



Communication Controller for Linux on System z9 and zSeries

IPTG – INN over IP with STUNNEL

Sample Definitions for Communications
Controller for Linux on System z9 and zSeries

Target Audience

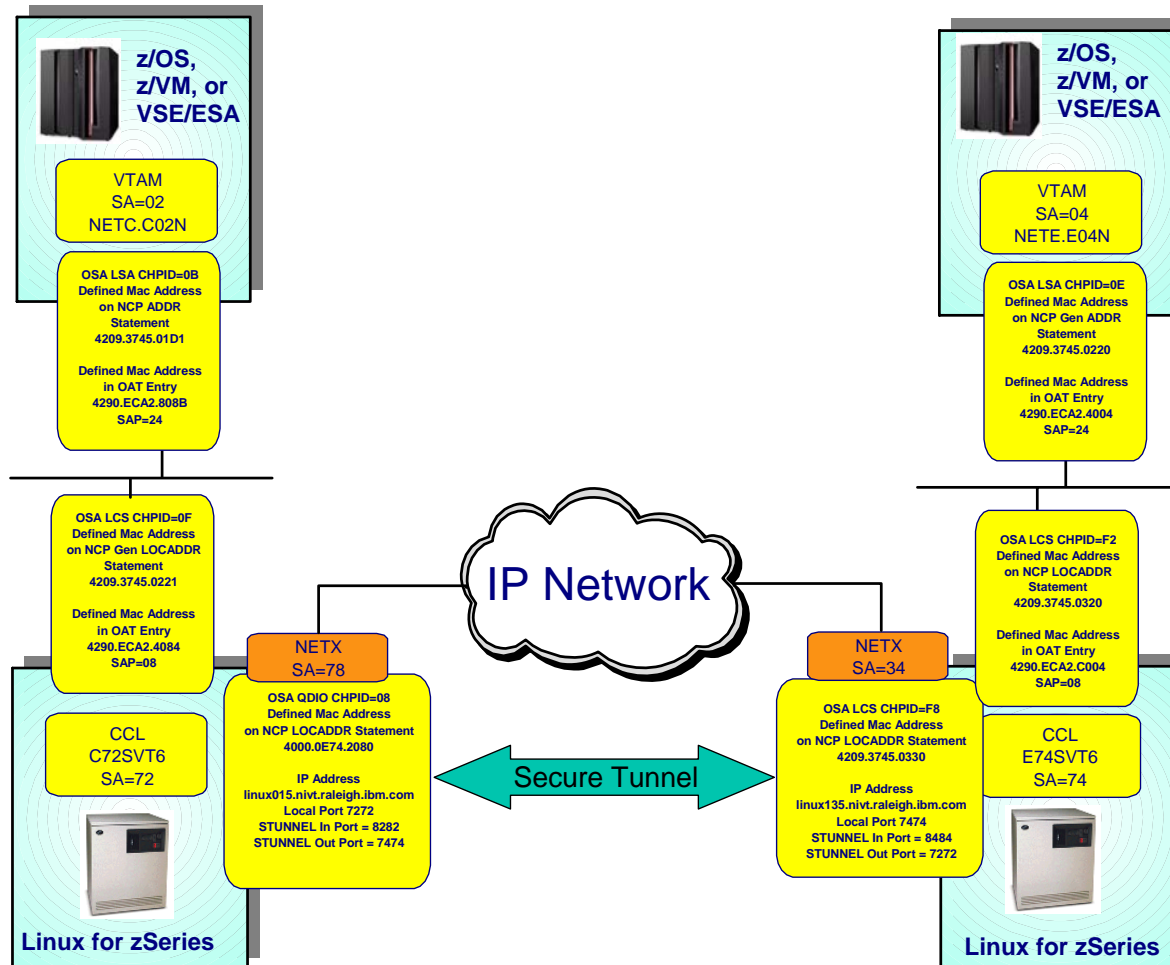
- Customers wanting a sub-area connection between two Communication Controller for Linux on System z9 and zSeries nodes using a secured IP network as a transport medium.
- This connection cannot be made CCL to VTAM or CCL to 3745/46

Purpose of this Paper

The intent of this paper is to provide a sample solution for customers during the migration from 3745/3746-900 FEPs to Communication Controller for Linux on System z9 and zSeries (CCL). This document will provide working examples of the following:

- NCP Physical and Logical lines
- Flat file definitions for IPTG
- Definitions to setup the Secure Tunnel (STUNNEL)

Test Configuration



Resources Used for Solution Verification

- Two z/OS Communications Servers
- Two Linux IDs running as guest under z/VM
 - 512mb of memory
 - 3 Virtual CPs
 - 2 3390-3 DASD volumes
- Four OSA Copper Ethernet OSA adapters
 - LCS Connections from Communication Controller for Linux on System z9 and zSeries to LSA connection in z/OS.
- Two QDIO OSA Adapters
- Layer 2 or Layer 3 Ethernet Switches

STUNNEL Setup

- **Install STUNNEL Package using YAST**
 - From YAST main menu, select “Software”
 - Select “Install and Remove Software”
 - In the Search field enter “stunnel”
 - In the package list check the box next to stunnel
 - Click the Accept button to install the stunnel package

- **Generate PEM Files**
 - `cd /etc/stunnel`
 - `openssl req -newkey rsa:1024 -keyout key.pem -nodes -x509 -days 365 -out cert.pem`
 - `cat key.pem cert.pem >stunnel.pem`
 - `rm key.pem cert.pem`

STUNNEL Notes

- When defining the STUNNEL configuration, you are permitted to configure both secure and non-secured connections in the same profile by defining multiple LOCALNODE definitions. Each LOCALNODE definition would have a unique port.
- Connections to VTAM were intentionally omitted from this presentation. Please refer to other Communication Controller for Linux on System z9 and zSeries presentations for these connectivity options.
- For an IPTG connection, the side with largest TIC3 physical LOCADDR will listen for a connection from the other side.

STUNNEL Flows

- Remote IP-TG PU connects to local stunnel
- Local stunnel opens connection to remote stunnel
- Remote stunnel connects to the target IP-TG port
- Data on IPTG connection is sent to local stunnel
- Local stunnel encrypts and forwards data to remote stunnel
- Remote stunnel decrypts and forwards data to the target IP-TG port

C72SVT6 – TIC3 Physical Line Definitions

```
*****
* PHYSICAL TOKEN RING INTERFACE FOR TCP/IP CONNECTIONS - TIC3 2080 *
*****
*
C72IPPG  GROUP  ECLTYPE=(PHY,SUB),ADAPTER=TIC3,ANS=CONT,ISTATUS=ACTIVE, X
          RCVBUFC=32000,MAXTSL=16732,RETRIES=(20,5,5)
*
C72IPPL  LINE   ADDRESS=(2080,FULL),PORTADD=80,                                X
          LOCADD=40000C722080,NPACOLL=NO
C72IPPP  PU     ADDR=01,XMONLNK=YES
*
```


C72SVT6 – TIC3 Logical Lines Definitions

```

*****
* LOGICAL INN TCP/IP CONNECTIONS *
*****
*
C72IPLG  GROUP  ECLTYPE=(LOGICAL,SUBAREA),ANS=CONT,ISTATUS=ACTIVE,      X
          PHYSRSC=C72IPPP,SDLCST=(C72PRI,C72SEC),NPACOLL=NO,           X
          T2TIMER=(1.5,2.0,3),LOCALTO=13.5,REMOTTO=18.2
*
*-----
* Linkstation to E74 - IPTG INN Connection
*-----
*
C72IPLL5  LINE   TGN=1,TGCONF=SINGLE,MONLINK=YES
C72IPLP5  PU     ADDR=0840000E742080,SSAP=(08,H),NETID=NETX
*

```

C72SVT6 – Flat File Definitions

```
ccldefs
  TCPDEFS
* -----
* Define Local IPTG Port
* -----
*
  LOCALNODE
    IPPORT      7272
    IPTOS       LOWDELAY
*
* -----
* Define stunnel IPTG Port
* -----
*
  REMOTENODE
    IPTOS       LOWDELAY
    PUNAME      C72IPLP5
    HOST        linux015.nivt.raleigh.ibm.com
    IPPORT      7474
  ENDTCPDEFS
endccldefs
```

STUNNEL Configuration Files – C72SVT6

- `stunnel_out.conf`
 - `client = yes`
 - `pid = /var/run/stunnel_out.pid`
 - `[ccl2ipll]`
 - `accept = 7474`
 - `connect = linux135.nivt.raleigh.ibm.com:8484`

E74SVT6 – TIC3 Physical Line Definitions

```
*****
*  PHYSICAL TOKEN RING INTERFACE FOR TCP/IP CONNECTIONS - TIC3 2080  *
*****
*
E74IPPG  GROUP  ECLTYPE=(PHY,SUB),ADAPTER=TIC3,ANS=CONT,ISTATUS=ACTIVE, X
          RCVBUFC=32000,MAXTSL=16732,RETRIES=(20,5,5)
*
E74IPPL  LINE   ADDRESS=(2080,FULL),PORTADD=80,                                X
          LOCADD=40000E742080,NPACOLL=NO
E74IPPP  PU     ADDR=01,XMONLNK=YES
*
```

E74SVT6 – TIC3 Logical Line Definitions

```

*****
* LOGICAL INN TCP/IP CONNECTIONS *
*****
*
E74IPLG  GROUP  ECLTYPE=(LOGICAL,SUBAREA),ANS=CONT,ISTATUS=ACTIVE,      X
          PHYSRSC=E74IPPP,SDLCST=(E74PRI,E74SEC),NPACOLL=NO,          X
          T2TIMER=(1.5,2.0,3),LOCALTO=13.5,REMOTTO=18.2
*
*-----
* Linkstation to C72 - IPTG INN Connection
*-----
*
E74IPLL5  LINE   TGN=1,TGCONF=SINGLE,MONLINK=YES
E74IPLP5  PU     ADDR=0840000C722080,SSAP=(08,H),NETID=NETX
*

```

E74SVT6 – Flat File Definitions

```
ccldefs
  TCPDEFS
*-----
* Define Local IPTG Port
*-----
*
  LOCALNODE
    IPPORT          7474
    IPTOS           LOWDELAY
*
  ENDTCPDEFS
endccldefs
```

STUNNEL Configuration Files – E74SVT6

- `stunnel_in.conf`
 - `[ccl2iptg]`
 - `accept = 8484`
 - `connect = 7474`
 - `TIMEOUTclose = 0`

Starting CCL from Linux

- From the Linux console, change to the CCL directory:
 - `cd /opt/ibm/ndh`
- Load the CCL kernel module
 - `./load_ndh.sh`
 - You will receive the message
NDH kernel modules loaded. You are now able to run the cclengine
- From the Linux console, change to the CCL directory:
 - `cd /opt/ibm/Communication_Controller_for_Linux/`
- Start STUNNEL on both Linux IDs
 - Linux135 run `"stunnel /etc/stunnel/stunnel_in.conf"`
 - Linux015 run `"stunnel /etc/stunnel/stunnel_out.conf"`
- Start the CCL engine
 - `nohup ./cclengine -mC72SVT6 -p2072 SVTC72 &`
 - If you use telnet or ssh into the Linux host you will want to preface the command with "nohup" so that the process will remain active even after the telnet/ssh session is terminated.

Activating NCP using XCA from NETC.C02N

- **From NETC.C02N activate the XCA major node**

```
V NET,ACT,ID=C02XCA,ALL
IST097I VARY ACCEPTED
IST093I C02XCA ACTIVE
IST464I LINK STATION C02ETHP1 HAS CONTACTED SA 72
IST093I C02ETHP1 ACTIVE
```

- **From NETC.C02N activate the NCP**

```
V NET,ACT,ID=C72SVT6,ALL
IST097I VARY ACCEPTED
IST093I C72SVT6 ACTIVE
IST093I C72PU88A ACTIVE
IST093I C72PU89A ACTIVE
IST093I C72IPPP ACTIVE
IST464I LINK STATION C72PG2A HAS CONTACTED C02NPU SA 2
IST093I C72PG2A ACTIVE
IST720I C72IPLP5 HAS CONTACTED E74SVT6 IN NETX, SA 34
IST093I C72IPLP5 ACTIVE
```

Activating NCP using XCA from NETE.E04N

- **From NETE.E04N activate the XCA major node**

```
V NET,ACT,ID=E04XCA,ALL
IST093I E04XCA ACTIVE
IST464I LINK STATION E04ETHPU HAS CONTACTED E74SVT6 SA 74
IST093I E04ETHPU ACTIVE
```

- **From NETE.E08N activate the NCP**

```
V NET,ACT,ALL,ID=E74SVT6
IST097I VARY ACCEPTED
IST093I E74SVT6 ACTIVE
IST093I E74PU92A ACTIVE
IST093I E74PU93A ACTIVE
IST093I E74IPPP ACTIVE
IST464I LINK STATION E74PG1A HAS CONTACTED E04NPU SA 4
IST093I E74PG1A ACTIVE
IST720I E74IPLP5 HAS CONTACTED C72SVT6 IN NETX, SA 78
IST093I E74IPLP5 ACTIVE
```