NAME llc1 – Logical Link Control Protocol Class 1 Driver

SYNOPSIS
#include <sys/stropts.h>
#include <sys/ethernet.h>
#include <sys/dlpi.h>
#include <sys/llc1.h>

AVAILABILITY x86

DESCRIPTION

The **llc1** driver is a multi-threaded, loadable, clonable, STREAMS multiplexing driver supporting the connectionless Data Link Provider Interface, **dlpi**(7P), implementing IEEE 802.2 Logical Link Control Protocol Class 1 over a STREAM to a MAC level driver. Multiple MAC level interfaces installed within the system can be supported by the driver. The **llc1** driver provides basic support for the LLC1 protocol. Functions provided include frame transmit and receive, XID, and TEST, multicast support, and error recovery and reporting.

The cloning, character-special device, **/dev/llc1**, is used to access all LLC1 controllers configured under **llc1**.

The **llc1** driver is a "Style 2" Data Link Service provider. All messages of types **M_PROTO** and **M_PCPROTO** are interpreted as DLPI primitives. An explicit **DL_ATTACH_REQ** message by the user is required to associate the opened stream with a particular device (**ppa**). The **ppa** ID is interpreted as an unsigned long and indicates the corresponding device instance (unit) number. An error (**DL_ERROR_ACK**) is returned by the driver if the **ppa** field value does not correspond to a valid device instance number for this system.

The values returned by the driver in the **DL_INFO_ACK** primitive in response to the **DL_INFO_REQ** from the user are as follows:

> The maximum Service Data UNIT (SDU) is derived from the MAC layer linked below the driver. In the case of an Ethernet driver, the SDU will be 1497.
>
> The minimum SDU is 0.
>
> The **dlsap** address length is 7.
>
> The MAC type is **DL_CSMACD** or **DL_TPR** as determined by the driver linked under **llc1.** If the driver reports that it is **DL_ETHER,** it will be changed to **DL_CSMACD;** otherwise the type is the same as the MAC type.
>
> The **sap** length value is –1, meaning the physical address component is followed immediately by a 1-octet **sap** component within the DLSAP address.
>
> The service mode is **DL_CLDLS.**
>
> No optional quality of service (QOS) support is included at present, so the QOS fields should be initialized to 0.
>
> The provider style is **DL_STYLE2.**
>
> The DLPI version is **DL_VERSION_2.**

The broadcast address value is the broadcast address returned from the lower level driver.

Once in the **DL_ATTACHED** state, the user must send a **DL_BIND_REQ** to associate a particular Service Access Point (SAP) with the stream. The **llc1** driver interprets the **sap** field within the **DL_BIND_REQ** as an IEEE 802.2 “SAP,” therefore valid values for the **sap** field are in the [0-0xFF] range with only even values being legal.

The **llc1** driver DLSAP address format consists of the 6-octet physical (e.g., Ethernet) address component followed immediately by the 1-octet **sap** (type) component producing a 7-octet DLSAP address. Applications should *not* hard-code to this particular implementation-specific DLSAP address format, but use information returned in the **DL_INFO_ACK** primitive to compose and decompose DLSAP addresses. The **sap** length, full DLSAP length, and **sap**/physical ordering are included within the **DL_INFO_ACK.** The physical address length can be computed by subtracting the absolute value of the **sap** length from the full DLSAP address length or by issuing the **DL_PHYS_ADDR_REQ** to obtain the current physical address associated with the stream.

Once in the **DL_BOUND** state, the user may transmit frames on the LAN by sending **DL_UNITDATA_REQ** messages to the **llc1** driver. The **llc1** driver will route received frames up all open and bound streams having a **sap** which matches the IEEE 802.2 DSAP as **DL_UNITDATA_IND** messages. Received frames are duplicated and routed up multiple open streams if necessary. The DLSAP address contained within the **DL_UNITDATA_REQ** and **DL_UNITDATA_IND** messages consists of both the **sap** (type) and physical (Ethernet) components.

In addition to the mandatory, connectionless DLPI message set, the driver additionally supports the following primitives:

The **DL_ENABMULTI_REQ** and **DL_DISABMULTI_REQ** primitives enable/disable reception of specific multicast group addresses. A set of multicast addresses may be iteratively created and modified on a per-stream basis using these primitives. These primitives are accepted by the driver in any driver state that is valid while still being attached to the **ppa**.

The **DL_PHYS_ADDR_REQ** primitive returns the 6-octet physical address currently associated (attached) to the stream in the **DL_PHYS_ADDR_ACK** primitive. This primitive is valid only in states following a successful **DL_ATTACH_REQ.**

The **DL_SET_PHYS_ADDR_REQ** primitive changes the 6-octet physical address currently associated (attached) to this stream. Once changed, all streams subsequently opened and attached to this device will obtain this new physical address. Once changed, the physical address will remain set until this primitive is used to change the physical address again or the system is rebooted, whichever occurs first.

The **DL_XID_REQ/DL_TEST_REQ** primitives provide the means for a user to issue an **LLC XID** or **TEST** request message. A response to one of these messages will be in the form of a **DL_XID_CON/DL_TEST_CON** message.

The **DL_XID_RES/DL_TEST_RES** primitives provide a way for the user to respond to the receipt of an **XID** or **TEST** message that was received as a **DL_XID_IND/DL_TEST_IND** message.

XID and **TEST** will be automatically processed by **llc1** if the **DL_AUTO_XID/DL_AUTO_TEST** bits are set in the **DL_BIND_REQ.**

FILES

/dev/llc1

SEE ALSO

dlpi(7P)