Understanding and Troubleshooting SDLC to LLC Network Media Translation

Document ID: 12250

Contents

Introduction

This document provides information to understand and troubleshoot a Synchronous Data Link Control (SDLC) to Logical Link Control (LLC) network media translation.

Prerequisites

Requirements

There are no specific requirements for this document.

Components Used

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

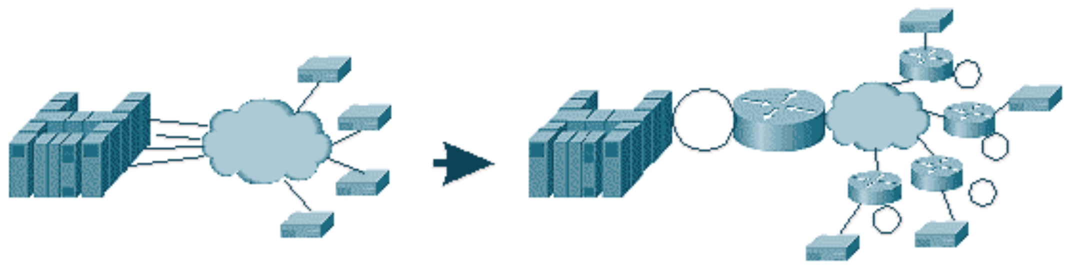
Conventions

Refer to Cisco Technical Tips Conventions for more information on document conventions.

SDLLC

SDLC−to−LAN conversion (SDLLC) is used to convert a SDLC session for a Physical Unit 2 (PU2.0) device to a Logical Link Control, type 2 (LLC2) session. This is very useful if you have a large amount of remote controllers fed into a single Token Ring port on a front−end processor (FEP).

The left side of this diagram displays a FEP with many SDLC lines leaving to remote locations. The right side of this diagram displays the same scenario with Cisco routers.



The routers allow the FEP to have only the Token Ring interface. From that point, there are multiple remote locations performing SDLLC to the host, as well as regular source−route bridge (SRB) traffic.

Note: Using SDLLC for LLC to SDLC conversion only applies for PU2.0 devices, not for Physical Unit type 2.1 (PU2.1). PU2.1 is supported in data−link switching (DLSw).

To configure SDLLC, you need a SRB in the router. Refer to Understanding and Troubleshooting Local Source−Route Bridging for information on how to configure a SRB.

SDLC Configuration

Because SDLLC converts from an SDLC interface, you first need SDLC correctly configured. Complete these steps to configure SDLC:

1. Issue the **encapsulation sdlc** command to change the serial encapsulation to SDLC.
2. Issue the **sdlc role primary** command to change the role of the router to primary in the SDLC line.

   Note: In Serial Tunneling (STUN) environments, there are primary and secondary roles. Refer to Configuring and Troubleshooting Serial Tunneling (STUN) for more information.
3. Issue the **sdlc address xx** command to configure SDLC polling address.

SDLLC Configuration

To configure SDLLC, the first command issued is **traddr**. This command defines what the SDLC converts to in the LLC2 environment. Complete these steps to configure SDLLC:

1. Issue the **sdllc traddr xxxx.xxxx.xx00 lr bn tr** command to enable SDLLC media translation on a serial interface.

   This command tells the router the virtual MAC address of the SDLC station. Then the command specifies the local ring number (**lr**), the bridge number (**bn**), and the target ring number (**tr**). The **lr** has to be unique in the network. The **bn** can be a value from 1 to 15. The **trn** must be the virtual ring in the router. If you are configuring local SDLLC, you can make this point to a virtual ring or to an interface (physical ring connected to the Token Ring interface) in the router.

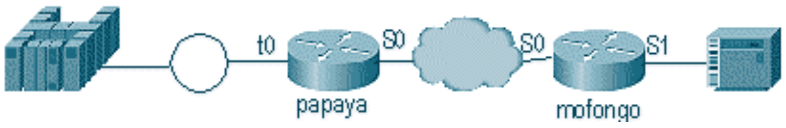
   Note: The last two digits of the MAC address in this command are **00**. You cannot set the last two digits of **traddr** because the router uses these digits to insert the SDLC address of this line. If you do specify the last two digits, the router replaces them with the SDLC address. Then the host does not respond for that MAC address. For example, if traddr MAC is configured as 4000.1234.5678 and SDLC address is 0x01, the router uses the MAC of 4000.1234.5601 to represent the SDLC device in the LLC domain. Furthermore, the traddr MAC is in non−canonical format, which is the same format as Token Ring frame.

2. Issue the **sdllc xid** *address* **xxxxxxxx** command to specify the exchange identification (XID) value appropriate for the SDLC station to match Virtual Telecommunications Access Method (VTAM) values.

   This is determined from the IDBLK and IDNUM in the switch major node in VTAM. If this does not match, the XID exchange fails.
3. Issue the **sdllc partner** *mac−address sdlc−address* command to enable connections for SDLLC.

   This specifies the MAC address of the partner, which is usually the host.

A simple SDLLC sample configuration is displayed. The SDLC attached controller appears as a local Token Ring attached device to the FEP.



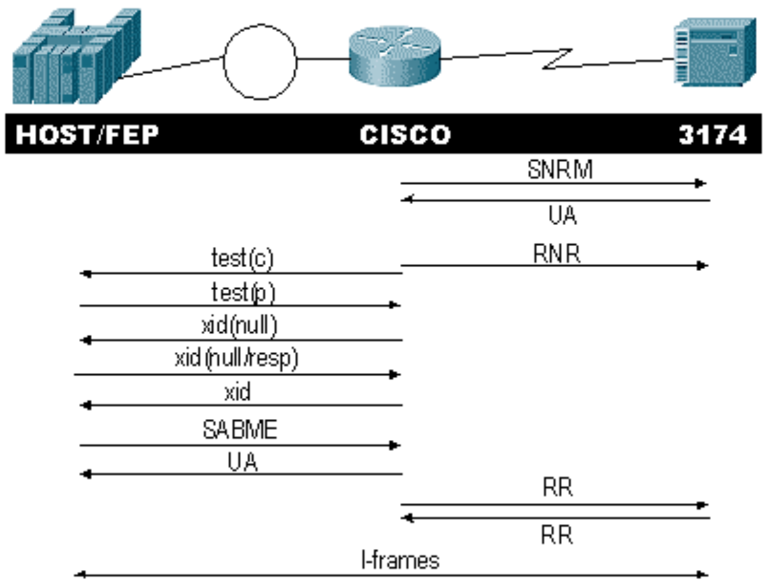
Papaya	Mofongo

`source−bridge ring−group 100`
`source−bridge remote−peer 100 tcp 1.1.1.1`
`source−bridge remote−peer 100`
`tcp 1.1.2.1 local−ack`
 	
`interface tokenring 0`
`ip address 1.1.3.1 255.255.255.0`
`source−bridge 33 2 100`
`source−bridge spanning`
 	
`interface loopback 0`
`ip address 1.1.1.1 255.255.255.0`
	`source−bridge ring group 100`
	`source−bridge remote−peer 100 tcp 1.1.2.1`
	`source−bridge remote−peer 100`
	`tcp 1.1.1.1 local−ack`
	`source−bridge sdllc local−ack`
	
	`interface serial 0`
	`encapsulation sdlc−primary`
	`sdlc address c6`
	`sdllc traddr 4000.3174.1100 333 3 100`
	`sdllc partner 4000.1111.1111 c1`
	`sdllc xid c1 17200c6`
	
	`interface loopback 0`
	`ip address 1.1.2.1 255.255.255.0`

Debugging SDLLC

An SDLLC problem requires that you troubleshoot two different environments: the SDLC world, and the Logical Link Control, type 2 (LLC2) world to where you are translating the frames. Because you can only have one type of controller, debugging SDLLC is easier to understand than data−link switching (DLSw)/SDLC.

First, notice the flows for this specific session startup:



Check for the Set Normal Response Mode (SNRM) response from the controller. The router does not start the LLC portion until the SDLC portion is up and running.

Issue these commands to verify the SNRM response:

- **sdlc_state**
- **sdllc_state**

In this example, the SNRM is sent to the controller, which changes the state of the line to SNRMSENT. If the router remains in this state, then it has not received the unnumbered acknowledgment (UA) from the controller. This can mean that something is wrong with the SDLC line. If this occurs, the debug is displayed as:

```
%LINEPROTO-5-UPDOWN: Line protocol on Interface Serial1, changed state to up
s4f#
SDLLC_STATE: Serial1 C6 DISCONNECT
   -> SDLC PRI WAIT
SDLC_STATE: (5234984) Serial1 C6 DISCONNECT
   -> SNRMSENT
%SYS-5-CONFIG_I: Configured from console by console
%LINK-3-UPDOWN: Interface Serial1, changed state to up
Serial1 SDLC output     C693
Serial1 SDLC input      C673
SDLC_STATE: (5235700) Serial1 C6 SNRMSENT
   -> CONNECT
SDLLC_STATE: Serial1 C6 SDLC PRI WAIT
   -> NET UP WAIT
SDLC_STATE: (5235700) Serial1 C6 CONNECT
   -> USBUSY
```

If the router receives the UA, the **sdlc_state** moves from SNRM_SENT to CONNECT. Next, the SDLLC state moves from SDLC_PRI_WAIT to NET_UP_WAIT. When this occurs, the router can start bringing up the LLC side of the connection. The final action is to begin sending receive not ready(s) (RNRs) to the SDLC line. This disables the controller from sending any information until the LLC side is operational.

Next, the router sends an explorer to find the location of its partner.

```
SDLLC: O TEST, dst 4000.1111.1111 src 4000.3174.11c6 dsap 0 ssap 0
To0: out: MAC: acfc: 0x8040 Dst: 4000.1111.1111 Src: c000.3174.11c6  bf: 0x82 0x304A210
To0: out: RIF: 8800.14D3.0642.0210
```

```
To0: out: LLC: 0000F300 00800000 000C3BF0 7D000000 00800000 000C3BF0 ln: 25
SDLLC: NET UP WAIT    recv FORWARD TEST P/F(F3) 4000.3174.11c6 c000.1111.1111 00 01 -> Ser
caching rif
```

The preceding output displays the test poll being sent and received. Because this example has a locally attached controller and Token Ring, the test poll leaves the router searching for the partner address. After the router receives the test frame, it begins the XID exchange. The router caches the Routing Information Field (RIF) for this session, which you can verify with the **show rif** command. Because this is a PU2.0, the router sends a Format 0 Type 2 XID to the host after the response to the XID null.

```
SDLLC: O xid(null), 4000.1111.1111 4000.3174.11c6 4 4 [1000.14D3.0641.0051.12C2.0194.01F1.
SDLLC: NET UP WAIT    recv FORWARD XID P/F(BF) 4000.3174.11c6 c000.1111.1111 04 05
   -> Serial1 C6
SDLLC: O xid(0T2), 4000.1111.1111 4000.3174.11c6 4 4 [1000.14D3.0641.0051.12C2.0194.01F1.0
SDLLC: NET UP WAIT    recv FORWARD SABME P/F(7F) 4000.3174.11c6 c000.1111.1111 04 04
   -> Serial1 C6
SDLLC: SABME for Serial1 C6 in NET UP WAIT
%SDLLC-5-ACT_LINK: SDLLC: Serial1 LINK address C6 ACTIVATED: Net connect
SDLLC_STATE: Serial1 C6 NET UP WAIT    -> CONNECT
```

After the XID exchange, the router receives the Set Asynchronous Balanced Mode Extended (SABME) from the host. This finalizes the startup procedure, and the router responds with a UA to the host. Now, the state of the SDLC line changes from USBUSY to CONNECT, and I−frames can pass through the router.

```
SDLC_STATE: (5235944) Serial1 C6 USBUSY
   -> CONNECT
Serial1 SDLC output     C611
Serial1 SDLC input      C611
s4f#
```

DLSw Media Translation

DLSw provides a major ehancement to media translation because it supports PU2.1. This enables it to have SDLLC to LLC2 conversion for controllers, such as the 5494 and the 5394 (with upgrade option to PU2.1 − IBM RPQ 8Q0775) to AS/400s. This removes the need for STUN and bad AS/400 multipoint lines.

The configuration parameters for DLSw media translation are a little different from the SDLLC parameters. There is one DLSw command that is added, the rest are SDLC commands. Complete these steps to configure DLSw media translation:

1. Issue the **encapsulation sdlc** command to change the serial encapsulation to SDLC.

   Because you are going to terminate the SDLC line in the router, the router must act as primary for polling purposes. This is different from STUN because the primary is going to be the HOST or the AS/400.
2. Issue the **sdlc role primary** command to change the role of the router to primary in the SDLC line.
3. Issue the **sdlc address xx** command to configure the SDLC polling address.

   This is where DLSw differs from SDLLC. In SDLLC, you specify commands with the **sdllc** keyword. In DLSw, specify commands with the **sdlc** keyword.
4. Issue the **sdlc vmac xxxx.xxxx.xx00** command to configure the virtual MAC address for the SDLC controller.

   This parameter tells the router the virtual MAC address for this SDLC controller in the LLC2 environment. Remember to leave the last byte set to **00** because the polling address is added there (**sdlc address**).
5. Issue the **sdlc xid nn xxxxxxxx** command to configure the XID for this PU 2.0.

In this command, ***nn*** is the polling address of the controller and **xxxxxxxx** is the XID for this PU2.0 (the IDBLOCK and IDNUM that is coded in the switch major node in VTAM).

Note: If you have a PU2.1, there is negotiation of XID. Thus, the command changes.

6. Issue the **sdlc xid nn xid−poll** command to configure the XID for this PU 2.1.

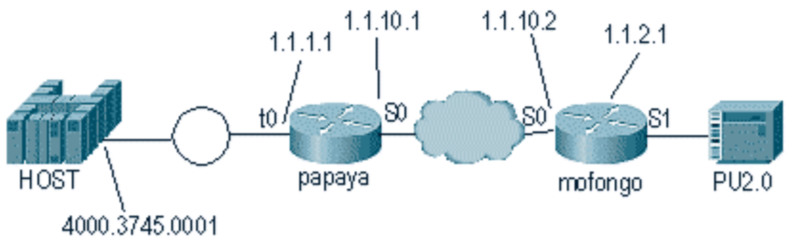
   In this command, ***nn*** is the polling address of the station.
7. Issue the **sdlc partner xxxx.xxxx.xxxx nn** command to configure the router partner MAC address.

   In this command, ***nn*** is the polling address for the controller in question. It is important to specify the controller address, because in multipoint lines there can be a controller headed for one host and another controller headed for a different host.
8. Issue the **sdlc dlsw nn** command to configure DLSw for the specific controller.

   In this command, ***nn*** is the polling address of the controller or controllers in the multidrop. This command allows you to specify multiple polling addresses in one command.

   Note: Beware of bug #CSCdi75481. Refer to Bug Toolkit (registered customers only) for more information. If the **sdlc dlsw nn** command is not removed before changing the SDLC address of the router, the CLS code cannot correctly communicate DLSw with the SDLC interface. This causes the interface to behave as if nothing was configured. This bug has been fixed in Cisco IOS® Software Releases 11.1(8.1) 11.1(8.1)AA01(01.03) 11.1(8.1)AA01(01.02) and later.

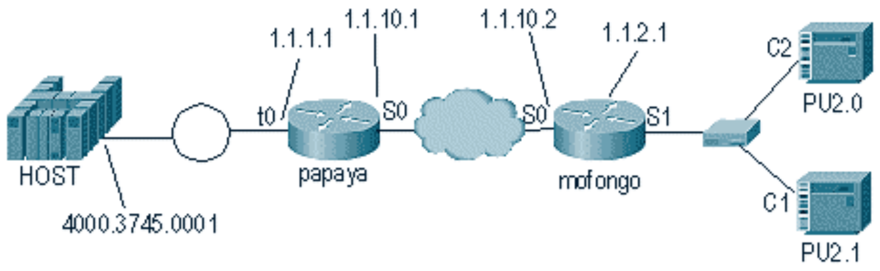
A sample configuration for a DLSw SDLC PU2.0 controller is displayed.



Papaya	Mofongo

`source−bridge ring−group 100`
`dlsw local−peer peer−id 1.1.1.1`
`dlsw remote−peer 0 tcp 1.1.2.1`
`!`
`interface serial 0`
`ip address 1.1.10.1 255.255.255.0`
`!`
`interface tokenring 0`
`ip address 1.1.1.1 255.255.255.0`
`ring−speed 16`
`source−bridge 1 1 100`
`source−bridge spanning`
	`dlsw local−peer peer−id 1.1.2.1`
	`dlsw remote−peer 0 tcp 1.1.1.1`
	`!`
	`interface loopback 0`
	`ip address 1.1.2.1`
	`!`
	`interface serial 0`
	`ip address 1.1.10.2 255.255.255.0`
	`!`
	`interface serial 1`
	`no ip address`
	`encapsulation sdlc`
	`sdlc role primary`
	`sdlc vmac 4000.3174.0000`
	`sdlc address c1`
	`sdlc xid c1 01767890`
	`sdlc partner 4000.3745.0001 c1`
	`sdlc dlsw c1`

When coding a multidrop, remember that PU2.1s are more intelligent and have more information to exchange than a regular PU2.0 device. This is important when configuring a multidrop environment, because you need to code the line as primary for the PU2.0 device. You also need to add the **xid−poll** for the SDLC address of the PU2.1 device so the code understands what to do with each of the controllers. This is an example of the configuration.



Papaya	Mofongo
source−bridge ring−group 100 dlsw local−peer peer−id 1.1.1.1 dlsw remote−peer 0 tcp 1.1.2.1 ! interface serial 0 ip address 1.1.10.1 255.255.255.0 ! interface tokenring 0 ip address 1.1.1.1 255.255.255.0 ring−speed 16 source−bridge 1 1 100 source−bridge spanning	dlsw local−peer peer−id 1.1.2.1 dlsw remote−peer 0 tcp 1.1.1.1 ! interface loopback 0 ip address 1.1.2.1 ! interface serial 0 ip address 1.1.10.2 255.255.255.0 ! interface serial 1 no ip address encapsulation sdlc sdlc role primary sdlc vmac 4000.3174.0000 sdlc address c1 xid−poll sdlc partner 4000.9404.0001 c1 sdlc address c2 01767890 sdlc partner 4000.9404.0001 c2 sdlc dlsw c1 c2

show Commands

Refer to Data−Link Switching Plus for more information on the show commands used for DLSw media translation.

Debugging SDLC Packets during DLSw/SDLC for PU2.1

```
%LINK−3−UPDOWN: Interface Serial2, changed state to up
```

The first thing to occur is an XID, or **BF** to the SDLC broadcast address of **FF**.

```
Serial2 SDLC output     FFBF
```

Next, an XID is received from the 5494. This is an XID format 2 type 3, which is displayed in this **debug sdlc packet** command output:

```
Serial2 SDLC input
0046C930: DDBF3244 073000DD 0000B084 00000000  ...........d....
0046C940: 00000001 0B000004 09000000 00070010  ................
0046C950: 17001611 01130012 F5F4F9F4 F0F0F2F0  ........54940020
0046C960: F0F0F0F0 F0F0F0F0 0E0CF4D5 C5E3C14B  00000000..4NETA.
0046C970: C3D7F5F4 F9F4                        CP5494
```

These are explanations of several of the fields from this command:

- **073000DD** This field is the Block ID and ID Num that is configured in the 5494. Block ID and ID Num are referred to as the XID, and are sent by the 5494 to the peer during session negotiation.
- **NETA** This field is the Advanced Peer−to−Peer Networking (APPN) Network Identifier (NETID) that is being used. Normally, this field matches the NETID that is configured in the peer. In this case, the peer is an AS/400.
- **CP5494** This field is the Control Point (CP) name of the 5494.
- **DD** This field is the SDLC address.

Next, the XID is received from the AS/400:

```
Serial2 SDLC output
004BC070:     FFBF 324C0564 52530000 000A0800    ...<.........
004BC080: 00000000 00010B30 0005BA00 00000007  ................
004BC090: 000E0DF4 D5C5E3C1 4BD9E3D7 F4F0F0C1  ...4NETA.RTP400A
004BC0A0: 1017F116 11011300 11F9F4F0 F4C6F2F5  ..1......9404F25
004BC0B0: F1F0F0F0 F4F5F2F5 F3460505 80000000  100045253.......
004BC0C0:
Serial2 SDLC input
0046C270:                   DDBF3244 073000DD          ........
0046C280: 0000B084 00000000 00000001 0B000004  ...d............
0046C290: 09000000 00070010 17001611 01130012  ................
0046C2A0: F5F4F9F4 F0F0F2F0 F0F0F0F0 F0F0F0F0  5494002000000000
0046C2B0: 0E0CF4D5 C5E3C14B C3D7F5F4 F9F4      ..4NETA.CP5494
Serial2 SDLC output
004C0B10:     FFBF 324C0564 52530000 00F6C800    ...<.......6H.
004C0B20: 00000080 15010B10 0005BA00 00000007  ................
004C0B30: 000E0DF4 D5C5E3C1 4BD9E3D7 F4F0F0C1  ...4NETA.RTP400A
004C0B40: 1017F116 11011300 11F9F4F0 F4C6F2F5  ..1......9404F25
004C0B50: F1F0F0F0 F4F5F2F5 F3460505 80150000  100045253.......
004C0B60:
Serial2 SDLC input
0046BBC0: DDBF3244 073000DD 0000B084 00000000  ...........d....
0046BBD0: 00000001 0B000004 09000000 00070010  ................
0046BBE0: 17001611 01130012 F5F4F9F4 F0F0F2F0  ........54940020
0046BBF0: F0F0F0F0 F0F0F0F0 0E0CF4D5 C5E3C14B  00000000..4NETA.
0046BC00: C3D7F5F4 F9F4                        CP5494
```

- **05645253** This field is the Block Id and Id Num of the AS/400.
- **RTP400A** This field is the CP name of the AS/400. The CP name is found in the Display Network Attributes (DSPNETA) file on the AS/400.

Then, the SNRM (93) and UA (73) are displayed on the line. Prior to the SNRM, the router always uses the broadcast address. From this point on, the router always uses the actual polling address of DD.

```
Serial2 SDLC output     DD93
Serial2 SDLC input      DD73
Serial2 SDLC output     DD11
Serial2 SDLC input      DD11
```

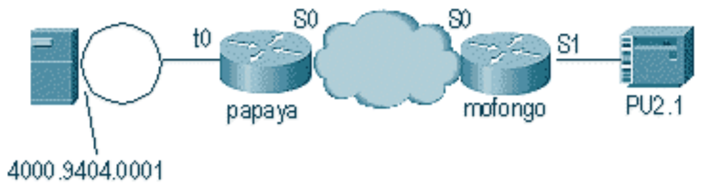
At this point, the connection suspends because of the steady Reciever Ready (RR) state between the router and the 5494.

Note: If the router on which you need to run the debug has other SDLC interfaces, and you are not logging buffered, the router can suspend. Understanding when you can run a debug to the terminal versus logging comes with experience. If you are not sure, always use logging buffered and the **show log** command to display SDLC debugs

Vary the controller off on the AS/400. This enables you to see DISC (53) and UA (73) that result on the SDLC side of the session.

```
Serial2 SDLC output      DD53
Serial2 SDLC input       DD73
```

DLSw Media Translation Example



After the interface comes up and up, the router begins the process by determining the location of the remote controller.

```
%LINEPROTO-5-UPDOWN: Line protocol on Interface Serial4, changed state to up
DLSW Received-ctlQ : CLSI Msg : ID_STN.Ind   dlen: 46
CSM: Received CLSI Msg : ID_STN.Ind   dlen: 46 from Serial4
CSM:   smac 4000.5494.00dd, dmac 4000.9404.0001, ssap 4 , dsap 4
%DLSWC-3-RECVSSP: SSP OP = 4( ICR  ) -explorer from peer 10.17.2.198(2065)
DLSw: new_ckt_from_clsi(): Serial4 4000.5494.00dd:4->4000.9404.0001:4
```

After receiving the ICR frame, DLSw starts the finite state machine (FSM) for this session. This is performed by the **REQ_OPNSTN.Req** and **REQ_OPNSTN.Cfm** messages that are between DLSw and Cisco Link Services Interface (CLSI).

```
DLSw: START-FSM (488636): event:DLC-Id state:DISCONNECTED
DLSw: core: dlsw_action_a()
DISP Sent : CLSI Msg : REQ_OPNSTN.Req   dlen: 106
DLSw: END-FSM (488636): state:DISCONNECTED->LOCAL_RESOLVE

DLSW Received-ctlQ : CLSI Msg : REQ_OPNSTN.Cfm CLS_OK dlen: 106
DLSw: START-FSM (488636): event:DLC-ReqOpnStn.Cnf state:LOCAL_RESOLVE
DLSw: core: dlsw_action_b()
CORE: Setting lf size to FF
```

After the conversation with CLSI, DLSw sends **session startup CUR** frames to the remote router. These occur between the two routers only.

```
%DLSWC-3-SENDSSP: SSP OP = 3( CUR   )  to peer 10.17.2.198(2065) success
DLSw: END-FSM (488636): state:LOCAL_RESOLVE->CKT_START

%DLSWC-3-RECVSSP: SSP OP = 4( ICR   )  from peer 10.17.2.198(2065)
DLSw: 488636 recv FCI 0 - s:0 so:0 r:0 ro:0
DLSw: recv RWO
DLSw: START-FSM (488636): event:WAN-ICR state:CKT_START
DLSw: core: dlsw_action_e()
DLSw: sent RWO
DLSw: 488636 sent FCI 80 on  ACK   - s:20 so:1 r:20 ro:1
%DLSWC-3-SENDSSP: SSP OP = 5( ACK   )  to peer 10.17.2.198(2065) success
DLSw: END-FSM (488636): state:CKT_START->CKT_ESTABLISHED
```

Once the circuit is established, the router sends the XID that was stored and starts the XID exchange. It is important to understand where the XIDs are. In this example, the data−link control (DLC)−Id means that the XID came from the local DLC station, and the WAN−XID came from the remote router, or remote station.

```
DLSw: START-FSM (488636): event:DLC-Id state:CKT_ESTABLISHED
DLSw: core: dlsw_action_f()
DLSw: 488636 sent FCA on  XID
%DLSWC-3-SENDSSP: SSP OP = 7( XID   )  to peer 10.17.2.198(2065) success
```

```
DLSw: END-FSM (488636): state:CKT_ESTABLISHED->CKT_ESTABLISHED

%DLSWC-3-RECVSSP: SSP OP = 7( XID  )  from peer 10.17.2.198(2065)
DLSw: 488636 recv FCA on  XID   - s:20 so:0 r:20 ro:0
DLSw: START-FSM (488636): event:WAN-XID state:CKT_ESTABLISHED
DLSw: core: dlsw_action_g()
DISP Sent : CLSI Msg : ID.Rsp   dlen: 12
DLSw: END-FSM (488636): state:CKT_ESTABLISHED->CKT_ESTABLISHED

%DLSWC-3-RECVSSP: SSP OP = 7( XID  )  from peer 10.17.2.198(2065)
DLSw: START-FSM (488636): event:WAN-XID state:CKT_ESTABLISHED
DLSw: core: dlsw_action_g()
DISP Sent : CLSI Msg : ID.Req   dlen: 88
DLSw: END-FSM (488636): state:CKT_ESTABLISHED->CKT_ESTABLISHED

DLSW Received-ctlQ : CLSI Msg : ID.Ind   dlen: 82
DLSw: START-FSM (488636): event:DLC-Id state:CKT_ESTABLISHED
DLSw: core: dlsw_action_f()
%DLSWC-3-SENDSSP: SSP OP = 7( XID  )  to peer 10.17.2.198(2065) success
DLSw: END-FSM (488636): state:CKT_ESTABLISHED->CKT_ESTABLISHED

%DLSWC-3-RECVSSP: SSP OP = 7( XID  )  from peer 10.17.2.198(2065)
DLSw: START-FSM (488636): event:WAN-XID state:CKT_ESTABLISHED
DLSw: core: dlsw_action_g()
DISP Sent : CLSI Msg : ID.Rsp   dlen: 88
DLSw: END-FSM (488636): state:CKT_ESTABLISHED->CKT_ESTABLISHED

DLSW Received-ctlQ : CLSI Msg : ID.Ind   dlen: 82
DLSw: START-FSM (488636): event:DLC-Id state:CKT_ESTABLISHED
DLSw: core: dlsw_action_f()
%DLSWC-3-SENDSSP: SSP OP = 7( XID  )  to peer 10.17.2.198(2065) success
DLSw: END-FSM (488636): state:CKT_ESTABLISHED->CKT_ESTABLISHED

%DLSWC-3-RECVSSP: SSP OP = 7( XID  )  from peer 10.17.2.198(2065)
DLSw: START-FSM (488636): event:WAN-XID state:CKT_ESTABLISHED
DLSw: core: dlsw_action_g()
DISP Sent : CLSI Msg : ID.Rsp   dlen: 88
DLSw: END-FSM (488636): state:CKT_ESTABLISHED->CKT_ESTABLISHED

DLSW Received-ctlQ : CLSI Msg : ID.Ind   dlen: 82
DLSw: START-FSM (488636): event:DLC-Id state:CKT_ESTABLISHED
DLSw: core: dlsw_action_f()
%DLSWC-3-SENDSSP: SSP OP = 7( XID  )  to peer 10.17.2.198(2065) success
DLSw: END-FSM (488636): state:CKT_ESTABLISHED->CKT_ESTABLISHED
```

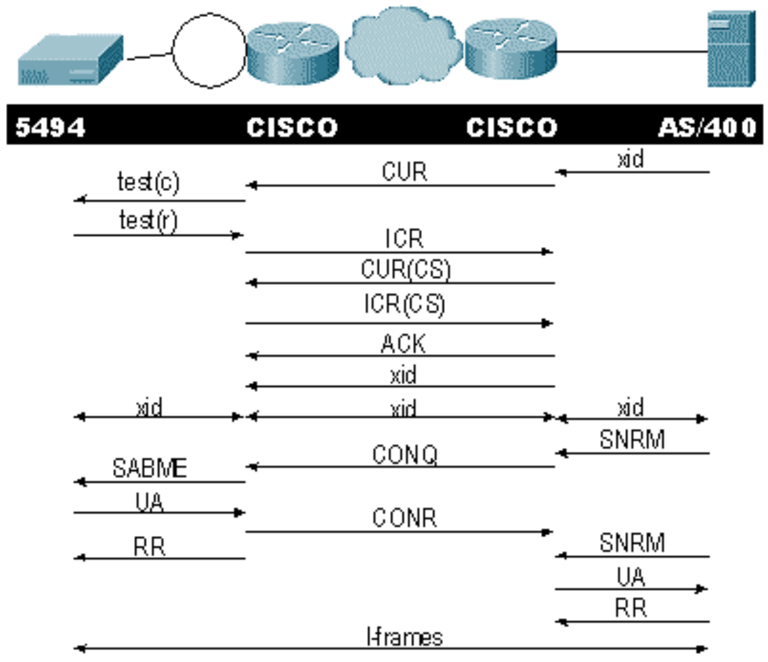
The router receives the **CONQ** from the AS/400 (SABME). This is translated to the serial line as an SNRM. Then the router waits for the UA on the serial line (**CONNECT.Cfm**), and sends the **CONR** to the other side. This changes the session state to **CONNECTED**.

```
%DLSWC-3-RECVSSP: SSP OP = 8( CONQ )  from peer 10.17.2.198(2065)
DLSw: START-FSM (488636): event:WAN-CONQ state:CKT_ESTABLISHED
DLSw: core: dlsw_action_i()
DISP Sent : CLSI Msg : CONNECT.Req   dlen: 16
DLSw: END-FSM (488636): state:CKT_ESTABLISHED->CONTACT_PENDING

DLSW Received-ctlQ : CLSI Msg : CONNECT.Cfm CLS_OK dlen: 8
DLSw: START-FSM (488636): event:DLC-Connect.Cnf state:CONTACT_PENDING
DLSw: core: dlsw_action_j()
%DLSWC-3-SENDSSP: SSP OP = 9( CONR )  to peer 10.17.2.198(2065) success
DISP Sent : CLSI Msg : FLOW.Req   dlen: 0
DLSw: END-FSM (488636): state:CONTACT_PENDING->CONNECTED
```

DLSw Performing Reverse Media Translation

Another common setup is **reverse−sdllc**. In reverse SDLLC, the primary station is attached via an SDLC line to the router. This is usually seen in host environments when users want to migrate the host to a Token Ring attachment. Reverse SDLLC changes the way DLSw handles the SDLC line because it is often not clear if the remote PU is active or not.

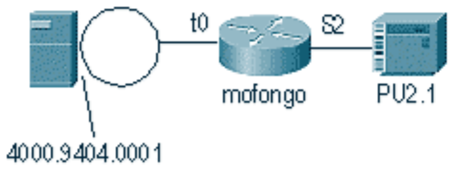


First, because the AS/400 is primary in this case, or set to be negotiable in the role, it needs to start the session. When the AS/400 sends the first XID after the serial line becomes operational, the router starts the search process for the remote controller. After the circuit is set up, XID negotiation can begin in the line.

When the XID negotiation finishes, the AS/400 sends SNRM the router. This causes the router to send the CONQ, and expect the CONR from the remote router. The router cannot respond with the UA until it sees an SNRM, and after it receives the CONR. In almost all versions of the code, the router waits 30 seconds until it times−out the session. This is in regards to receiving SNRMs from the primary device once the primary device receives the CONR from the remote host.

In the latest Cisco IOS 11.1 code, the defaults changed to one minute instead of 30 seconds. In the AS/400, this timeout is called the **non productive response timer** and defaults to 32 seconds.

Local DLSw Media Translation



```
%LINEPROTO−5−UPDOWN: Line protocol on Interface Serial2, changed state to up
%SYS−5−CONFIG_I: Configured from console by console
DLSW Received−ctlQ : CLSI Msg : ID_STN.Ind   dlen: 46
CSM: Received CLSI Msg : ID_STN.Ind   dlen: 46 from Serial2
```

The first thing you notice in DLSw local is the XID from the serial side. This XID needs to be stored until the router sends the LLC test frames/responses through.

```
CSM:   smac 4000.5494.00dd, dmac 4000.9404.0001, ssap 4 , dsap 4
DISP Sent : CLSI Msg : TEST_STN.Req   dlen: 46
DISP Sent : CLSI Msg : TEST_STN.Req   dlen: 46
DISP Sent : CLSI Msg : TEST_STN.Req   dlen: 46
CSM: Write to all peers not ok - PEER_NO_CONNECTIONS
DLSW Received-ctlQ : CLSI Msg : TEST_STN.Ind   dlen: 43
CSM: Received CLSI Msg : TEST_STN.Ind   dlen: 43 from TokenRing0
CSM:   smac c000.9404.0001, dmac 4000.5494.00dd, ssap 0 , dsap 4
```

Next, the test station leaves the router and the response returns from the AS/400. Now, the router can create the local FSM.

Note: Remember that this is a local session.

```
DLSw: csm_to_local(): Serial2-->TokenRing0  4000.5494.00dd:4->4000.9404.0001:4
DLSw: START-LFSM TokenRing0 (4000.9404.0001->4000.5494.00dd) event:ADMIN-START
DLSw: LFSM-A: Opening DLC station
DISP Sent : CLSI Msg : REQ_OPNSTN.Req   dlen: 106
DLSw: END-LFSM (4000.9404.0001->4000.5494.00dd): state:DISCONNECTED ->OPN_STN_PEND

DLSw: START-LFSM Serial2 (4000.5494.00dd->4000.9404.0001) event:ADMIN-START
DLSw: LFSM-A: Opening DLC station
DISP Sent : CLSI Msg : REQ_OPNSTN.Req   dlen: 106
DLSw: END-LFSM (4000.5494.00dd->4000.9404.0001): state:DISCONNECTED ->OPN_STN_PEND

DLSW Received-ctlQ : CLSI Msg : REQ_OPNSTN.Cfm CLS_OK dlen: 106
DLSw: START-LFSM TokenRing0 (4000.9404.0001->4000.5494.00dd) event:DLC-ReqOpnStn.Cnf
DLSw: LFSM-B: DLC station opened
DLSw: END-LFSM (4000.9404.0001->4000.5494.00dd): state:OPN_STN_PEND ->ESTABLISHED

DLSW Received-ctlQ : CLSI Msg : REQ_OPNSTN.Cfm CLS_OK dlen: 106
DLSw: START-LFSM Serial2 (4000.5494.00dd->4000.9404.0001) event:DLC-ReqOpnStn.Cnf
DLSw: LFSM-B: DLC station opened
DLSw: processing saved clsi message
```

After the router has confirmed locally that the FSM is ready, it can send the XID to the partner. In this example, the partner is the AS/400 (**ID.Req**).

```
DLSw: START-LFSM Serial2 (4000.5494.00dd->4000.9404.0001) event:DLC-Id
DLSw: LFSM-X: forward XID to partner
DISP Sent : CLSI Msg : ID.Req   dlen: 12
DLSw: END-LFSM (4000.5494.00dd->4000.9404.0001): state:ESTABLISHED  ->ESTABLISHED

DLSw: END-LFSM (4000.5494.00dd->4000.9404.0001): state:OPN_STN_PEND ->ESTABLISHED

DLSW Received-ctlQ : CLSI Msg : ID.Cfm CLS_OK dlen: 32
DLSw: START-LFSM TokenRing0 (4000.9404.0001->4000.5494.00dd) event:DLC-Id
DLSw: LFSM-X: forward XID to partner
DISP Sent : CLSI Msg : ID.Rsp   dlen: 12
DLSw: END-LFSM (4000.9404.0001->4000.5494.00dd): state:ESTABLISHED  ->ESTABLISHED
```

Then, an XID is received from the Token Ring. The **ID.Ind** has a length of 108. The router forwards this XID to the partner in this scenario, which is the SDLC line. This is indicated by the **ID.Req** that was sent. Each time the router receives a packet, it needs to start the linear finite state machine (LFSM). This is the key to understanding this debug, because it informs you where it starts and which points it is going.

```
DLSW Received-ctlQ : CLSI Msg : ID.Ind   dlen: 108
DLSw: START-LFSM TokenRing0 (4000.9404.0001->4000.5494.00dd) event:DLC-Id
DLSw: LFSM-X: forward XID to partner
```

```
DISP Sent : CLSI Msg : ID.Req   dlen: 88
DLSw: END-LFSM (4000.9404.0001->4000.5494.00dd): state:ESTABLISHED  ->ESTABLISHED
```

Next, the XID response is received from the serial line and is forwarded to the partner (the Token Ring station in this example). This continues until the XID exchange is finished for this PU2.1 device.

```
DLSW Received-ctlQ : CLSI Msg : ID.Ind   dlen: 82
DLSw: START-LFSM Serial2 (4000.5494.00dd->4000.9404.0001) event:DLC-Id
DLSw: LFSM-X: forward XID to partner
DISP Sent : CLSI Msg : ID.Rsp   dlen: 80
DLSw: END-LFSM (4000.5494.00dd->4000.9404.0001): state:ESTABLISHED  ->ESTABLISHED

DLSW Received-ctlQ : CLSI Msg : ID.Ind   dlen: 108
DLSw: START-LFSM TokenRing0 (4000.9404.0001->4000.5494.00dd) event:DLC-Id
DLSw: LFSM-X: forward XID to partner
DISP Sent : CLSI Msg : ID.Rsp   dlen: 88
DLSw: END-LFSM (4000.9404.0001->4000.5494.00dd): state:ESTABLISHED  ->ESTABLISHED

DLSW Received-ctlQ : CLSI Msg : ID.Ind   dlen: 82
DLSw: START-LFSM Serial2 (4000.5494.00dd->4000.9404.0001) event:DLC-Id
DLSw: LFSM-X: forward XID to partner
DISP Sent : CLSI Msg : ID.Rsp   dlen: 80
DLSw: END-LFSM (4000.5494.00dd->4000.9404.0001): state:ESTABLISHED  ->ESTABLISHED

DLSW Received-ctlQ : CLSI Msg : ID.Ind   dlen: 108
DLSw: START-LFSM TokenRing0 (4000.9404.0001->4000.5494.00dd) event:DLC-Id
DLSw: LFSM-X: forward XID to partner
DISP Sent : CLSI Msg : ID.Rsp   dlen: 88
DLSw: END-LFSM (4000.9404.0001->4000.5494.00dd): state:ESTABLISHED  ->ESTABLISHED

%LINK-3-UPDOWN: Interface Serial2, changed state to up
DLSW Received-ctlQ : CLSI Msg : ID.Ind   dlen: 82
DLSw: START-LFSM Serial2 (4000.5494.00dd->4000.9404.0001) event:DLC-Id
DLSw: LFSM-X: forward XID to partner
DISP Sent : CLSI Msg : ID.Rsp   dlen: 80
DLSw: END-LFSM (4000.5494.00dd->4000.9404.0001): state:ESTABLISHED  ->ESTABLISHED
```

After the XID exchange, the router receives an SABME from the AS/400 via the **CONNECT.Ind**. This tells the router to send a **CONNECT.Req** to the SDLC line, which is the SNRM. Then, a **CONNECT.Cfm** (UA) message is received from the serial line, which causes the DLSw code to send a **CONNECT.Rsp** (UA) to the AS/400.

```
DLSW Received-ctlQ : CLSI Msg : CONNECT.Ind   dlen: 8
DLSw: START-LFSM TokenRing0 (4000.9404.0001->4000.5494.00dd) event:DLC-Connect.Ind
DLSw: LFSM-C: starting local partner
DLSw: START-LFSM Serial2 (4000.5494.00dd->4000.9404.0001) event:ADMIN-CONN
DLSw: LFSM-D: sending connect request to station
DISP Sent : CLSI Msg : CONNECT.Req   dlen: 16
DLSw: END-LFSM (4000.5494.00dd->4000.9404.0001): state:ESTABLISHED  ->CONN_OUT_PEND

DLSw: END-LFSM (4000.9404.0001->4000.5494.00dd): state:ESTABLISHED  ->CONN_IN_PEND

DLSW Received-ctlQ : CLSI Msg : CONNECT.Cfm CLS_OK dlen: 8
DLSw: START-LFSM Serial2 (4000.5494.00dd->4000.9404.0001) event:DLC-Connect.Cnf
DLSw: LFSM-E: station accepted the connection
DLSw: START-LFSM TokenRing0 (4000.9404.0001->4000.5494.00dd) event:ADMIN-CONN
DLSw: LFSM-F: accept incoming connection
DISP Sent : CLSI Msg : CONNECT.Rsp   dlen: 20
DLSw: END-LFSM (4000.9404.0001->4000.5494.00dd): state:CONN_IN_PEND ->CONNECTED

DISP Sent : CLSI Msg : FLOW.Req   dlen: 0
DLSw: END-LFSM (4000.5494.00dd->4000.9404.0001): state:CONN_OUT_PEND->CONNECTED
```

The session when the controller (SDLC) shuts down is displayed.

```
%LINEPROTO−5−UPDOWN: Line protocol on Interface Serial2, changed state to down
%LINK−5−CHANGED: Interface Serial2, changed state to administratively down
DLSW Received−ctlQ : CLSI Msg : DISCONNECT.Ind   dlen: 8
DLSw: START−LFSM Serial2 (4000.5494.00dd−>4000.9404.0001) event:DLC−Disc.Ind
DLSw: LFSM−Q: acknowledge disconnect
DISP Sent : CLSI Msg : DISCONNECT.Rsp   dlen: 4
```

Next, the router sends a DISC to the AS/400 (DISCONNECT.Rsp). Then, it starts tearing down the local circuit.

```
DLSw: START−LFSM TokenRing0 (4000.9404.0001−>4000.5494.00dd) event:ADMIN−STOP
DLSw: LFSM−Z: close dlc station request
DISP Sent : CLSI Msg : CLOSE_STN.Req   dlen: 4
DLSw: END−LFSM (4000.9404.0001−>4000.5494.00dd): state:ESTABLISHED  −>CLOSE_PEND

DISP Sent : CLSI Msg : CLOSE_STN.Req   dlen: 4
DLSw: END−LFSM (4000.5494.00dd−>4000.9404.0001): state:ESTABLISHED  −>CLOSE_PEND

DLSW Received−ctlQ : CLSI Msg : CLOSE_STN.Cfm CLS_OK dlen: 8
DLSw: START−LFSM TokenRing0 (4000.9404.0001−>4000.5494.00dd) event:DLC−CloseStn.Cnf
DLSw: LFSM−Y: driving partner to close circuit
DLSw: START−LFSM Serial2 (4000.5494.00dd−>4000.9404.0001) event:ADMIN−STOP
DLSw: END−LFSM (4000.5494.00dd−>4000.9404.0001): state:CLOSE_PEND   −>CLOSE_PEND

DLSw: END−LFSM (4000.9404.0001−>4000.5494.00dd): state:CLOSE_PEND   −>DISCONNECTED

DLSW Received−ctlQ : CLSI Msg : DISCONNECT.Ind   dlen: 8
DLSw: START−LFSM Serial2 (4000.5494.00dd−>4000.9404.0001) event:DLC−Disc.Ind
DLSw: END−LFSM (4000.5494.00dd−>4000.9404.0001): state:CLOSE_PEND   −>CLOSE_PEND

DLSW Received−ctlQ : CLSI Msg : CLOSE_STN.Cfm CLS_OK dlen: 8
DLSw: START−LFSM Serial2 (4000.5494.00dd−>4000.9404.0001) event:DLC−CloseStn.Cnf
DLSw: LFSM−Y: removing local switch entity
DLSw: END−LFSM (4000.5494.00dd−>4000.9404.0001): state:CLOSE_PEND   −>DISCONNECTED
```

After the router receives the **DISCONNECT.Ind** (UA) from the AS/400, it finishes clearing the session and moves to a disconnect state.

Related Information

- **IBM Technologies**
- **Technical Support & Documentation − Cisco Systems**