voice-sipproxy/peer-sip-servers pool sipproxy-
pool transport-config /f5-voice-sipproxy/tc-sdp-rewrite
```

4. Create the message-routing SIP profile router.

```
create ltm message-routing sip profile router /f5-voice-sipproxy/router-sdp-rewrite
{app-service none defaults-from /Common/siprouter traffic-group /Common/traffic-
group-1 }
```

   You need to first create the message-routing routes referencing the virtual servers. Later, add these routes back into the SIP profile router.

Create virtual servers

A SIP server and media virtual server is required in either direction:

1. Externally facing the SBCs for an inbound call
2. Internally facing the sip-proxy for outbound calls

sbc-facing-sip

```
ltm virtual vs-facing-sbc-sip {
   destination 10.10.2.11:5080
   ip-protocol tcp
   mask 255.255.255.255
   partition f5-voice-sipproxy
   profiles {
       /Common/f5-tcp-progressive { }
       router-sdp-rewrite { }
       sipsession-sdp-rewrite { }
   }
   rules {
       sip-sdp-handler
   }
   serverssl-use-sni disabled
   source 0.0.0.0/0
   translate-address enabled
   translate-port enabled
   vlans {
       /Common/external
   }
   vlans-enabled
```

sbc-facing-media (vs-media-sbc-facing)

```
ltm virtual vs-media-sbc-facing {
    destination 10.10.2.11:any
    ip-protocol udp
    mask 255.255.255.255
    partition f5-voice-sipproxy
    profiles {
        /Common/udp { }
    }
    rules {
        media-handler
    }
    serverssl-use-sni disabled
    source 0.0.0.0/0
    translate-address enabled
    translate-port enabled
    vlans {
        /Common/external
    }
    vlans-enabled
```

sipservers-facing-sip (vs-facing-sip-servers)

```
ltm virtual vs-facing-sip-servers {
    destination 10.30.3.5:5080
    ip-protocol tcp
    mask 255.255.255.255
```

```
    partition f5-voice-sipproxy
    profiles {
        /Common/f5-tcp-progressive { }
        router-sdp-rewrite { }
        sipsession-sdp-rewrite { }
    }
    rules {
        sip-sdp-handler
    }
    serverssl-use-sni disabled
    source 0.0.0.0/0
    translate-address enabled
    translate-port enabled
    vlans {
        /Common/openshift_vxlan
    }
    vlans-enabled
```

sipservers-facing-media (vs-facing-sip-servers)

```
ltm virtual vs-media-servers-facing {
    destination 10.30.3.5:any
    ip-protocol udp
    mask 255.255.255.255
    partition f5-voice-sipproxy
    profiles {
        /Common/udp { }
    }
    rules {
        media-handler
    }
    serverssl-use-sni disabled
    source 0.0.0.0/0
    translate-address enabled
    translate-port enabled
    vlans {
        /Common/openshift_vxlan
    }
    vlans-enabled
```

1. Create the message-routing SIP routes.

```
create ltm message-routing sip route /f5-voice-sipproxy/route-to-sbcs peers {
 /f5-voice-sipproxy/peer-sbc } virtual-server /f5-voice-sipproxy/vs-facing-sbc-sip

create ltm message-routing sip route /f5-voice-sipproxy/route-to-sip-servers peers {
 /f5-voice-sipproxy/peer-sip-servers } virtual-server /f5-voice-sipproxy/vs-facing-
 sip-servers
```

2. Add the SIP routes to the message-routing SIP profile router.

```
modify ltm message-routing sip profile router /f5-voice-sipproxy/router-sdp-rewrite {
 routes add { /f5-voice-sipproxy/route-to-sbcs /f5-voice-sipproxy/route-to-sip-
 servers } }
```