Infrared Data Association

Link Management Protocol

Version 1.1

23rd January 1996

Authors:

Andy Seaborne, Stuart Williams (Hewlett-Packard Company)
Frank Novak (IBM Corporation)

Editors:

Iain Millar, Stuart Williams (Hewlett-Packard Company)

Significant Contributors:

Dave Suvak, Rick Pennington (Hewlett-Packard Company)
Tim Williams (IBM Corporation)

Document Status: Version 1.1

Major changes from Version 1.0:

- Incorporate all the version 1.0 errata into the specification.
- Document accepted by the Technical Committee

INFRARED DATA ASSOCIATION (IrDA) - NOTICE TO THE TRADE -

SUMMARY:

Following is the notice of conditions and understandings upon which this document is made available to members and non-members of the Infrared Data Association.

- Availability of Publications, Updates and Notices
- Full Copyright Claims Must be Honored
- Controlled Distribution Privileges for IrDA Members Only
- Trademarks of IrDA - Prohibitions and Authorized Use
- No Representation of Third Part Rights
- Limitation of Liability
- Disclaimer of Warranty
- Certification of Products Requires Specific Authorization from IrDA after Product Testing for IrDA Specification Conformance

IrDA PUBLICATIONS and UPDATES:

Single issues of each IrDA publication, including notifications, updates, and revisions, are distributed to IrDA members in good standing during the course of each year as a benefit of annual IrDA membership. Additional copies are available to IrDA members for a fee which covers the cost of reproduction and distribution. Annual subscriptions of IrDA publications are available to non-IrDA members for a pre-paid fee. Requests for publications, membership applications or more information should be addressed to: Infrared Data Association, P.O. Box 3883, Walnut Creek, California, U.S.A. 94598; or e-mail address: jlaroche@netcom.com; or by calling John LaRoche at (510) 943-6546 or faxing requests to (510) 934-5241.

COPYRIGHT:

1. Prohibitions: IrDA claims copyright in all IrDA publications. Any unauthorized reproduction, distribution, display or modification, in whole or in part, is strictly prohibited.

2. Authorized Use: Any authorized use of IrDA publications (in whole or in part) is under NONEXCLUSIVE USE LICENSE ONLY. No rights to sublicense, assign or transfer the license are granted and any attempt to do so is void.

DISTRIBUTION PRIVILEGES for IrDA MEMBERS ONLY:

IrDA Members Limited Reproduction and Distribution Privilege: A limited privilege of reproduction and distribution of IrDA copyrighted publications is granted to IrDA members in good standing and for sole purpose of reasonable reproduction and distribution to non-IrDA members who are engaged by contract with an IrDA member for the development of IrDA certified products. Reproduction and distribution by the non-IrDA member is strictly prohibited.

TRANSACTION NOTICE to IrDA MEMBERS ONLY:

Each and every copy made for distribution under the limited reproduction and distribution privilege shall be conspicuously marked with the name of the IrDA member and the name of the receiving party. Upon reproduction for distribution, the distributing IrDA member shall promptly notify IrDA (in writing or by e-mail) of the identity of the receiving party.

A failure to comply with the notification requirement to IrDA shall render the reproduction and distribution unauthorized and IrDA may take appropriate action to enforce its copyright, including but not limited to, the termination of the limited reproduction and distribution privilege and IrDA membership of the non-complying member.

TRADEMARKS:

1. Prohibitions: IrDA claims exclusive rights in its trade names, trademarks, service marks, collective membership marks and certification marks (hereinafter collectively "trademarks"), including but not limited to the following trademarks: INFRARED DATA ASSOCIATION (wordmark alone and with IR logo), IrDA (acronym mark alone and with IR logo), IR logo, IR DATA CERTIFIED (composite mark), and MEMBER IrDA (wordmark alone and with IR logo). Any unauthorized use of IrDA trademarks is strictly prohibited.

2. Authorized Use: Any authorized use of a IrDA collective membership mark or certification mark is by NONEXCLUSIVE USE LICENSE ONLY. No rights to sublicense, assign or transfer the license are granted and any attempt to do so is void.

NO REPRESENTATION of THIRD PARTY RIGHTS:

IrDA makes no representation or warranty whatsoever with regard to IrDA member or third party ownership, licensing or infringement/non-infringement of intellectual property rights. Each recipient of IrDA publications, whether or not an IrDA member, should seek the independent advice of legal counsel with regard to any possible violation of third party rights arising out of the use, attempted use, reproduction, distribution or public display of IrDA publications.

IrDA assumes no obligation or responsibility whatsoever to advise its members or non-members who receive or are about to receive IrDA publications of the chance of infringement or violation of any right of an IrDA member or third party arising out of the use, attempted use, reproduction, distribution or display of IrDA publications.

LIMITATION of LIABILITY:

BY ANY ACTUAL OR ATTEMPTED USE, REPRODUCTION, DISTRIBUTION OR PUBLIC DISPLAY OF ANY IrDA PUBLICATION, ANY PARTICIPANT IN SUCH REAL OR ATTEMPTED ACTS, WHETHER OR NOT A MEMBER OF IrDA, AGREES TO ASSUME ANY AND ALL RISK ASSOCIATED WITH SUCH ACTS, INCLUDING BUT NOT LIMITED TO LOST PROFITS, LOST SAVINGS, OR OTHER CONSEQUENTIAL, SPECIAL, INCIDENTAL OR PUNITIVE DAMAGES. IrDA SHALL HAVE NO LIABILITY WHATSOEVER FOR SUCH ACTS NOR FOR THE CONTENT, ACCURACY OR LEVEL OF ISSUE OF AN IrDA PUBLICATION.

DISCLAIMER of WARRANTY:

All IrDA publications are provided "AS IS" and without warranty of any kind. IrDA (and each of its members, wholly and collectively, hereinafter "IrDA") EXPRESSLY DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE AND WARRANTY OF NON-INFRINGEMENT OF INTELLECTUAL PROPERTY RIGHTS. IrDA DOES NOT WARRANT THAT ITS PUBLICATIONS WILL MEET YOUR REQUIREMENTS OR THAT ANY USE OF A PUBLICATION WILL BE UN-INTERRUPTED OR ERROR FREE, OR THAT DEFECTS WILL BE CORRECTED. FURTHERMORE, IrDA DOES NOT WARRANT OR MAKE ANY REPRESENTATIONS REGARDING USE OR THE RESULTS OR THE USE OF IrDA PUBLICATIONS IN TERMS OF THEIR CORRECTNESS, ACCURACY, RELIABILITY, OR OTHERWISE. NO ORAL OR WRITTEN PUBLICATION OR ADVICE OF A REPRESENTATIVE (OR MEMBER) OF IrDA SHALL CREATE A WARRANTY OR IN ANY WAY INCREASE THE SCOPE OF THIS WARRANTY.

LIMITED MEDIA WARRANTY:

IrDA warrants ONLY the media upon which any publication is recorded to be free from defects in materials and workmanship under normal use for a period of ninety (90) days from the date of distribution as evidenced by the distribution records of IrDA. IrDA's entire liability and recipient's exclusive remedy will be replacement of the media not meeting this limited warranty and which is returned to IrDA. IrDA shall have no responsibility to replace media damaged by accident, abuse or misapplication. ANY IMPLIED WARRANTIES ON THE MEDIA, INCLUDING THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, ARE LIMITED IN DURATION TO NINETY (90) DAYS FROM THE DATE OF DELIVERY. THIS WARRANTY GIVES YOU SPECIFIC LEGAL RIGHTS, AND YOU MAY ALSO HAVE OTHER RIGHTS WHICH VARY FROM PLACE TO PLACE.

CERTIFICATION and GENERAL:

Membership in IrDA or use of IrDA publications does NOT constitute IrDA certification of products. It is the sole responsibility of each manufacturer, whether or not an IrDA member, to obtain certification of products in accordance with IrDA rules for certification.

All rights, prohibitions of right, agreements and terms and conditions regarding use of IrDA publications and IrDA rules for certification of products are governed by the laws and regulations of the United States. However, each manufacturer is solely responsible for compliance with the import/export laws of the countries in which they conduct business. The information contained in this document is provided as is and is subject to change without notice.

Contents

1. Introduction

1.1 Purpose

The Link Management Protocol is part of a standard for IrDA devices that supports walk-up, ad hoc connection between IrDA devices. Software on one device can discover the services available on other devices. The protocol provides support for multiple software applications/entities to operate independently and concurrently, sharing the single link provided by the IrDA Link Access Protocol (IrLAP) between the primary device and each secondary device.

This involves several things; discovery, multiplexing the link, and controlling the link.

Discovery entails each IrDA device maintaining an information base of services that the device currently has available. These services are modeled as objects with attributes that describe the object. This information may be queried from another device.

Multiplexing the link enables independent entities to exchange data over a single IrLAP link.

Link Control involves managing the use of the multiplexed link, including provision for clients that want exclusive control of the IrLAP link connection.

1.2 Scope

This specification is one of a family of specifications intended to facilitate the interconnection of electronic devices using a directed half duplex serial infrared physical communications medium such as that provided by the IrDA serial infrared physical layer.

This specification describes the functions, features, protocol and services for interconnection between Link Management peers. Only the definition of the protocol will be discussed in this specification. No attempt is made to document how the IrDA Link Management protocol should be implemented.

The Link Management protocol will be referred to hereafter as IrLMP.

IrLMP constitutes one piece of the required IrDA protocol stack. It uses services provided by the data-link layer (IrLAP) and provides services to clients above (i.e., transport entities and/or applications).

Elements of the protocol are covered in the following sections:

- Section 2 provides an overview of IrLMP.
- Section 3 provides a detailed look at the Link Management Multiplexer including its internal organization, the services it provides, frame formats, state machines, and state event tables.
- Section 4 outlines the Link Management Information Access Service including its internal organization, the services it provides, frame formats, state machines, and state event tables.
- Section 5 describes the objects and attributes that must be supported for IrDA compliance.

- Section 6 details what is and is not required for a minimal IrLMP implementation.
- Section 7 lists several examples that illustrate how Link Management functions.

1.3 References

[IRLAP] Infrared Data Association, “Serial Infrared Link Access Protocol (IrLAP)”, Version 1.0, Version 1.0, April 27, 1994.

[ISO8859] International Standardisation Organisation (ISO), “Information Processing - 8-bit single-byte coded graphic character sets”, ISO/IEC 8859-1 to 10, 1987-1992.

[IRPNP] Infrared Data Association, “Plug and Play Extensions to Link Management Protocol”, Version 1.0, September 30, 1994.

[TINYTP] Infrared Data Association, “TinyTP: A Flow-Control Mechanism for use within IrLMP”, Version 0.1b, March 22, 1995.

1.4 Acronyms and Definitions

Attribute The pairing of an attribute name and an attribute value.

Attribute Name An identifier that conveys the semantics associated with an attribute value, scoped by the class of the object containing the attribute.

Attribute Value The value associated with a named attribute.

Attribute Value Type An identifier that precedes some attribute values. It identifies the syntax used to express an attribute value.

Class Name An identifier that distinguishes between object classes.

Connectionless LSAP-SEL This is the reserved LSAP-SEL (0x70). All connectionless data packets are delivered to the LSAP associated with this LSAP-SEL.

Device Address The IrLAP device address of a station. This is a 32-bit identifier that is randomly selected by a station. It is expected to be relatively static between successive initializations of the IrLAP communication services. However, it may change between successive initializations when duplicate use of the same device address is detected during the XID discovery process.

Device Object Class An IrDA-defined object class used to store attributes related to the physical device rather than those related to the services that the device supports.

Information Base	A collection of object instances within a station.
IrLAP	An acronym for the IrDA-defined data-link layer protocol; Infrared Link Access Protocol
IrLAP-Connection	At any given instant there may be at most one IrLAP-Connection between a given pair of stations. One of the stations must assume the IrLAP primary role while the other must assume the IrLAP secondary role.
IrLAP-Connection Address	During the lifetime of an IrLAP-Connection it is referred to by an IrLAP-Connection address in the range 0x01-0x7E (0x00 and 0x7F are not used for connections). This address is assigned by the station that assumes the primary role. The IrLAP-Connection address also services as an abbreviation for the primary and secondary device addresses of the stations involved in an IrLAP-Connection.
IrLAP-Connection Endpoint	Each end of an IrLAP-Connection terminates at an IrLAP-Connection Endpoint. An IrLAP-Connection Endpoint also serves as a local reference to the IrLAP-Connection that it terminates. An IrLAP-Connection Endpoint is identified solely by the IrLAP-Connection Address.
IrLAP Primary	The station involved in IrDA communications that assumes responsibility for the organization of data flow and for unrecoverable data link error conditions.
IrLAP Secondary	Any station having an active IrLAP connection and not assuming the IrLAP Primary role.
LSAP	An acronym for Link Service Access Point. A collection of LSAP-Connection Endpoints within the same station that share a common valued LSAP-SEL are grouped at an LSAP.
LSAP Address	See LSAP-ID
LSAP-Connection	The communication channel provided by Link Management between two LSAPs is referred to as an LSAP-Connection. There may be at most one LSAP-Connection between the same pair of LSAPs. However, a single LSAP may contain several LSAP-Connection Endpoints. An LSAP-Connection is uniquely identified by the unordered pairing of the LSAP-Address at each end of the connection. For example, if A and B are the LSAP-Addresses for each end of an LSAP-Connection, then <A,B> and <B,A> both

	identify the same connection. All LSAP-Connections require an IrLAP link connection to exist with two exceptions; services offered at Connectionless LSAP-SEL and intra-station connections (which have an implied link and do not require a link to be established explicitly).
LSAP-Connection Endpoint	Each end of an LSAP-Connection terminates at an LSAP-Connection Endpoint. All Link Management Multiplexer service primitives (except Discovery and Connectionless) are invoked at an LSAP-Connection Endpoint. An LSAP-Connection Endpoint also serves as a local reference to the LSAP-Connection that it terminates.
LSAP-ID	An LSAP-ID identifies a particular LSAP at a particular station and is represented as the concatenation of <DeviceAddress> and <LSAP-SEL>.
LSAP-SEL	An acronym for LSAP selector. A selector that distinguishes between LSAPs within a Station. Legal values for an LSAP-SEL lie in the range 0x00-0x7F. With the exception of the special LSAP-SEL values 0x00 (LM-IAS), 0x70 (Connectionless Data service), 0x71-0x7E (reserved), and 0x7F (reserved for broadcast and currently not implemented), the assignment of LSAP-SEL values is arbitrary.
Object Class	Defines the semantics of a collection of attributes that will be held by object instances of that class. Typically an object class will be defined to carry the attributes necessary to make contact with and use a particular class of Link Management client.
Object ID	An identifier that distinguishes between object instances within an information base.
Object Instance	Each object instance within an information base is assigned a class name and an object identifier. Each object instance contains between 0 and 255 attributes. The semantics of the attributes are defined by the object class associated with the object's class name.
Station	A station is the logical endpoint of an IrLAP communication channel. A station also represents the sum of the services offered by Link Management at the endpoint. A synonym for IrLAP-connection endpoint.

1.5 Byte Ordering

This document represents IrLMP frames as collections of octets (bytes) with each octet being composed of 8 bits numbered 0-7. Bit 0 is always the least significant bit (LSB) and bit 7 is always the most significant bit (MSB). In some cases IrLMP frames contain components that are composed of multiple bytes. These larger components are represented as n*8 bits where n is the number of bytes in the component. The least significant bit is numbered bit 0 while the most significant bit is numbered (n*8)-1. The least significant byte of a multi-byte component is defined to be the byte that contains bits 0-7. Bytes are represented throughout this document in the following forms:

- Diagrammatic - a byte is represented as a rectangle. In some cases bit fields have special meaning and are indicated for clarity. The most significant bit is the bit on the left and the least significant bit is the bit on the right. An example is given below

7	6	5	4	3	2	1	0
C	DLSAP-SEL						

- Hexadecimal - a byte is represented with two hex digits with the least signification nibble on the right, the most significant nibble on the left, and both digits preceded by 0x. An example is the value 5 which is written as 0x05.

- Tabular - a byte is represented by a table with each row of the table corresponding to a bit. The least significant bit occupies the first row of the table and the most significant bit occupies the last row of the table. An example is given below.

Byte 1	
Bit	Meaning
0	PDA/Palmtop
1	Computer
2	Printer
3	Modem
4	Fax
5	Telephony
6	LAN Access
7	Extension

1.6 State Machine Rules

In sections 3 and 4 of this document precise descriptions of the IrLMP procedures are specified using state transition diagrams and state transition tables. Textual descriptions are added to aid the first-time reader. The following notes apply to all state transition diagrams and transition tables listed in this specification:

- The state transition diagrams and transition tables are included in the precise description of operation are supplied in order to clearly specify the behavior of the protocol. Designers and implementor may choose any design/implementation technique they wish, provided the resulting external behavior is identical to the external behavior of the specified state machines.

- Events not recognized in a particular state are assumed to remain pending (i.e., they get queued) until any masking flag is modified or a state transition occurs that allows the event to be recognized.

- Any inconsistency between a state transition diagram, a state transition table, and any textual description should be resolved by using the state transition table.

Predicate and Set Notation

A number of actions and predicates in the in the Actions and Events columns of the State Transition diagrams make use of set variables. The following set operations are commonly used:

$\cup$ Set Union
$\cap$ Set Intersection
- Removal the intersection with another set eg: A = A - B removes from A those members that A has common with B.

| | Set Modulus, ie. |A| is the number of members of set A.

$\wedge$ Logical ANDing of predicates
$\vee$ Logical Oring of predicates.

$\subseteq$ Strict subset predicate: $A \subseteq B$ is true if A is a strict subset of B
$\in$ Set membership predicate: $a \in A$ is true if a is an element of set A.

$\forall$ Iteration: $\forall a \in A$ <someAction>. Repeat <someAction> for each member of set A.

2. Link Management Overview

2.1 Description

The following subsections describe what Link Management is, what it does, and how it fits into the overall IrDA architecture. They are only intended to provide an overview. A more detailed description of Link Management and its components is given in sections 3 and 4 of this specification.

2.2 Architectural Components

The overall IrDA Protocol Architecture is shown in Figure 1. Link Management defines two components within this architecture: the Link Management Information Access Service (LM-IAS) and the Link Management Multiplexer (LM-MUX).

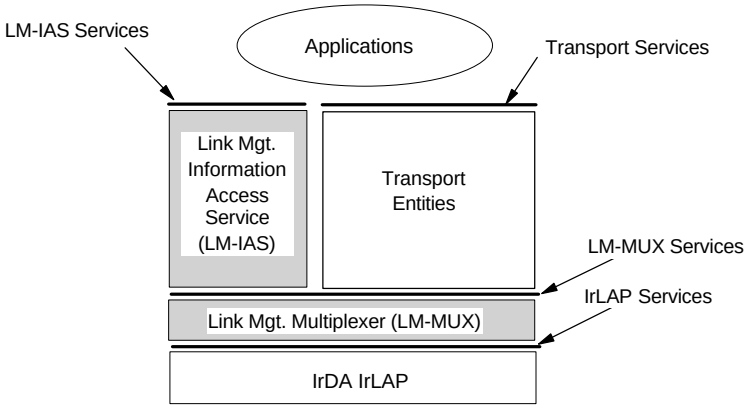


Figure 1. Link Management in IrDA Architecture

2.2.1 Information Access Service

Each LM-IAS entity maintains an information base so that one IrDA device can discover what services another IrDA compliant device offers as well as finding out about the device itself. This information is held in a number of objects in the information base.

The information model defines the external conceptual view of the information held by an LM-IAS entity. It defines the operations used to access the information and the format of transmitted data. This specification does not dictate the internal organization of an implementation that is used to meet this specification nor does this specification describe how information is registered with the local LM-IAS entity.

The LM-IAS entity does not control or mandate the information held in the information base except for the “Device” object as described in section 5.

2.2.2 Link Management Multiplexer

The LM-MUX provides services to both the local LM-IAS entity and to transport entities or applications that bind to the LM-MUX. IrLAP provides a reliable connection between a pair of IrDA devices. The LM-MUX provides multiple data link connections over IrLAP.

2.2.3 Transport Protocol

A transport protocol provides one or more connections between applications. This should be contrasted with IrLAP which provides a single connection between stations.

A transport entity may be embedded within an application or may be distinct and provide services to a number of applications concurrently. All references to transport protocol in this document should be taken to include both of the examples cited.

2.2.4 Applications

Applications are the entities that provide user oriented functionality. There are two typical ways in which applications use and provide services. They either use another software entity as transport, and hence do not use the multiplexer facilities directly, or they incorporate the transport functionality within themselves and directly use the facilities of the multiplexer.

2.3 Service Interfaces

Service primitives are provided at two points: between the LM-IAS entity and application software, and by the multiplexer between the multiplexer and its clients. These external service primitives are provided by IrDA devices that express the interaction between two devices across the infra-red medium.

2.3.1 Information Access Service Interface

The LM-IAS provides a mechanism for exchanging information about participating devices and the services they offer.

2.3.2 Link Management Multiplexer Service Interface

The LM-MUX provides services to both Link Management itself and to its service users at Link Service Access Points (LSAPs). IrLAP provides a reliable connection between a pair of IrDA devices. The LM-MUX provides a mechanism to enable multiple data connections over IrLAP, as well as sharing control of the single IrLAP connection between a pair of stations.

2.3.3 Transport Service Interface

Services provided by transport protocols should be detailed in the appropriate transport protocol specification. Link Management does not control which transport services are directly exposed to clients.

2.3.4 IrLAP Service Interface

Services provided by the IrDA Link Access Protocol are outlined in [IRLAP]. These services are not directly exposed. The LM-MUX is the sole user of the services provided by IrLAP.

2.4 Link Model

Multiplexing allows more than one protocol entity to use the single IrLAP link connection between any two IrDA devices independently and concurrently with other protocol entities.

The LM-MUX can be in one of two modes, multiplexed or exclusive. When in multiplexed mode, several LSAP connections may actively use the underlying IrLAP connection. When in exclusive mode, just one LSAP connection may be active on the IrLAP connection.

2.4.1 Multiplexed Mode

The IrLMP multiplexer, LM-MUX, provides multiple independent LSAP-connections per IrLAP connection. These LSAP connections may be used by individual clients of the LM-MUX within the two devices connected by the IrLAP connection. The LM-MUX relieves the client entities of the requirement to coordinate access to the single IrLAP connection. However, the LM-MUX DOES NOT provide a per LSAP connection flow control which results in the need for LM-MUX clients to be aware of possible dead-lock problems.

Since IrDA Link Access Protocol (IrLAP) supports a single IrLAP-connection between any given pair of devices, there is a single flow controlled channel between the devices at the IrLAP level. However, IrLAP flow-control is NOT a suitable mechanism for the provision of flow control to multiple channels multiplexed on top of an IrLAP-connection. If, by not consuming incoming frames, an LM-MUX client were to cause the underlying IrLAP flow-control mechanism to flow control off (so-called back-pressure), all the LSAP-connections multiplexed on top of the IrLAP connection would be halted (in the direction toward the device applying flow-control). This can cause dead-locks in some circumstances.

The only situation where it is acceptable to use IrLAP flow control as a means to directly flow-control an LSAP connection is when it is only possible to open ONE LSAP connection over and above the connection used to access the Information Access Services, see section 4.

In multiplexed mode with multiple clients, an LM-MUX entity requires that those clients accept any incoming frame (I frame or UI frame). If the LM-MUX client is not able to accept the data presented, the LM-MUX will discard the data. In order to provide reliable data delivery, it is the responsibility of the LM-MUX client to ensure either:

- that this does not arise by implementing an appropriate flow control mechanism on top of the link protocol, e.g. TinyTP [TINYTP], or

- implement an appropriate scheme to retransmit any lost data frames.

2.4.2 Exclusive Mode

Some protocols and applications may require special control of an LSAP-connection in order to:

- achieve a reduced, dependable latency;
- control the link turnaround through their use of the link.

Alternatively a single service that needs only one application-to-application connection may wish to depend on IrLAP flow-control rather than incur the overhead of a flow-controlled transport protocol.

These uses are accommodated by letting an application or transport protocol request exclusive use of the IrLAP connection.

Although exclusive mode is established on behalf of an established LSAP-connection. Unreliable datagrams may be sent down such a connection using the LM_UData service.

In exclusive mode, since there is only a single LSAP-connection being serviced, flow-control on the LSAP-connection may depend upon IrLAP flow-control. Hence, in exclusive mode reliably delivered I frames are buffered within the receiving LM-MUX until the intended LM-MUX client is capable of receiving them. However, since UI frames are not subject to flow-control they are handled in the same way as they are in multiplexed mode. i.e. they are NOT held indefinitely within the receiving LM-MUX and must be accepted when delivered to an LM-MUX client.

Whilst in exclusived mode the LM-MUX discards outgoing connectionless data sent using the LM_ConnectionlessData.request service primitive. The corresponding LM_Connectionless.confirm primitive indicates that discard has occurred since the LM-MUX is in exclusive mode.

3. Link Management Multiplexer

3.1 Introduction

The following subsections describe the framework used in this document to outline the operation of the LM-MUX. It is not necessary that an implementation of the LM-MUX is structured according to the same framework. However, any implementation of an LM-MUX is required to exhibit identical external behavior (at its upper and lower service boundaries) as that described by this specification.

3.1.1 External Interfaces

Figure 2 identifies the three types of Service Access Points provided at the upper LM-MUX service boundary and the IrLAP Service Access Point used at the lower LM-MUX service interface, i.e. the IrLAP service boundary.

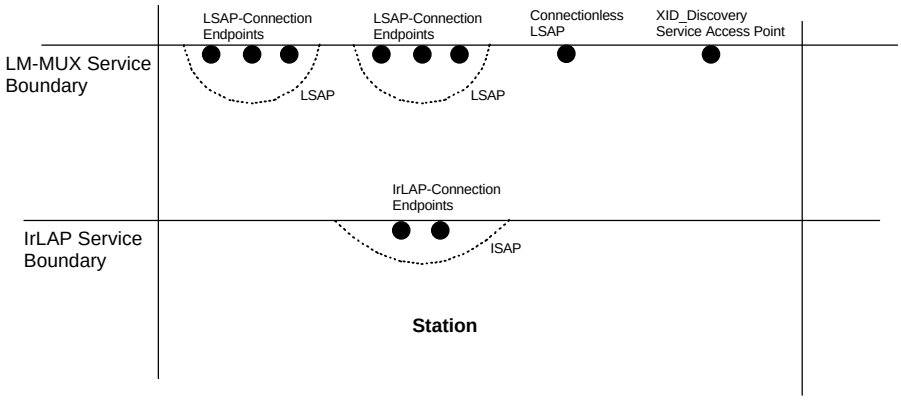


Figure 2. LM-MUX External Interfaces

a) LM_Connect, LM_Disconnect, LM_Data and LM_UData service primitives are invoked at LSAP-connection endpoints. LSAP-connection endpoints are grouped together at an LSAP.

b) LM_ConnectionlessData primitives are invoked at the Connectionless LSAP. LM_ConnectionlessData.indication primitives are delivered to ALL LM-MUX clients that bind[1] to the Connectionless LSAP.

c) LM_Discover and LM_Sniff primitives are invoked at the XID_Discovery Service Access Point. LM_Discover.confirm primitives are directed to the LM-MUX clients that invoked the corresponding LM_Discover.request primitive. LM_Discover.indication primitives are delivered to all LM-MUX clients currently bound to the XID_Discovery SAP.

d) All IrLAP service primitives are invoked at an IrLAP-connection endpoint. There is one IrLAP Service Access Point (ISAP) per station.

[1] Binding mechanisms are a local matter and are not subject to specification in this document.

3.1.2 Service Access Points, Connections and Endpoints

The primary purpose of the LM-MUX is to provide connection-orientated data transfer services between multiple LM-MUX clients (e.g., transport entities or directly bound attached applications). Peer LM-MUX clients are connected by an LSAP-Connection. LSAP-connections between stations are carried over IrLAP-connections.

Figure 3 illustrates the relationships between Stations, LSAPs, LSAP-Connections, LSAP-Connection Endpoints, IrLAP-Connections and IrLAP-Connection Endpoints.

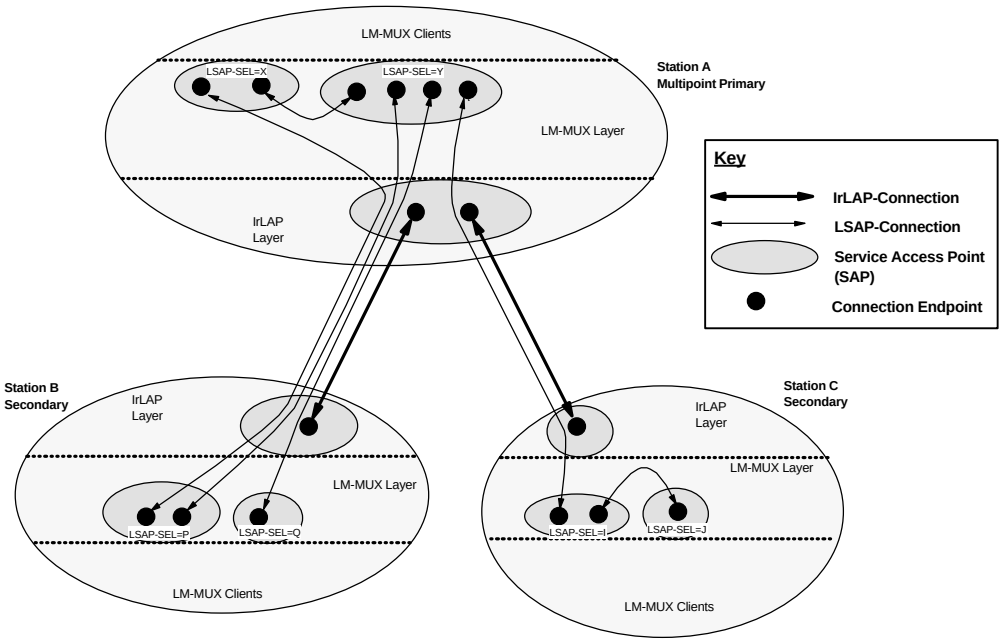


Figure 3. LSAP-Connections, IrLAP Connections and Stations

Within a station LSAPs are distinguished by the value of LSAP-SEL. The LSAP-SEL values for both ends of an LSAP-connection are carried in LM-PDUs. There is no requirement that the LSAPs at each end of a connection are assigned the same LSAP-SEL value. It follows that:

a) There may be no more than one LSAP-connection between the same pair of LSAPs.

   An LSAP may terminate more than one LSAP-connection from the same peer station provided that the other end of the LSAP-connections terminate at distinct LSAPs (e.g., in Figure 3 the two LSAP-connections between Station A and Station B that terminate at the LSAP with LSAP-SEL=Y in Station A; or the two LSAP-connections that terminate at the LSAP with LSAP-SEL=P in Station B).

b) Within a station it is possible for Intra-Station LSAP-connections to be made that do not use an IrLAP connection (e.g., Figure 3 shows such connections between LSAPs in stations A and C).

c) An LSAP-SEL value used in composing an LSAP-address is scoped by the station to which it relates. In Figure 3 the sets of LSAP-SEL values at each station may overlap. Thus, X, P and I may all represent the same LSAP-SEL value.

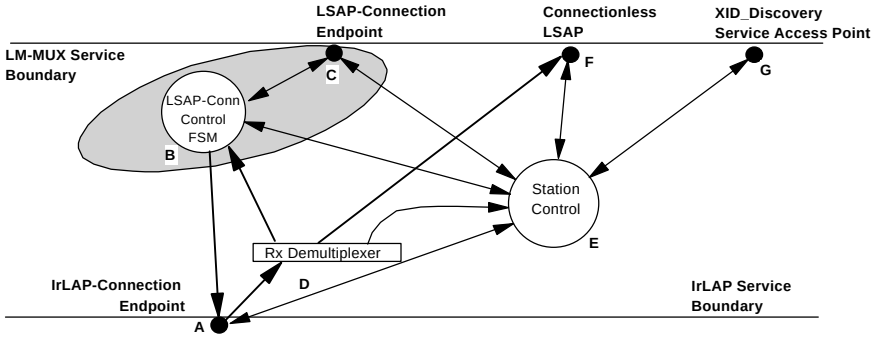
d) There is a single IrLAP Service Access Point (ISAP) per station. The ISAP in an IrLAP point-to-multipoint primary station may support more than one concurrent IrLAP-connection endpoint.

e) Multiple LSAP-connections may make use of the same IrLAP-connection.

3.2 Internal Organization of LM-MUX

This section exposes the fabric of the Link Management Multiplexer (LM-MUX) that lies between its external interfaces. Figure 4 and Figure 5 give a detailed breakdown of how the various primitives (events) are received and generated by LM-MUX and are routed within the LM-MUX entity.

Figure 4 shows the top level routing of events between the external interfaces at the LM-MUX service boundaries; a LSAP-Connection Control Finite State Machine (FSM) associated with each LSAP-connection endpoint; the per station Receive Demultiplexer, and the per station Station Control entity.



A to D Events
IrLAP_Data.indication
IrLAP_UnitData.indication
A to E Events
IrLAP_Connect.indication
IrLAP_Connect.confirm
IrLAP_Disconnect.indication
IrLAP_Status.confirm
IrLAP_Status.indication
IrLAP_Reset.indication
IrLAP_Reset.confirm
IrLAP_Discover.indication
IrLAP_Discover.confirm
IrLAP_Primary.indication
IrLAP_Primary.confirm
IrLAP_NewAddress.confirm
B to A Events
IrLAP_Data.request

B to C Events
LM_Connect.indication
LM_Connect.confirm
LM_Disconnect.indication
LM_Idle.confirm
LM_Data.indication
LM_UData.indication
LM_Status.indication
LM_Status.confirm
B to E Events
LS_Connect.request
LS_Disconnect.request
LS_LockOut.confirm
LS_Status.request
LS_Idle.request

C to B Events
LM_Connect.request
LM_Connect.response
LM_Disconnect.request
LM_Idle.request
LM_Data.request
LM_UData.request
LM_Status.request
C to E Events
LM_AccessMode.request
D to B Events
IrLAP_Data.indication
D to E Events
IrLAP_Data.indication (AccessMode LM-PDU)
E to F Events
LM_ConnectionlessData .confirm

D to F Events
LM_ConnectionlessData .indication
E to A Events
IrLAP_Connect.request
IrLAP_Connect.response
IrLAP_Disconnect.request
IrLAP_Data.request
IrLAP_UnitData.request
IrLAP_Status.request
IrLAP_Sniff.request
IrLAP_Discover.request
IrLAP_Primary.request
IrLAP_Primary.response
IrLAP_NewAddress.request

E to B Events
LS_Connect.confirm
LS_Disconnect.indication
LS_LockOut.request
LS_Status.indication
LS_Status.confirm
E to C Events
LM_AccessMode.indication
LM_AccessMode.confirm
E to G Events
LM_Sniff.confirm
LM_DiscoverDevices.indication
LM_DiscoverDevices.confirm
F to E Events
LM_ConnectionlessData .request
G to E Events
LM_DiscoverDevices.request
LM_Sniff.request

Figure 4. Multiplexer Internal Organization

3.2.1 LSAP-Connection Control FSM

An LSAP-connection control FSM is associated with each LSAP-connection endpoint. It is responsible for the connection and disconnection of a single LSAP-connection with a peer LSAP-connection endpoint. During active LSAP-connection establishment the FSM requests the use of a suitable IrLAP-connection from Station Control by issuing an LS_Connect.request internal service primitive. When a suitable IrLAP-connection is available Station Control responds with an LS_Connect.confirm that identifies which IrLAP-connection (endpoint) should be used for the LSAP-connection. If Station Control is unable to provide a suitable IrLAP-connection it responds with an LC-Disconnect.indication primitive.

Once an IrLAP-connection is available the initiating LSAP-connection control FSM sends an Connect LM-PDU to the peer FSM. This carries any user-data supplied in the LM_CONNECT.request primitive that initiated the LSAP-connection. The arrival of the Connect LM-PDU at the peer FSM results in the generation of an LM_CONNECT.indication primitive being delivered at the peer LSAP-connection endpoint. The peer LM-MUX client may reject the LSAP-connection, in which case it invokes LM_Disconnect.request at its LSAP-connection endpoint. This results in a Disconnect LM-PDU being returned to the initiating endpoint which in turn results in an LM_Disconnect.indication being returned to the initiating LM-MUX client. Alternatively, to accept the incoming LSAP-connection the responding LM-MUX client invokes an LM_Connect.response primitive at its LSAP-connection endpoint. This results in the return of a Connect Confirm LM-PDU to the initiating FSM and an LM_Connect.confirm primitive is sent from the LSAP-connection endpoint to the initiating LM-MUX client.

Attempts to exchange data between peer LM-MUX clients will fail unless an LSAP-connection has been established between them. Data LM-PDUs arriving at an LSAP-connection control FSM (encapsulated in an IrLAP_Data.indication primitive) are discarded unless they are sent by the established peer of an LSAP-connection. Likewise, LM_Data.request primitives sent into an LSAP-connection endpoint are rejected until an LSAP-connection is established through that endpoint. The sequence of events involved in establishing and clearing LSAP-connections is discussed in more detail in section 7.

An LSAP-connection FSM also forwards LM_Status service primitives between the LSAP-connection endpoint and Station Control once an LSAP-connection has been established.

3.2.2 Receive Demultiplexer

The Receive Demultiplexer is responsible for the routing of all IrLAP_Data.indication and IrLAP_Unitdata.indication primitives that arrive via any of the active IrLAP-connection endpoints.

The Receive Demultiplexer is also responsible for selecting a new LSAP-connection endpoint for the delivery of an incoming Connect LM-PDU. If it is not possible to match an incoming Connect LM-PDU with an LSAP-connection endpoint, the Receive Demultiplexer must generate a Disconnect LM-PDU to return to the originator of the Connect LM-PDU.

The Receive Demultiplexer routes IrLAP_Data.indication and IrLAP_Unitdata.indication primitives as follows:

a) Delivery of any LM-PDUs, via IrLAP_Data.indication or IrLAP_Unitdata.indication, with DLSAP-SEL or SLSAP values in the range 0x71..0x7F is currently undefined and as such they should be discarded.

b) Any Control LM-PDUs delivered via IrLAP_Unitdata.indication or IrLAP_Data.indication with the the *expedited* parameter set to *true* are discarded. ie. Control LM-PDUs delivered in IrLAP UI frames are discarded.

c) All AccessMode Request or AccessMode Confirm LM-PDUs (delivered in I frames via IrLAP_Data.indication with the *expedited* parameter set *false*) are routed to Station Control.

d) All other defined Control LM-PDUs are routed to an LSAP-Connection Control FSM in the same way as Data LM-PDUs (see below).

e) Undefined Control LM-PDUs or Connect, Connect Confirm and Disconnect Control LM-PDUs addressed to a non-existent LSAP-Connection endpoint are discarded. In the case of a Connect LM-PDU a Disconnect LM-PDU is returned to the originator with the reason code of noPeerMuxClient (0x08).

f) All Data LM-PDUs delivered via IrLAP_Unitdata.indication primitives (UI frames send outside an IrLAP connection) except for those addressed between Connectionless LSAPs (DLSAP-SEL=SLSAP-SEL=0x70) are discarded.

g) Data LM-PDUs carried in I or UI IrLAP frames containing DLSAP-SEL and SLAP-SEL values of 0x70 are delivered to the Connectionless LSAP (if implemented).

h) All other Data LM-PDUs (I frames or UI frames within a connection) that carry a normal connection address (DLSAP-SEL and SLSAP-SEL values in the range 0x00..0x6F inclusive) are routed to the LSAP-Connection Control FSM at the LSAP-connection endpoint identified by the triple:

   <IrLAP-connection Address><DLSAP-SEL><SLSAP-SEL>

   The IrLAP-connection address serves as an abbreviation for the remote device address during the lifetime of the underlying IrLAP connection. Note that for stations that are capable of only point-to-point IrLAP primary or IrLAP secondary operation, LSAP-connection endpoint identification is based solely on the tuple:

   <DLSAP-SEL><SLSAP-SEL>

i) If no LSAP-connection endpoint can be identified for an received Data LM-PDU a Disconnect LM-PDU is returned to the originator with a reason code *disconnected* (0x06).

3.2.3 Station Control

Figure 5 descends within the Station Control entity to expose the top-level Station Control FSM; a per IrLAP-connection endpoint IrLAP-Connection Control FSM; and supporting tables to maintain the relationships between LSAP-connections and IrLAP connections.

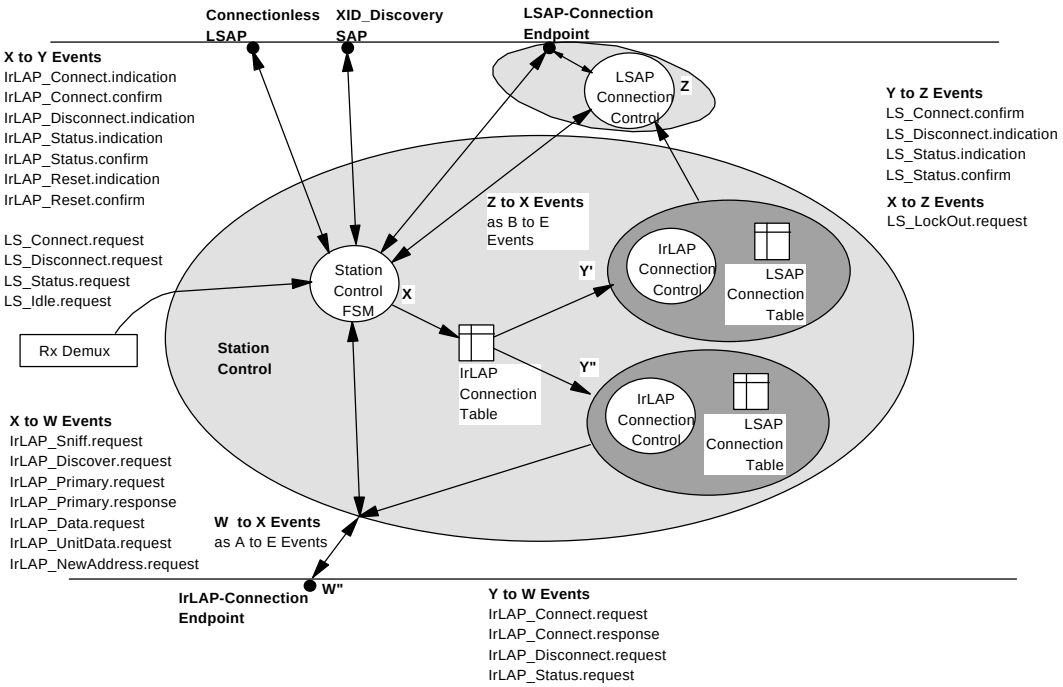


Figure 5. Station Control Internal Organization

Station Control is responsible for:

1. The transmission of connectionless data as a result of the invocation of LM_ConnectionlessData.request primitives at the Connectionless LSAP.

2. The operation of the XID_Discovery process and associated IrLAP device address resolution resulting from the invocation of LM_DIscovery.request and LM_Sniff.request primitives.

3. The connection and disconnection of IrLAP-connections.

4. The assignment of LSAP-connections to IrLAP-connections.

5. Transitions between the Exclusive and Multiplexed LM-MUX modes.

3.2.3.1 Station Control FSM

The Station Control FSM is primarily an event handler/dispatcher. However there are three mutually exclusive sets of operations that it must orchestrate.

1. XID_Discovery and IrLAP Device Address resolution

2. Connection and Disconnection of IrLAP-connection

3. LM-MUX Exclusive Mode

Also, based on the presence or absence of IrLAP-connection at the local station the Station Control FSM determines whether to forward an LM_ConnectionlessData.request as an IrLAP_Data.request primitive with the unacknowledged expedited data flag set, or as an IrLAP_Unitdata.request primitive.

For each IrLAP-connection endpoint Station Control maintains a separate IrLAP-connection control FSM and an associated LSAP-connection table that enumerates the LSAP-connections that use that IrLAP connection.

While XID discovery/IrLAP device address resolution is in progress the Station Control FSM services only discovery related events. Internal LS_xxxx primitives, client invoked LM-MUX primitives and IrLAP-connection related events are held pending until discovery/address resolution has completed.

The IrLAP-Connection Table maintains a record status of the IrLAP-connection endpoints present within the station. Indeed each IrLAP-connection control FSM and associated LSAP-Connection Table maybe regarded as being embedded in a row of the IrLAP-Connection table. The information associated with each active IrLAP-connection includes a record of the remote device address. This is the basis upon which Station Control determines whether a suitable IrLAP-connection exists to service an LSAP-connection or whether a new IrLAP-connection need be established.

In order to establish more than one IrLAP-connection Station Control must acquire the IrLAP primary role. This is attempted when the need to make a second IrLAP-connection is detected.

3.2.3.2 IrLAP-Connection Control FSM

Associated with each IrLAP-connection is a control FSM whose role is to progress the establishment of an IrLAP-connection and to ensure its disconnection when there cease to be any active LSAP-connection present on the IrLAP-connection.

The IrLAP-connection control FSMs also duplicate LS_Connection.confirm, LS_Disconnect.indication, LS_Status.indication, and LS_Status.confirm internal events to each of the LSAP-connection endpoints associated with the local end of an IrLAP connection.

3.2.3.3 Connection Tables

The IrLAP-connection table and LSAP-connection tables are used to maintain the relationships between LSAP-connections and IrLAP-connections shown in Figure 5. The notion of tables is used here for convenience of description. Any means of maintaining the required relationships is acceptable. Note that for stations that are capable of only point-to-point IrLAP primary or IrLAP secondary operation the IrLAP-connection table maintains just a single entry, i.e. is essentially NULL, and there is a single instance of the IrLAP-connection control FSM and associated LSAP-connection table.

3.3 IrLMP Service Specification

There are three groups of external IrLMP service primitives; discovery, link control, and data transfer. The following sections outline these primitives.

3.3.1 Link Management Discovery

The link management discovery process uses the mechanisms provided by the IrDA Link Access Protocol so that applications can find out what other IrDA devices are reachable. The IrDA Link Management Protocol provides basic information for each device it manages to contact.

Application writers should note that the discovery process is inherently imperfect. The dynamic infra-red environment means that a device may not be able to reply to the request for its address and device addresses may need to change because of address conflicts.

3.3.1.1 Discovery Service Primitives

3.3.1.1.1 LM_DiscoverDevices

The `LM_DiscoverDevices` service causes a single IrLAP discovery operation if it is possible, that is, if the link is in contention state. If the link is currently in use, the results of the last discovery operation this device was involved in are returned. This may only include the device at the other end of the link if this device did not instigate the last discovery process.

If an address conflict is detected, IrLMP will attempt to resolve each set of conflicting addresses ***once*** . It will then ***remove*** all entries with conflicting addresses.

This service uses the broadcast IrLAP discovery service, sent to all devices that can be reached.

```
LM_DiscoverDevices.request(nrSlots)
LM_DiscoverDevices.confirm(Status
                    ,List of(deviceAddress, DeviceInfo, method ))
LM_DiscoverDevices.indication(deviceAddress, DeviceInfo, method)
```

nrSlots	Parameter to the IrLAP process representing the number of time slots for devices to respond (see [IRLAP]).
status	Indicates whether an active discovery process was completed or that cached data is provided.
deviceAddress	The 32-bit IrLAP device address.
DeviceInfo	This is the advertised device information as described in section 3.4.1
method	This indicates how the device was discovered (sniffing, active discovery, passive discovery, i.e., a side effect of being discovered).

This service primitive is required.

3.3.1.1.2 LM_Sniff

The LM_Sniff service is a way to invoke the sniffing services provided by IrLAP. This is invoked at the Station entity, and no other services are available until this mode is canceled. The request is satisfied when another device connects at the IrLAP level, or if the request is canceled. The request is implicitly cancelled by the invocation of LM_ConnectionlessData.request, LM_Connect.request or LM_DiscoverDevices.request. A LM_Sniff.confirm we a status valued of ‘cancelled’ is generated when the sniff request is implicitly cancelled.

```
LM_Sniff.request(option)
LM_Sniff.confirm(status, deviceAddress)
```

option	Start/Stop	
status	Connection established or	
	Sniff refused because:	a) IrLAP connection already present
		b) Sniff already active
deviceAddress	IrLAP deviceAddress of device connected to	

This service primitive is optional.

3.3.2 Link Management Link Control

3.3.2.1 Link Control Service Primitives

3.3.2.1.1 LM_Connect

Once the LSAP for a transport entity on a remote device has been identified, the LSAP for the local transport entity and the remote entity must be bound for data to be sent.

LSAPs are bound in pairs. There may be at most one LSAP-connection between any given pair of LSAPs.

```
LM_Connect.request(Called LSAP, Requested QoS, Client Data)
LM_Connect.indication(Calling LSAP, Resultant QoS, Client Data)
LM_Connect.response(Calling LSAP, Client Data)
LM_Connect.confirm(Called LSAP, Resultant QoS, Client Data)
```

LSAP	A reference to an LSAP (typically an LSAP-ID)
QoS	Quality of Service parameters. The parameters that can be changed are detailed below. Which parameters are actually allowed to be changed is implementation specific.
Client Data	Data that a service user wants to send along in the connection packet. Typically this might be used as a signature field to help decide whether to accept the connection, or simply to piggyback a small amount of data. Due to IrLAP data size restrictions only data of up to 60 bytes is guarenteed to be delivered[2].

Quality of Service parameters (see [IRLAP]):
- baud rate
- max. turn around time [Input only]
- data size
- disconnect threshold

[2] Although not illegal, transmission of more that 60 bytes data is discouraged. Reception of data greater than 60 bytes may be the basis for immediately disconnecting; or it may be ignored; or it may be delivered to the LM-MUX client.

A LSAP service user may request a Quality of Service (QoS) for the IrLAP link. If there are no other LSAP connections, or the IrLAP link does not yet exist, an attempt to provide the requested QoS will be made. The connection will not be rejected simply because the QoS could not be met. If the connection succeeds, the actual QoS parameters will be returned. If the actual QoS is not sufficient, it is up to the LSAP Service User to disconnect.

This service primitive is required.

3.3.2.1.2 LM_Disconnect

Ask that an LSAP connection be broken. A LSAP service user cannot refuse to do this.

```
LM_Disconnect.request(Reason, Client Data)
LM_Disconnect.indication(Reason, Client Data)
```

Reason	The reason the connection is being/was closed.
Client Data	Service user data as described in section 3.3.2.1.1

There is no guarantee that the client data is delivered. The encoding of the reason code is described in section 3.4.2.2.

This service primitive is required.

3.3.2.1.3 LM_Status

The request/confirm pair provide information on whether there is still unacknowledged data in the IrLAP queue. Since IrLAP does not provide a graceful close, this information is useful in determining when it is safe to disconnect.

A variety of general status indications are delivered as they occur. It is implementation dependent how they are made available to service users. An LSAP-connection endpoint is informed when a transition to exclusive mode on behalf of another LSAP-connection occurs, *LockStatus=Locked*, after which all data transfer requests will be rejected. Likewise the endpoint is informed when data transfer may resume, *LockStatus=Unlocked*. The *LinkStatus* parameter conveys status indications that originate from IrLAP.

```
LM_Status.request()
LM_Status.indication(LinkStatus,LockStatus)
LM_Status.confirm(UnAcked Data Flag)
```

LinkStatus	Ok, No progress, noisy (see [IRLAP]).
LockStatus	NoChange, Locked, UnLocked
UnAcked Data Flag	True/False - indicates whether unacked data is in the IrLAP queue.

This service primitive is required.

3.3.2.1.4 LM_Idle

The LM_Idle service primitive is invoked at a LSAP connection endpoint. It is used to mark the LSAP connection (locally) as idle/active. When the connection is first established it is considered to be in the active state. If a service user wants to keep the connection open for a long time without actively using the link, it can choose to inform link management.

The LM_Idle service has two purposes. Firstly, to allow the establishment of an exclusive mode LSAP connection by another LSAP service user when all other LSAP connections have been marked idle. The second purpose is to allow the IAS client and server to mark the LSAP connection as idle so that the underlying IrLAP connection may be disconnected in the event that there are no other active LSAP connections. This prevents a full IrLAP disconnect and reconnect between bursts of IAS queries, see section 4.

LM_Idle.request(Req mode)
LM_Idle.confirm(status, Actual mode)

mode	Active/Idle
status	Success/Failure

This service primitive is optional.

3.3.2.1.5 LM_AccessMode

Change between exclusive and multiplex mode.

```
LM_AccessMode.request(Requested Mode)
LM_AccessMode.indication(Resultant Mode)
LM_AccessMode.confirmation(status, Actual Mode)
```

Mode	requested/actual mode (Exclusive/Multiplexed)
status	success/failure of request

This service primitive is optional.

3.3.3 Link Management Data Transfer

3.3.3.1 Data Transfer Service Primitives

3.3.3.1.1 LM_Data

Send an I frame to the remote LSAP. The I frame is sent reliably in the absence of failures of the link level connection but the sender of the data is not told when it arrives. If the underlying IrLAP link level connection breaks, data may be lost. No information about lost data is available to the sender. However, both LSAP clients will become aware of the potential loss as they will each receive a LM_Disconnect.indication. The size of user data is constrained to fit within one IrLAP I frame.

```
LM_Data.request(Data)
LM_Data.indication(Data)
```

Data	User data

This service primitive is required.

3.3.3.1.2 LM_UData

Send a UI frame to the remote. UI frames are sent and delivered before any outstanding I frames for the same destination LSAP. UI frames are not sent reliably. The size of user data is constrained to fit within one IrLAP UI frame.

```
LM_UData.request(Data)
LM_UData.indication(Data)
```

Data	User data

This service primitive is required.

3.3.3.1.3 LM_ConnectionlessData

Send out of connection UI frames. These are sent unreliably. There is a single delivery point for delivery of such UI frames. This document does not specify a mechanism for determining the correct destination software entity within a device for the frame. Such frames are broadcast to all devices in range. No account is taken of primary/secondary roles. The size of the user data is constrained to fit within one IrLAP UI frame.

```
LM_ConnectionlessData.request(Data)
LM_ConnectionlessData.indication(Data)
LM_ConnectionlessData.confirm(status,[reason])
```

Data	User data
status	success/fail of request
reason	Optional reason for failure

The .confirm primitive is used to indicate whether the corresponding .request was passed to IrLAP, `status=success`, or discarded by LM-MUX, `status=fail`. Currently the only reason for the LM-MUX to discard the request is because it is in exclusive mode. Note that .confirm only indicates that responsibility for delivery has passed to IrLAP. It does NOT indicate that any resulting UI frame has been successfully delivered.

This service primitive is optional.

3.4 Frame Formats

3.4.1 DeviceInfo Field Format

Link Management controls the content of the DeviceInfo field in the IrLAP discovery process. This section describes the format of that field, used both for broadcast XID frames and directed XID frames during address resolution.

The DeviceInfo field may be empty, i.e. zero length, however, when this field is non-empty it will always carry one or more octets of hints. Service hints are useful when devices with different services exist in the same IR space (.e.g., a laptop, a printer, and a modem). The optional Device Nickname is useful when many devices of the same type (and potentially similar services) exist in the same IR space (e.g., several PDAs). The format of the DeviceInfo field is shown below.

Service Hints	Device Nickname
n octets	0 to 23 - n octets

Currently upto 2 octets of Sevice Hints are defined below.

3.4.1.1 Service Hints

The first two octets of the DeviceInfo field contain the IrLMP hint mask. All undefined hints are set to zero. The eighth bit of every hint byte (bit 7, 15, 23, ...) is an extension bit and indicates whether or not an additional hint byte is included in the DeviceInfo field. It is permissible to truncate the hint mask to a single byte by clearing the first extension bit. Table 1 summarizes the current IrLMP hint mask.

Hints bit 0 indicates the presense or absense of a instance of the Plug and Play object class “PnP” described in [IRPNP]. When set an instance of this the PnP object class is expected to be present.

The service hints are managed by Link Management but Link Management does not guarantee the accuracy of the information. Service hints should not be taken to mean a particular service is provided by the device. They are merely to provide assistance in choosing a device to contact during the discovery process.

Byte 1	
Bit	Function
0	PnP Compatible
1	PDA/Palmtop
2	Computer
3	Printer
4	Modem
5	Fax
6	LAN Access
7	Extension

Byte 2	
Bit	Function
8	Telephony
9	File Server
10	rsvd
11	rsvd
12	rsvd
13	rsvd
14	rsvd
15	Extension

Table 1. IrLMP Service Hints

3.4.1.2 Device Nickname

The Device Nickname field consists of the following elements.

Character Set	Name
1 octet	0 to 23-(n+1) octets

The device nickname may be a truncated version of the name returned as the value of the “Device” object’s DeviceName attribute (refer to section 5 for a description of the DeviceName attribute). Character set encoding should follow the same rules as laid out in section 4.3.3.2.4.

It should be noted that the total number of bytes in the deviceInfo field must not exceed 23 bytes even though the IrLAP specification allows up to 32 bytes. This is to prevent the XID process from spilling over into the next slot. See [IRLAP]. What this means is that if additional hint bytes are transmitted, the device nickname must be shortened by a corresponding number of bytes. For example, if three hint bytes are transmitted instead of two, the device nickname can have a maximum length of 19 bytes (3 hint bytes + 1 character set byte + 19 device nickname bytes = 23 total bytes).

3.4.2 LM-PDU Formats

All LM-PDU frames are sent as IrLAP data frames. Link Management uses a two-octet header within the IrLAP data frame encoded as follows:

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
C	DLSAP-SEL							r	SLSAP-SEL						

where C is the control bit. When the control bit is set to 1, it indicates that the frame is a command frame. When the control bit is set to 0, the LM_SDU is treated as data. The r bit is reserved for future use and should be set to 0.

3.4.2.1 Data Transfers

Data transfer frames are encoded as follows:

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	
0	DLSAP-SEL							0	SLSAP-SEL							Data ....

3.4.2.2 Link Control Frames

Link control frames are encoded as follows:

7	6-0	7	6-0	7	6-0	
1	DLSAP-SEL	0	SLSAP-SEL	A	opcode	parameters

where the A bit when set as 0 signifies a command request at the source side and should be interpreted as a command indication at the destination side. When the A bit is set as 1 it is command response at the source side and a command confirmation at the destination side. All frames are sent as reliable data.

A	opcode	Frame	Command	parameters
0	1	I	Connect	rsvd = 0x00 (Optional) or rsvd = 0x00, LMS-UserData (Optional)
1	1	I	Connect (confirm)	rsvd = 0x00 (Optional) or rsvd = 0x00, LMS-UserData (Optional)
0	2	I	Disconnect	reason, LMS-UserData (Unspecified)
0	3	I	AccessMode	rsvd=0x00, mode
1	3	I	AccessMode (confirm)	status, mode

Table 2. Link Control Frame Values

The *rsvd*, *status*, *reason* and *mode* parameters are all single octet values as defined in the following tables.

Reason	Code
User Request	0x01
Unexpected IrLAP Disconnect	0x02
Failed to establish IrLAP connection	0x03
IrLAP Reset	0x04
Link Management Initiated Disconnect	0x05
Data delivered on disconnected LSAP-Connection	0x06
Non Responsive LM-MUX Client	0x07
No available LM-MUX Client	0x08
Connection Half Open	0x09
Illegal Source Address (i.e. 0x00)	0x0a
Unspecified Disconnect Reason	0xff

Table 3. Disconnect Reason Codes

The disconnect reason codes are considered to be advisory and give an indication as to why the LM-MUX connection was terminated. The ‘Unspecified’ reason code is always valid and other reason codes should only be used when they indicate the correct reason for the disconnection.

Mode	Value
Multiplexed	0x00
Exclusive	0x01

Table 4. IrLMP Mode Values

Status	Value
success	0x00
failure	0x01
unsupported	0xff /* Access Mode Confirm Only */

Table 5. IrLMP Control LM-PDU Status Values.

3.5 Detailed Descriptions

3.5.1 Introduction

The following sections specify in detail the IrLMP operating procedures. These procedures define the behavior of the IrLMP layer during each phase of operation. The operation procedures include: station control, IrLAP link connection control, and LSAP-connection control.

3.5.2 Station Control

3.5.2.1 Purpose

The Station Control FSM is responsible for the coordination of the Link Control and LSAP-Connection FSMs. The Station Control FSM also directly controls the discovery process, address resolution, sniffing, and access mode. With the exception of the IrLAP IrLAP_DATA.indication, all incoming IrLAP events are initially handled by the Station Control FSM. Only one instance of this FSM exists.

3.5.2.2 Overview

The Station Control FSM controls the major phases of activity with the station. These may be regarded as:

1. Discover and Address Resolution, handled by the states DISCOVER and RESOLVE ADDR.

2. LM-MUX transitions between multiplexed and exclusive mode, handled by the states EXCLUSIVE PEND, EXCLUSIVE and READY PEND.

3. Primary/Secondary Role exchange, handled by the state ROLE EXCHANGE.

4. Sniffing, handled by the state SNIFF.

The READY state is central to all this activity. Discovery occurs when an LM_DiscoverDevices.request primitive is received when there are no established or establishing IrLAP connections. IIrLAP returns conflicting deviceInfo record, ie. there are multiple entries for the same device address, address resolution is perform to ensure that the deviceInfo records that are returned relate distinct.device addresses. If IrLAP connection exist when the LM_DiscoverDevices is received then the cached results of a previous operation are returned, possibly augmented with further deviceInfo records reported to station control by IrLAP via IrLAP_Discover.indication primitives.

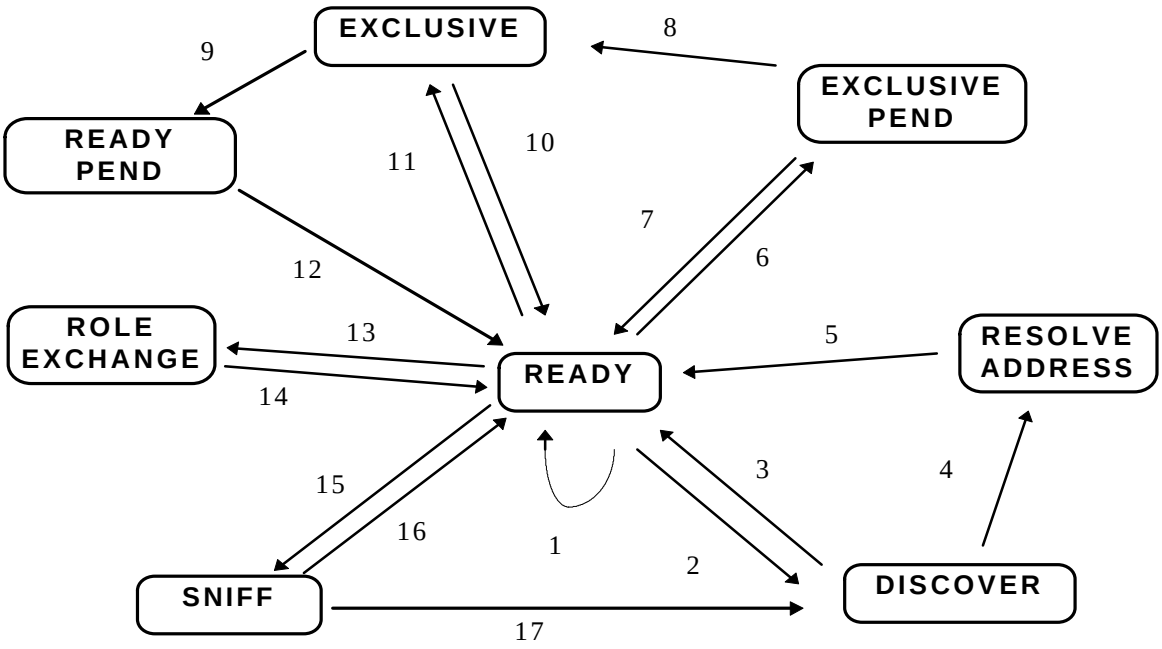
The READY state guards the start of a transition from multiplexed to exclusive mode by ensuring that the local conditions are meet before allowing the transition to commence. If the transition is being made in response to a remote request and local conditions permit the transition is direct to the EXCLUSIVE state. If the transition is the result of a local LM_AccessMode.request and and local conditions allow an AccessMode Request LM-PDU is sent to the remote Station Control FSM and the transition is made to the EXCLUSIVE PEND state where a result from the remote peer is awaited. If the transition to exclusive mode is acceptable to the remote peer a further transition to the EXCLUSIVE state occurs. Otherwise a return transition to the READY state is made and any local locks previously established are released. A similar set if transitions occurs on the way out of exclusive mode back to multiplexed mode. In this case the mode change, exclusive to multiplexed may not be refused by the remote peer, so there is no possible transition from READY PEND back to EXCLUSIVE.

Primary/secondary role exchange is request by a point-to-multipoint capable LM-MUX if it needs to establish more than one IrLAP connection. In this case the requesting Station Control invokes the IrLAP_Primary service and transitions to the state ROLE EXCHANGE. The remote Station Control will permit the role exchange in all states where there is just a single IrLAP connection. Following the result (successful exchange or not) the Station Control FSM transitions back to the READY state either progressing the LM-Connect.request that (indirectly) initiated the IrLAP connection attempt or (indirectly) failing it.

A request to start sniffing is honoured only if there are no existing IrLAP connections. Snifffing can terminate due to: the passive establishment of an IrLAP connection; an attempt to send Connectionless Data; an active attempt to establish an LSAP-connection; and a request to do DeviceDiscovery. All but the latter, which transitions directly to DISCOVERY, cause a transition from the SNIFF state back to the ready state.

3.5.2.3 Precise Description

3.5.2.3.1 Station Control State Transition Diagram



3.5.2.3.2 Station Control State Transition Table

State	Event	Action	Next State	
READY	IrLAP_Connect.indication	Forward [IrLAP_Connect.indication];	READY	1
		IrLAP_Disconnect.request /* No resources to accept connection */	READY	1
	IrlAP_Connect.confirm	Forward [IrLAP_Connect.confirm]	READY	1
	IrLAP_Disconnect.indication	Forward [IrLAP_Disconnect.indication]	READY	1
	IrLAP_Status.confirm	Forward [IrLAP_Status.confirm]	READY	1
	IrLAP_Status.indication	Forward [IrLAP_Status.indication]	READY	1
	IrLAP_Reset.indication	Forward [IrLAP_Reset.indication]	READY	1
	IrLAP_Reset.confirm	Forward [IrLAP_Reset.confirm]	READY	1
	IrLAP_Discover.indication(Log)	/* Accumulate in CacheLog */ CacheLog = CacheLog ∪ Log LM_DiscoverDevices.indication (status=passive,Log)	READY	1
		/* Replace CacheLog */ CacheLog = Log LM_DiscoverDevices.indication (status=passive,Log)	READY	1
	IrLAP_Discover.confirm	Error /* No outstanding request */	READY	1
	IrLAP_NewAddress.confirm	Error /* No outstanding request */	READY	1
	IrLAP_Primary.indication ∧ Connected = ∅	Error /* No IrLAP connection */	READY	1
	IrLAP_Primary.indication ∧ #Connected = 1	IrLAP_Primary.response(deny=false) /* Allow the swap */	READY	1
	IrLAP_Primary.indication ∧ #Connected > 1	IrLAP_Primary.response(deny=true) /* Disallow the swap */	READY	1
	IrLAP_Primary.confirm	Error /* No outstanding request */	READY	1
	LS_Connect.request(deviceAddress) ∧ ((deviceAddress ∈ Connected) ∨ (Connected = ∅))	Forward [LS_Connect.request]	READY	1
	LS_Connect.request(deviceAddress) ∧ deviceAddress ∉ Connected ∧ Connected ≠ ∅ ∧ MultiPointSupportEnabled	IrLAP_Primary.request	ROLE EXCHANGE	13
	LS_Connect.request(deviceAddress) ∧ deviceAddress ∉ Connected ∧ Connected ≠ ∅ ∧ !MultiPointSupportEnabled ∧ IdleIrLAPConnections = ∅	LS_Disconnect.indication (noIrLAPConnection)	READY	1
	LS_Connect.request(deviceAddress) ∧ deviceAddress ∉ Connected ∧ Connected ≠ ∅ ∧ !MultiPointSupportEnabled ∧ IdleIrLAPConnections ≠ ∅	∀ IrLAP Connections ∈ IdleIrLAPConnections Forward [LS_ForceDisconnect.request] Forward [LS_Connect.request]	READY	1
	LS_Disconnect.request	Forward [LS_Disconnect.request]	READY	1
	LS_Status.request	Forward [LS_Status.request]	READY	1
	LS_Idle.request(mode)	Forward[LS_Idle.request(mode)]	READY	1
	LM_AccessMode.request (mode=exclusive) ∧ LocalLockable(lsap_ce) = true	exclusiveLsap_ce = lsap_ce; LocalLock(exclusiveLsap_ce) IrLAP_Data.request (AccessMode Request LM-PDU [mode=exclusive], expedited=false) StartStationWDTimer	EXCLUSIVE PEND	6
	LM_AccessMode.request (mode=exclusive) ∧ LocalLockable(lsap_ce) = false	LM_AccessMode.confirm (status=localFail,mode=multiplexed)	READY	1
	LM_AccessMode.request (mode=multiplexed)	LM_AccessMode.confirm (status=success,mode=multiplexed)	READY	1
	IrLAP_Data.indication (AccessMode Request LM-PDU [mode=multiplexed])	Error /* Not in exclusive mode */	READY	1

State	Event	Action	Next State	
	IrLAP_Data.indication (AccessMode Request LM-PDU [mode=exclusive]) ∧ LocalLockable(lsap_ce)=true	/* Identify THE exclusive connection */ exclusiveLsapCe=lsap_ce; LocalLock(exclusiveLsap_ce) IrLAP_Data.request (AccessMode Confirm LM-PDU [status=success, mode=exclusive], expedited=false) /* Tell the newly exclusive lsap_ce that its exclusive*/ LM_AccessMode.indication (mode=exclusive)	EXCLUSIVE	11
	IrLAP_Data.indication (AccessMode Request LM-PDU [mode=exclusive]) ∧ LocalLockable(lsap_ce)=false	IrLAP_Data.request (AccessMode Confirm LM-PDU [status=fail, mode=multiplexed], expedited=true)	READY	1
	IrLAP_Data.indication (AccessMode Confirm LM-PDU)	Error /* No outstanding request */	READY	1
	LM_ConnectionlessData.request(data) ∧ Connected = ∅	IrLAP_UnitData.request (Data LM-PDU [DLSAP-SEL=0x70, SLSAP-SEL=0x70, data]) LM_ConnectionlessData.confirm (status=success)	READY	1
	LM_ConnectionlessData.request(data) ∧ Connected ≠ ∅	IrLAP_UnitData.request (Data LM-PDU [DLSAP-SEL=0x70, SLSAP-SEL=0x70, data]) LM_ConnectionlessData.confirm (status=success)	READY	1
	LM_DiscoverDevices.request ∧ Connected = ∅	IrLAP_Discover.request	DISCOVER	2
	LM_DiscoverDevices.request ∧ Connected ≠ ∅	LM_DiscoverDevices.confirm (status=cache,CacheLog);	READY	1
	LM_Sniff.request(option=start) ∧ Connected = ∅	IrLAP_Sniff.request(cancel=false)	SNIFF	15
	LM_Sniff.request(option=start) ∧ Connected ≠ ∅	LM_Sniff.confirm (status=refused,deviceAddress=null)	READY	1
	LM_Sniff.request(option=cancel)	Error /* Not Sniffing! */	READY	1
	StationWDTimerExpired	/* Ignore */	READY	1
DISCOVER	IrLAP_Connect.indication	IrLAP_Disconnect.request /* reject the connection attempt */	DISCOVER	
	IrIAP_Connect.confirm	Error /* No pending IrLAP Connections */	DISCOVER	
	IrLAP_Disconnect.indication	Error /* No IrLAP connections */	DISCOVER	
	IrLAP_Status.confirm	Error /* No IrLAP connections */	DISCOVER	
	IrLAP_Status.indication	Error /* No IrLAP connections */	DISCOVER	
	IrLAP_Reset.indication	Error /* No IrLAP connections */	DISCOVER	
	IrLAP_Reset.confirm	Error	DISCOVER	
	IrLAP_Discover.indication(Log)	/* Ignore */	DISCOVER	
	IrLAP_Discover.confirm(Log) ∧ AddressConficts(Log) = ∅	CacheLog = Log; LM_DiscoverDevices.confirm (status=newLog, Log)	READY	3

State	Event	Action	Next State	
	IrLAP_Discover.confirm(Log) ∧ AddressConficts(Log) ≠ ∅	Conflicts = AddressConflicts(Log); CacheLog = Log CacheLog = CacheLog -Conflicts; ConflictAddresses = ExtractAddresses(Conflicts); resolveAddress = ConflictAddresses[0]; ConflictAddresses = ConflictAddresses - {resolveAddress}; IrLAP_NewAddress.request (resolveAddress);	RESOLVE ADDR	4
	IrLAP_NewAddress.confirm	Error /* No Outstanding Request */	DISCOVER	
	IrLAP_Primary.indication	Error /* No IrLAP Connections */	DISCOVER	
	IrLAP_Primary.confirm	Error /* No outstanding request */	DISCOVER	
	IrLAP_Data.indication (AccessMode Request LM-PDU)	Error /* No IrLAP Connections */	DISCOVER	
	IrLAP_Data.indication (AccessMode Confirm LM-PDU)	Error /* No IrLAP Connections */	DISCOVER	
	LM_AccessMode.request	Error /* No IrLAP connection */	DISCOVER	
	LS_Disconnect.request	Error /* No IrLAP connections */	DISCOVER	
	LS_Status.request	Error /* No IrLAP connections */	DISCOVER	
	LS_Idle.request(mode)	Error /* No IrLAP connection */	DISCOVER	
	IrLAP_Data.indication	Error /* No IrLAP connection */	DISCOVER	
	StationWDTimerExpired	/* Ignore */	DISCOVER	
	LS_Connect.request, LM_ConnectionlessData.request, LM_DiscoverDevices.request, LM_Sniff.request	***/* Left pending */***	***DISCOVER***	
RESOLVE ADDR	IrLAP_Connect.indication	IrLAP_Disconnect.request; /* Reject the connection attempt */	RESOLVE ADDR	
	IrlAP_Connect.confirm	Error /* No pending IrLAP Connections */	RESOLVE ADDR	
	IrLAP_Disconnect.indication	Error /* No IrLAP connections */	RESOLVE ADDR	
	IrLAP_Status.confirm	Error /* No IrLAP connections */	RESOLVE ADDR	
	IrLAP_Status.indication	Error /* No IrLAP connections */	RESOLVE ADDR	
	IrLAP_Reset.indication	Error /* No IrLAP connections */	RESOLVE ADDR	
	IrLAP_Reset.confirm	Error	RESOLVE ADDR	
	IrLAP_Discover.indication(Log)	/* Ignore */	RESOLVE ADDR	
	IrLAP_Discover.confirm(Log)	Error /* No outstanding request */	RESOLVE ADDR	
	IrLAP_NewAddress.confirm(Log) ∧ AddressConflicts (Log ∪ CacheLog)=∅ ∧ ConflictAddresses = ∅	CacheLog = CacheLog ∪ Log; LM_DiscoverDevices.confirm (status=newLog, CacheLog);	READY	5
	IrLAP_NewAddress.confirm(Log) ∧ AddressConflicts (Log ∪ CacheLog) = ∅ ∧ ConflictAddresses ≠ ∅	CacheLog = CacheLog ∪ Log; resolveAddress = ConflictAddresses[0]; ConflictAddresses = ConflictAddresses - {resolveAddress}; IrLAP_NewAddress.request (resolveAddress)	RESOLVE ADDR	
	IrLAP_NewAddress.confirm(Log) ∧ AddressConflicts (Log ∪ CacheLog) ≠ ∅ ∧ ConflictAddresses = ∅	Conflicts = AddressConflicts (CacheLog ∪ Log); CacheLog = CacheLog - Conflicts; LM_DiscoverDevices.confirm (status=newLog,CacheLog);	READY	5
	IrLAP_NewAddress.confirm(Log) ∧ AddressConflicts (Log ∪ CacheLog) ≠ ∅ ∧ ConflictAddresses ≠ ∅	Conflicts = AddressConflicts (CacheLog ∪ Log); CacheLog = CacheLog - Conflicts; resolveAddress = ConflictAddresses[0]; ConflictAddresses = ConflictAddresses - {resolveAddress}; IrLAP_NewAddress.request (resolveAddress)	RESOLVE ADDR	
	IrLAP_Primary.indication	Error /* No IrLAP Connections */	RESOLVE ADDR	
	IrLAP_Primary.confirm	Error /* No outstanding request */	RESOLVE ADDR	

State	Event	Action	Next State	
	IrLAP_Data.indication (AccessMode Request LM-PDU)	Error /* No IrLAP Connections */	RESOLVE ADDR	
	IrLAP_Data.indication (AccessMode Confirm LM-PDU)	Error /* No outstanding request */	RESOLVE ADDR	
	LM_AccessMode.request	Error /* No IrLAP connection */	RESOLVE ADDR	
	LS_Disconnect.request	Error /* No IrLAP connections */	RESOLVE ADDR	
	LS_Status.request	Error /* No IrLAP connections */	RESOLVE ADDR	
	LS_Idle.request(mode)	Error /* No IrLAP connection */	RESOLVE ADDR	
	IrLAP_Data.indication	Error /* No IrLAP connection */	RESOLVE ADDR	
	StationWDTimerExpired	/* Ignore */	RESOLVE ADDR	
	LS_Connect.request, LM_ConnectionlessData.request, LM_DiscoverDevices.request, LM_Sniff.request	***/* Left pending */***	***RESOLVE ADDR***	
EXCLUSIVE PEND	Connected = ∅	CancelStationWDTimer LocalUnlock(exclusiveLsap_ce)	READY	7
	IrLAP_Connect.confirm	Error /* No pending IrLAP Connections */	EXCLUSIVE PEND	
	IrLAP_Disconnect.indication	Forward [IrLAP_Disconnect.indication]	EXCLUSIVE PEND	
	IrLAP_Status.confirm	Forward [IrLAP_Status.confirm]	EXCLUSIVE PEND	
	IrLAP_Status.indication	Forward [IrLAP_Status.indication]	EXCLUSIVE PEND	
	IrLAP_Reset.indication	Forward [IrLAP_Reset.indication]	EXCLUSIVE PEND	
	IrLAP_Reset.confirm	Forward [IrLAP_Reset.confirm]	EXCLUSIVE PEND	
	IrLAP_Discover.indication(Log)	/* Accumulate in CacheLog */ CacheLog = CacheLog ∪ Log LM_DiscoverDevices.indication (status=passive,Log)	EXCLUSIVE PEND	
		/* Replace CacheLog */ CacheLog = Log LM_DiscoverDevices.indication (status=passive,Log)	EXCLUSIVE PEND	
	IrLAP_Discover.confirm	Error /* No outstanding request */	EXCLUSIVE PEND	
	IrLAP_NewAddress.confirm	Error /* No outstanding request */	EXCLUSIVE PEND	
	IrLAP_Primary.indication	IrLAP_Primary.response(deny=false) /* Allow the swap */	EXCLUSIVE PEND	
	IrLAP_Primary.confirm	Error /* No outstanding request */	EXCLUSIVE PEND	
	LS_Connect.request	Error /* LCC FSMs LOCKED-OUT */	EXCLUSIVE PEND	
	LS_Status.request	Forward [LS_Status.request]	EXCLUSIVE PEND	
	LS_Idle.request(mode)	Forward [LS_Idle.request(mode)]	EXCLUSIVE PEND	
	/* Access Mode race */ IrLAP_Data.indication (AccessMode Request LM-PDU [mode=exclusive]) ∧ lsap_ce = exclusiveLsap_ce	CancelStationWDTimer LM_AccessMode.confirm (status=success,mode=exclusive)	EXCLUSIVE	8
	IrLAP_Data.indication (AccessMode Request LM-PDU [mode=exclusive]) ∧ lsap_ce ≠ exclusiveLsap_ce	CancelStationWDTimer LocalUnlock(exclusiveLsap_ce); LM_AccessMode.confirm (status=raceFail,mode=multiplexed);	READY	7
	IrLAP_Data.indication (AccessMode Request LM-PDU [mode=multiplexed])	Error /* No Exclusive LSAP connection (yet!) */	EXCLUSIVE PEND	
	IrLAP_Data.indication (AccessMode Confirm LM-PDU [status=fail,mode=*])	CancelStationWDTimer LocalUnlock(exclusiveLsap_ce) LM_AccessMode.confirm (status=remoteFail, mode=multiplexed)	READY	7
	IrLAP_Data.indication (AccessMode Confirm LM-PDU [status=unsupported,mode=*])	CancelStationWDTimer LocalUnlock(exclusiveLsap_ce) LM_AccessMode.confirm (status=remoteUnsupported, mode=multiplexed)	READY	7
	IrLAP_Data.indication (AccessMode Confirm LM-PDU [status=success,mode=*])	CancelStationWDTimer LM_AccessMode.confirm (status=success, mode=exclusive)	EXCLUSIVE	8

State	Event	Action	Next State	
	LM_ConnectionlessData.request	/* Discard */ LM_ConnectionlessData.confirm (status=failed, reason=ExclusiveMode)	EXCLUSIVE PEND	
	LM_DiscoverDevices.request	LM_DiscoverDevices.confirm (status=cache,CacheLog);	EXCLUSIVE PEND	
	LM_Sniff.request(option=start)	LM_Sniff.confirm (status=refused,deviceAddress=null)	EXCLUSIVE PEND	
	LM_Sniff.request(option=cancel)	Error /* Not Sniffing */	EXCLUSIVE PEND	
	StationWDTimerExpired	/* Forcibly disconnect the sole IrLAP connection */ LM_AccessMode.confirm (status=timeOut, mode=multiplexed); Forward [LS_ForceDisconnect.request]	EXCLUSIVE PEND	
	IrLAP_Connect.indication, LS_Disconnect.request, LM_AccessMode.request	***/* Left Pending */***	***EXCLUSIVE PEND***	
EXCLUSIVE	Connected = ∅	LocalUnlock(exclusiveLsap_ce)	READY	10
	IrLAP_Connect.indication(ca)	IrLAP_Disconnect.request(ca)	EXCLUSIVE	
	IrlAP_Connect.confirm	Forward [IrLAP_Connect.confirm]	EXCLUSIVE	
	IrLAP_Disconnect.indication	Forward [IrLAP_Disconnect.indication]	EXCLUSIVE	
	IrLAP_Status.confirm	Forward [IrLAP_Status.confirm]	EXCLUSIVE	
	IrLAP_Status.indication	Forward [IrLAP_Status.indication]	EXCLUSIVE	
	IrLAP_Reset.indication	Forward [IrLAP_Reset.indication]	EXCLUSIVE	
	IrLAP_Reset.confirm	Forward [IrLAP_Reset.confirm]	EXCLUSIVE	
	IrLAP_Discover.indication(Log)	/* Accumulate in CacheLog */ CacheLog = CacheLog ∪ Log LM_DiscoverDevices.indication (status=passive,Log)	EXCLUSIVE	
		/* Replace CacheLog */ CacheLog = Log LM_DiscoverDevices.indication (status=passive,Log)	EXCLUSIVE	
	IrLAP_Discover.confirm	Error /* No outstanding request */	EXCLUSIVE	
	IrLAP_NewAddress.confirm	Error /* No oustanding request */	EXCLUSIVE	
	IrLAP_Primary.indication	IrLAP_Primary.response(deny=false) /* Allow the swap (Only 1 IrLAP connection in Exclusive mode */	EXCLUSIVE	
	IrLAP_Primary.confirm	Error /* No outstanding request */	EXCLUSIVE	
	LS_Connect.request	Error /* LCC FSMs LOCKED-OUT */	EXCLUSIVE	
	LS_Disconnect.request∧ lsap_ce = exclusiveLsap_ce	LocalUnlock(exclusiveLsap_ce) Forward [LS_Disconnect.request]	READY	10
	LS_Disconnect.request ∧ lsap_ce ≠ exclusiveLsap_ce	Forward [LS_Disconnect.request]	EXCLUSIVE	
	LS_Status.request	Forward [LS_Status.request]	EXCLUSIVE	
	LS_Idle.request(mode)	Forward [LS_Idle.request(mode)]	EXCLUSIVE	
	LM_AccessMode.request (mode=exclusive) ∧ lsap_ce = exclusiveLsap_ce	LM_AccessMode.confirm (status=success, mode=exclusive)	EXCLUSIVE	
	LM_AccessMode.request (mode=multiplexed) ∧ lsap_ce = exclusiveLsap_ce	IrLAP_Data.request (AccessMode Request LM-PDU [mode=multiplexed], expedited=false) StartStationWDTimer	READY PEND	9
	LM_AccessMode.request (mode=*) ∧ lsap_ce ≠ exclusiveLsap_ce	LM_AccessMode.confirm (status=localFailure,mode=multiplexed)	EXCLUSIVE	
	IrLAP_Data.indication (AccessMode Request LM-PDU [mode=multiplexed]) ∧ lsap_ce = exclusiveLsap_ce	LocalUnlock(exclusiveLsap_ce) LM_AccessMode.indication (mode=multiplexed); IrLAP_Data.request (AccessMode Confirm LM-PDU [status=success, mode=multiplexed], expedited=false);	READY	

State	Event	Action	Next State	
	IrLAP_Data.indication (AccessMode Request LM-PDU [mode=exclusive]) ∧ lsap_ce = exclusiveLsap_ce	Error /* Already in Exclusive Mode*/	EXCLUSIVE	
	IrLAP_Data.indication (AccessMode Request LM-PDU [mode=*])∧ lsap_ce ≠ exclusiveLsap_ce	Error /* Already in Exclusive Mode */	EXCLUSIVE	
	IrLAP_Data.indication (AccessMode Confirm LM-PDU)	Error /* No outstanding request */	EXCLUSIVE	
	LM_ConnectionlessData.request	/* Discard */ LM_ConnectionlessData.confirm (status=failed, reason=ExclusiveMode)	EXCLUSIVE	
	LM_DiscoverDevices.request	LM_DiscoverDevices.confirm (status=cache,CacheLog);	EXCLUSIVE	
	LM_Sniff.request(option=start)	LM_Sniff.confirm (status=refused,deviceAddress=null)	EXCLUSIVE	
	LM_Sniff.request(option=cancel)	Error	EXCLUSIVE	
	StationWDTimerExpired	/* Ignore */	EXCLUSIVE	
READY PEND	Connected = ∅	CancelStationWDTimer LocalUnlock(exclusiveLsap_ce)	READY	12
	IrLAP_Connect.confirm	Error /* No pending IrLAP Connections */	READY PEND	
	IrLAP_Disconnect.indication	Forward [IrLAP_Disconnect.indication]	READY PEND	
	IrLAP_Status.confirm	Forward [IrLAP_Status.confirm]	READY PEND	
	IrLAP_Status.indication	Forward [IrLAP_Status.indication]	READY PEND	
	IrLAP_Reset.indication	Forward [IrLAP_Reset.indication]	READY PEND	
	IrLAP_Reset.confirm	Forward [IrLAP_Reset.confirm]	READY PEND	
	IrLAP_Discover.indication(Log)	/* Accumulate in CacheLog */ CacheLog = CacheLog ∪ Log LM_DiscoverDevices.indication (status=passive,Log)	READY PEND	
		/* Replace CacheLog */ CacheLog = Log LM_DiscoverDevices.indication (status=passive,Log)	READY PEND	
	IrLAP_NewAddress.confirm	Error /* No outstanding request */	READY PEND	
	IrLAP_Discover.confirm	Error /* No outstanding request */	READY PEND	
	IrLAP_Primary.indication	IrLAP_Primary.response(deny=false) /* Allow the swap */	READY PEND	
	IrLAP_Primary.confirm	Error /* No outstanding request */	READY PEND	
	LS_Connect.request	Error/* LCC FSMs LOCKED-OUT */	READY PEND	
	LS_Status.request	Forward [LS_Status.request]	READY PEND	
	LS_Idle.request(mode)	Forward [LS_Idle.request(mode)]	READY PEND	
	/* Access Mode race */ IrLAP_Data.indication (AccessMode Request LM-PDU [mode=multiplexed]) ∧ lsap_ce = exclusiveLsap_ce	CancelStationWDTimer LocalUnlock(exclusiveLsap_ce) LM_AccessMode.confirm (status=success,mode=multiplexed)	READY	12
	IrLAP_Data.indication (AccessMode Request LM-PDU [mode=multiplexed]) ∧ lsap_ce ≠ exclusiveLsap_ce	Error /* New Request can't leap frog outstanding Confirm */	READY PEND	
	IrLAP_Data.indication (AccessMode Request LM-PDU [mode=exclusive])	Error /* New Request can't leap frog outstanding Confirm */	READY PEND	
	IrLAP_Data.indication (AccessMode Confirm LM-PDU [status=*, mode=*]) ∧ lsap_ce = exclusiveLsap_ce	CancelStationWDTimer LocalUnlock(exclusiveLsap_ce) LM_AccessMode.confirm (status=success,mode=multiplexed)	READY	12
	IrLAP_Data.indication (AccessMode Confirm LM-PDU) ∧ lsap_ce ≠ exclusiveLsap_ce	Error /* Wrong lsap_ce */	READY-PEND	

State	Event	Action	Next State	
	LM_ConnectionlessData.request	/* Discard */ LM_ConnectionlessData.confirm (status=failed, reason=ExclusiveMode)	READY PEND	
	LM_DiscoverDevices.request	LM_DiscoverDevices.confirm (status=cache,CacheLog);	READY PEND	
	LM_Sniff.request(option=start)	LM_Sniff.confirm (status=refused,deviceAddress=null)	READY PEND	
	LM_Sniff.request(option=cancel)	Error	READY PEND	
	StationWDTimerExpired	/* Forcibly disconnect the sole IrLAP connection */ LM_AccessMode.confirm (status=timeOut, mode=multiplexed); Forward [LS_ForceDisconnect.request]	READY PEND	
	IrLAP_Connect.indication, LS_Disconnect.request, LM_AccessMode.request,	***/* Left Pending */***	***READY PEND***	
ROLE EXCHANGE	Connected = ∅	/* Progress connection that caused the role exchange attempt */ Forward [LS_Connect]	READY	14
	IrLAP_Connect.confirm	Error /* No pending IrLAP Connections */	ROLE EXCHANGE	
	IrLAP_Disconnect.indication	/* Progress the IrLAP disconnect */ Forward [IrLAP_Disconnect.indication]	ROLE EXCHANGE	14
	IrLAP_Status.confirm	Forward [IrLAP_Status.confirm]	ROLE EXCHANGE	
	IrLAP_Status.indication	Forward [IrLAP_Status.indication]	ROLE EXCHANGE	
	IrLAP_Reset.indication	Forward [IrLAP_Reset.indication]	ROLE EXCHANGE	
	IrLAP_Reset.confirm	Forward [IrLAP_Reset.confirm]	ROLE EXCHANGE	
	IrLAP_Discover.indication(Log)	/* Accumulate in CacheLog */ CacheLog = CacheLog ∪ Log LM_DiscoverDevices.indication (status=passive,Log)	ROLE EXCHANGE	
		/* Replace CacheLog */ CacheLog = Log LM_DiscoverDevices.indication (status=passive,Log)	ROLE EXCHANGE	
	IrLAP_Discover.confirm	Error /* No outstanding request */	ROLE EXCHANGE	
	IrLAP_NewAddress.confirm	Error /* No outstanding request */	ROLE EXCHANGE	
	IrLAP_Primary.confirm(deny=false)	/* Progress connection that caused the role exchange attempt */ Forward [LS_Connect]	READY	14
	IrLAP_Primary.confirm(deny=true)	/* Reject connection that caused the role exchange attempt */ LS_Disconnect.indication (noIrLAPconnection)	READY	14
	LS_Status.request	Forward [LS_Status.request]	ROLE EXCHANGE	
	LS_Disconnect.request	Forward [LS_Disconnect.request]	ROLE EXCHANGE	
	LS_Idle.request(mode)	Forward [LS_Idle.request(mode)]	ROLE EXCHANGE	
	IrLAP_Data.indication (AccessMode Confirm LM-PDU)	Error /* No outstanding request */	ROLE EXCHANGE	
	LM_ConnectionlessData.request(data)	IrLAP_UnitData.request (Data LM-PDU [DLSAP-SEL=0x70, SLSAP-SEL=0x70, data]) LM_ConnectionlessData.confirm (status=success)	ROLE EXCHANGE	
	LM_DiscoverDevices.request	LM_DiscoverDevices.confirm (status=cache,CacheLog);	ROLE EXCHANGE	
	LM_Sniff.request	LM_Sniff.confirm (status=refused,deviceAddress=null)	ROLE EXCHANGE	
	StationWDTimerExpired	/* Ignore */	ROLE EXCHANGE	

State	Event	Action	Next State	
	IrLAP_Connect.indication, IrLAP_Primary.indication, LS_Connect.request, LM_AccessMode.request, IrLAP_Data.indication (AccessMode Request LM-PDU)	***/* Left pending */***	***ROLE EXCHANGE***	
SNIFF	IrLAP_Connect.indication	LM_Sniff.confirm(ok,DeviceAddress) Forward [IrLAP_Connect.indication]	READY	16
	IrLAP_Connect.confirm	Error /* .indication completes a sniff */	SNIFF	
	IrLAP_Disconnect.indication	Error /* No IrLAP connection */	SNIFF	
	IrLAP_Reset.indication	Error /* No IrLAP connection */	SNIFF	
	IrLAP_Reset.confirm	Error /* No outstanding request */	SNIFF	
	IrLAP_Status.confirm	Error /* No IrLAP connection */	SNIFF	
	IrLAP_Status.indication	Error /* No IrLAP connection */	SNIFF	
	IrLAP_Discover.indication(Log)	/* Accumulate in CacheLog */ CacheLog = CacheLog ∪ Log LM_DiscoverDevices.indication (status=passive,Log)	SNIFF	
		/* Replace CacheLog */ CacheLog = Log LM_DiscoverDevices.indication (status=passive,Log)	SNIFF	
	IrLAP_Discover.confirm	Error /* No outstanding request */	SNIFF	
	IrLAP_NewAddress.confirm	Error /* No outstanding request */	SNIFF	
	IrLAP_Primary.indication	Error /* No IrLAP connection */	SNIFF	
	IrLAP_Primary.confirm	Error /* No outstanding request */	SNIFF	
	LM_AccessMode.request	Error /* No IrLAP connection */	SNIFF	
	LM_ConnectionlessData.request(data)	IrLAP_Sniff.request(cancel=true) LM_Sniff(cancelled) IrLAP_UnitData.request (Data LM-PDU [DLSAP-SEL=0x70, SLSAP-SEL=0x70, data]) LM_ConnectionlessData.confirm (status=success);	READY	16
	LM_Sniff.request(option=start)	/* Ignore */	SNIFF	
	LM_Sniff.request(option=cancel)	IrLAP_Sniff.reques(cancel=true); LM_Sniff.confirm(cancelled)	READY	16
	LS_Disconnect.request	/* Error no IrlLAP connection */	SNIFF	
	LS_Status.request	/* Error No IrLAP connection */	SNIFF	
	LS_Idle.request(mode)	/* Error No IrLAP connection */	SNIFF	
	LS_Connect.request	IrLAP_Sniff.request(cancel=true); LM_Sniff.confirm(cancelled); Forward [LS_Connect.request]	READY	16
	IrLAP_Data.indication	Error /* No IrLAP connection */	SNIFF	
	LM_DiscoverDevices.request	IrLAP_Sniff.reques(cancel=true); LM_Sniff.confirm(cancelled) IrLAP_Discover.request	DISCOVER	17
	StationWDTimerExpired	/* Ignore */	SNIFF	

3.5.2.3.3 State Definitions

READY.
The station is ready for requests.

DISCOVERY.
A discovery request has been passed to IrLAP. Awaiting response from IrLAP.

SNIFF.
The IrLAP sniffing processes has been initiated. Awaiting response from IrLAP.

RESOLVE ADDR.
One or more address conflicts were present in the discovery Log. IrLAP address resolution process has been started to resolve one of the conflicts.

EXCLUSIVE PEND
A request is being processed for entering exclusive mode. Local conditions have been met and a request has been sent to the remote device. Awaiting response from remote device.

EXCLUSIVE.
The station is in exclusive mode. Only one active LSAP is allowed.

READY PEND
A request has been made to leave exclusive mode. The request has been sent to the remote device and the local device is awaiting its response (the remote device cannot refuse).

ROLE EXCHANGE.
An implicit request to exchange roles (typically an attempt to establish a multipoint link). This is currently not supported in IrLAP.

3.5.2.3.4 State Variables and Functions

AddressConflicts(S)
Function whose value is the set of conflicting deviceInfo record present in the set S of deviceInfo records

AllLsapConnectionEndPoints
Function whose value is a set of references to ALL the LSAP-connection endpoints present within an LM-MUX entity, both those associated with an IrLAP connection and those that are passively waiting for a connection to be established.

CacheLog
Set variable of deviceInfo records obtained by the most recent discover process, possibly augmented with additional unsolicited records. This document specifies no strict caching policy. This set is maintained by the Station Control FSM and shared with instances of the ICC FSM.

ConflictAddresses
Set variable of IrLAP device addresses. Used to hold the result of ExtractAddresses(), below, and gradually emptied by address resolution. All set members are distinct (there are no duplicates).This variable is local to the Station Control FSM.

Conflicts
Set variable of deviceInfo records. Used to hold the result of AddressConflicts() above. This variable is local to the Station Control FSM.

Connected
Set variable of IrLAP device addresses. Used to hold the device address of each IrLAP peer device connected to the local device. This variable is maintained by the ICC FSMs and shared with the Station Control FSM.

exclusiveLsap_ce
Variable referencing the local LSAP-connection endpoint that holds locked access to the single IrLAP connection allowed in exclusive mode. This variable is local to the Station Control FSM.

stationMode
Variable shared with LSAP-connection control FSM to signal whether LM-MUX is in exclusive or multiplexed mode. This variable is modified by the ***LocalLock*** and ***LocalUnlock*** actions below.

ExtractAddresses(S)
Function whose value is a set of distinct IrLAP device addresses extracted from the set S of deviceInfo records.

IdleIrLAPConnections
Function whose value is a set of references to IrLAP connection endpoints for IrLAP connections that have no active LSAP-connection endpoints associated with them. LSAP-connection endpoints are marked active or idle by the LM_Idle service. The LCC FSM of a idle LSAP-connection endpoint is in either the DISCONNECTED or DTR_IDLE state. Strictly an LCC FSM in the DISCONNECTED state is not associated with any IrLAP connection.

LocalLockable(x)
Function whose boolean value evaluates to true if local conditions permit the LSAP-connection endpoint referenced by x to obtained exclusive access to the underlying IrLAP connection. The logic of LocalLockable is expressed as follows:

```
LocalLockable(locking_lsap_ce)
/* Check locking_lsap_ce is for established connection */
  if(locking_lsap_ce.LSAPConnectionContolFsm.state ≠ DTR)
    return(false);

  result = true;
  ∀ ( y ∈ AllLsapConnectionEndpoints)
    If (y ≠ locking_lsap_ce)
      If ( y.LSAPConnectionContolFsm.state ≠ DTR_IDLE ∧
           y.LSAPConnectionContolFsm.state ≠ DISCONNECTED )
        result = false;
return(result);
```

LSAP-connection endpoints whose LCC FSM is in the DTR-IDLE state are at one end of an established LSAP-connection that has been marked as locally idle using the LM_Idle service. LSAP-connection endpoints whose LCC FSM is in the DISCONNECTED state are NOT are not associated with any IrLAP connection and are passively waiting a connection.

Log
Set variable of deviceInfo records returned by IrLAP_Discover.indication, IrLAP_Discover.confirm and IrLAP_NewAddress.confirm primitives. The variable is local to the Station Control FSM.

lsap_ce
Variable containing a reference to the local LSAP connection endpoint associated with the current transition (where applicable). This variable is implictly set for each transition and is local to the Station Control FSM

resolveAddress
Variable containing the device address of the most resently attempted address resolution. This variable is local to the Station Control FSM.

3.5.2.3.5 Event Descriptions

MultiPointSupportEnabled
Predicate: The station is configured for/capable of point-to-multipoint operation.

!MultiPointSupportEnabled
Predicate: The station is not configure for/capable of point to multipoint operation, ie. the station currently supports a single point-to-point IrLAP connection.

AddressConficts(Log) „ ˇ
Predicate: There are conflicting deviceInfo records in the set of deviceInfo records named Log

AddressConficts(Log) = ˇ
Predicate: There are NO conflicting deviceInfo records in the set of deviceInfo records named Log

AddressConflicts(Log ¨ ***CacheLog)*** „ ˇ
Predicate: There are conflicting deviceInfo records in the union of the sets of deviceInfo records named Log and CacheLog.

AddressConflicts(Log ¨ ***CacheLog)*** = ˇ
Predicate: There are NO conflicting deviceInfo records in the union of the sets of deviceInfo records named Log and CacheLog.

ConflictAddresses = ˇ
Predicate: There are NO (remaining) conflicting device addresses during address resolution.

ConflictAddresses „ ˇ
Predicate: There are (remaining) conflicting device addresses during address resolution.

Connected „ ˇ
Predicate: There are NO established IrLAP Connections at this station. This predicate is used on its own in some states (ie. it is not coupled with an event). This form of use is particularly useful in the EXCLUSIVE PEND, EXCLUSIVE, READY PEND and ROLE EXCHANGE states as it picks up the disconnection of the IrLAP connection due to actions by the appropriate ICC FSM.

#Connected = 1
Predicate: There is exactly one established IrLAP connection at this station.

#Connected > 1
Predicate: There is more than one established IrLAP connection at this station.

deviceAddress ¸ ***Connected***
Predicate: There is an IrLAP connection to the device that currently has the address deviceAddress.

IdleIrLAPConnections = ˇ
Predicate: There are no established IrLAP connections that currently are supporting only idle LSAP-connections, ie. LSAP-connection that have had been set idle using the LM_Idle service.

IrLAP_Connect.confirm
Receipt of an IrLAP_Connect.confirm primitive from an underlying IrLAP connection endpoint.

IrLAP_Connect.indication
Receipt of an IrLAP_Connect.indication primitive from an underlying IrLAP connection endpoint.

IrLAP_Data.indication(AccessMode Confirm LM-PDU)
Receipt of an AccessMode Confirm LM-PDU.

IrLAP_Data.indication(AccessMode Request LM-PDU)
Receipt of an AccessMode Request LM-PDU.

IrLAP_Disconnect.indication
Receipt of an IrLAP_Disconnect.indication primitive from an underlying IrLAP connection endpoint.

IrLAP_Discover.confirm
Receipt of an IrLAP_Disconnect.confirm primitive from an underlying IrLAP connection endpoint.

IrLAP_Discover.indication
Receipt of an IrLAP_Discover.indication primitive from an underlying IrLAP connection endpoint.

IrLAP_NewAddress.confirm
Receipt of an IrLAP_NewAddress.confirm primitive from an underlying IrLAP connection endpoint.

IrLAP_Primary.confirm
Receipt of an IrLAP_Primary.confirm primitive from an underlying IrLAP connection endpoint.

IrLAP_Primary.indication
Receipt of an IrLAP_Primary.indication primitive from an underlying IrLAP connection endpoint.

IrLAP_Reset.confirm
Receipt of an IrLAP_Reset.confirm primitive from an underlying IrLAP connection endpoint.

IrLAP_Reset.indication
Receipt of an IrLAP_Reset.indication primitive from an underlying IrLAP connection endpoint.

IrLAP_Status.confirm
Receipt of an IrLAP_Status.confirm primitive from an underlying IrLAP connection endpoint.

IrLAP_Status.indication
Receipt of an IrLAP_Status.indication primitive from an underlying IrLAP connection endpoint.

LM_AccessMode.request
Receipt of an LM_AccessMode.request primitive from an LSAP connection endpoint.

LM_ConnectionlessData.request
Receipt of an LM_ConnectionlessData.request primitive from the Connectionless Service Access Point.

LM_DiscoverDevices.request
Receipt of an LM_DiscoverDevices.request primitive from the XID_Discovery Service Access Point.

LM_Sniff.request
Receipt of an LM_Sniff.request primitive from the XID_Discovery Service Access Point.

LocalLockable(lsap_ce) = false
Predicate: Local conditions DO NOT allow the LSAP-connection endpoint referenced by lsap_ce to enter exclusive mode.

LocalLockout(lsap_ce) = true
Predicate: Local conditions allow the LSAP-connection endpoint referenced by lsap_ce to enter exclusive mode.

LS_Connect.request
Receipt of an internal LS_Connect.request primitive received from an LSAP-connection control FSM.

LS_Disconnect.request
Receipt of an internal LS_Disconnect.request primitive received from an LSAP-connection control FSM.

LS_Status.request
Receipt of an internal LS_Status.request primitive received from an LSAP-connection control FSM.

LS_Idle.request(mode)
Receipt of an internal LS_Idle.request primitive received from an LSAP-connection control FSM.

lsap_ce „ exclusiveLsap_ce
Predicate: The local LSAP-connection endpoint causing associated with the event DOES NOT hold a local lock on the IrLAP connection.

lsap_ce = exclusiveLsap_ce
Predicate: The local LSAP-connection endpoint causing associated with the event holds a local lock on the IrLAP connection.

StationWDTimerExpired
The Watchdog Timer used to monitor transitions too and from exclusive mode has expired.

3.5.2.3.6 Action Descriptions

" *IrLAP Connections ¸ IdleIrLAPConnections Forward [LS_ForceDisconnect]*
Forcibly disconnect any IrLAP connections that have no active LSAP-connections using them ie. there are now LSAP-connections associated with the IrLAP-connection or ALL LSAP-connections associated with the IrLAP-connection are marked idle (corresponding LSAP-Connection Control FSM is in the DTR-IDLE state, see section 3.5.4)

CacheLog = CacheLog ¨ Log;
Append the set of newly discovered deviceInfo records (Log) to the set of cached deviceInfor records (CacheLog).

CacheLog = CacheLog - Conflicts;
Remove the set of conflicting deviceInfo records (Conflicts) from the set of cached deviceInfo records (CacheLog)

CacheLog = Log;
Replace the set of cached deviceInfo records (CacheLog) with the set of newly discovered deviceInfo records.

ConflictAddresses = ConflictAddresses - {resolveAddress};
Remove the IrLAP device address (resolveAddress) used in the previous address resolution cycle from the set of conflicting device addresses (ConflictAddresses).

ConflictAddresses = ExtractAddresses(Conflicts);
Extract the set of distinct IrLAP addresses from the set of (non-distinct) conflicting deviceInfo records and assign the result to the set of conflicting device addresses (ConflictAddresses)

Conflicts = AddressConflicts(Log ¨ CacheLog);
Extract the set of conflicting deviceInfo records (records that do not have distinct IrLAP device addresses) from the union of the newly discovered and previously cached sets of such records..

Conflicts = AddressConflicts(Log);
Extract the set of conflicting deviceInfo records from the set of newly discovered record. The result is assigned to the set Conflicts.

Error
This action marks an unexpected or illegal transition. It is not expected that any of these transitions will arise in a correct implementation. Some cases a 'safe' a response is generated by sucessive action statements. In most cases the events causing the transition are 'silently' ignored.

exclusiveLsap_ce = lsap_ce;
Maintain a reference to the LSAP-connection endpoint that has locally established exclusive access to the medium.

Forward [IrLAP_*] and ***Forward [LS_*]***
Send an event received from an IrLAP connection endpoint or an internal LS_ event to the appropriate IrLAP Connection Control (ICC) FSM (see section 3.5.3). The ICC FSMs leave no events pending and the state transition and accompanying actions of the ICC FSM are regarded as having occurred before the corresponding Forward action completes. ie. the Station Control FSM MAY NOT process the next action in a list of actions or attend to the next event arrival until the ICC FSM has complete the transition associated with the forwarded event. This is important in order to maintain the consistency of data shared between Station Control and the ICC FSMs.

IrLAP_Data.request (AccessMode Confirm LM-PDU [mode], expedited=false)
Send an Access Mode Confirm LM-PDU.

IrLAP_Data.request(Data LM-PDU [DLSAP-SEL=0x70, SLSAP-SEL=0x70, data], expedited=true])
Send a ConnectionlessData as unacknowledged expedited data within an IrLAP connection at the connection negotiated data rate.

IrLAP_Disconnect.request
Reject an incomming IrLAP connection due lack of resource to support it. This arises in the READY state if it is not possible to assign a ICC FSM to an incomming IrLAP connection or if an incomming connection is reported whilst in the DISCOVER or RESOLVE ADDR states.

IrLAP_Discover.request
Invoke IrLAP_Discover.request service primitive.

IrLAP_NewAddress.request
Invoke IrLAP_NewAddress.request service primitive.

IrLAP_Primary.request
Invoke IrLAP_Primary.request service primitive.

IrLAP_Primary.response
Invoke IrLAP_Primary.response service primitive.

IrLAP_Sniff.request
Invoke IrLAP_Sniff.request service primitive.

IrLAP_UnitData.request(Data LM-PDU [DLSAP-SEL=0x70, SLSAP-SEL=0x70, data])
Send a ConnectionlessData as unacknowledged broadcast data outside an IrLAP connection at the contention data mode data rate.

LM_AccessMode.confirm
Invoke the IrLMP LM_AccessMode.confirm primitive at the local LSAP_connection endpoint that caused the transition or refenced in the AccessMode LM-PDU that caused the transition (which should be identical to that referenced by exclusiveLsap_ce).

LM_AccessMode.indication
Report an AccessMode change requested by the remote end to the local LSAP-connection endpoint affected by the change.

LM_DiscoverDevices.confirm
Invoke an LM_DiscoverDevices.confirm primitive at the XID_Discover Service Access Point.

LM_DiscoverDevices.indication
Invoke and LM_DiscoverDevices.indication primitive at the XID_Discover Service Access Point.

LM_Sniff.confirm
Invoke an LM_Sniff.confirm primitive at the XID_Discover Service Access Point.

LocalLock(exclusiveLsap_ce)
Disable all local LSAP connection endpoints from accessing their supporting IrLAP connection with the exception of the endpoint referred to by exclusiveLsap_ce. The logic of this operation is as follows:

```
LocalLock (locking_lsap_ce)
  stationMode=exclusive;
  ∀ ( lsap_ce ∈ AllLsapConnectionEndpoints)
    If (lsap_ce ≠ locking_lsap_ce)
      LS_LockOut.request(lockOut=true);
        // Always succeed because action is guarded by LocallyLockable.
```

Note - Each instance of an ICC FSM maintains a distinct set named Associated per ICC FSM see section 3.5.3

LocalUnlock(exclusiveLsap_ce)
Re-enable all LSAP-connection endpoints (that have not subsequently disconnected) previously disabled bu LocalLock above. The logic of this operation is as follows:

```
LocalUnlock (locking_lsap_ce)
  stationMode=multiplexed
  ∀ ( lsap_ce ∈ AllLsapConnectionEndpoints)
    If (lsap_ce ≠ locking_lsap_ce)
      LS_LockOut.request(lockOut=false);
```

Note - Each instance of an ICC FSM maintains a distinct set named Associated per ICC FSM see section 3.5.3

LS_Disconnect.indication
Reject an LS_Connection.request due either because it was not possible to take the primary role in order to open the necessary IrLAP-connection or because point-to-multipoint capability is not enabled.

resolveAddress = ConflictAddresses[0];
Select one IrLAP device address from the set of conflicting IrLAP device addresses.

StartStationWDTimer:
Start (or restart) the watchdog timer used to monitor the transition to or from exclusive mode. If the timer expires during a mode transition the underlying IrLAP connection will be forcibly closed.

CancelStationWDTimer
Stop the station watchdog timer. This is invoked when the event that is being sought occurs before the watchdog timer expires. This occurs on all transitions from EXCLUSIVE PEND or from READY PEND that cause a change of state.

3.5.3 IrLAP Connection Control

3.5.3.1 Purpose

An instance of an IrLAP connection control FSM (ICC FSM) assists in the establishment of an IrLAP-connection and the association of LSAP connections with that IrLAP link. When all LSAP-connections associated with the IrLAP connection cease, the (ICC FSM) ensures that the IrLAP connection is disconnected (it is implementation behavior as to how soon after the the last LSAP-Connection disconnects and the IrLAP link is disconnected).

3.5.3.2 Overview

An instance of the ICC FSM is associated with each IrLAP connection. It maintains and association between the LSAP-connections using a given IrLAP connection and the IrLAP connection itself. The ICC FSM is initialised to the STANDBY state.

In th STANDBY state the ICC FSM is waiting for either:

1. a local request to establish an IrLAP connection which will occur as a result of an LM_Connect.request having been invoked at a local LSAP-connection endpoint, or

2. an incomming IrLAP connection as a result of a similar action at a remote peer.

A local request causes the invocation of an IrLAP_Connection.request at the IrLAP connection end-point associated with the ICC FSM and the inclusion of the LSAP-connection that caused the action in the set of associated LSAP-connection endpoints. The ICC FSM transitions to the U_CONNECT state.

An incomming IrLAP connection, signal via an IrLAP_Connect.indication primitive is accepted unconditionally and the ICC FSM transitions to the ACTIVE state.

In the U_CONNECT state the ICC FSM awaits the outcome of an attempt to form an IrLAP connection. Whilst in this state, requests to open an IrLAP connection (LS_Connect.request) to the same destination result in the requesting LSAP-connection endpoint being added to the set of associated LSAP connection endpoints. If the IrLAP connection is refused an LS_Disconnect.indication is sent to the LCC FSM of each associated LSAP-connection endpoint. This rejects the corresponding LM_Connect.request, The ICC clears the set of associated LSAP connection endpoints and transitions to the STANDBY state.

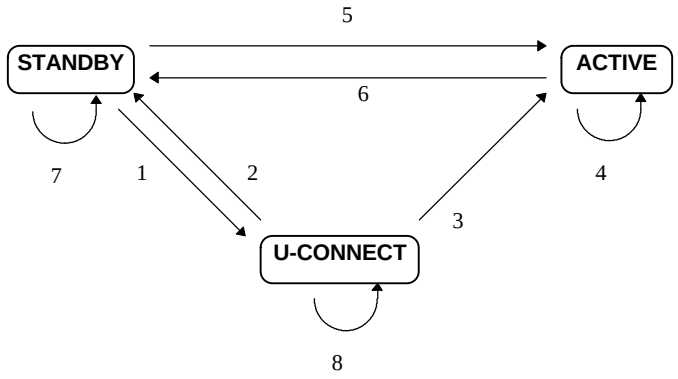
If the IrLAP connection is successfully established an LS_Connect.confirm is invoked at the LCC FSM of each LSAP-connection endpoint associated with the IrLAP connection. This indicates the availability of an open IrLAP connection. The ICC FSM transitions to the active state.

In the ACTIVE state the ICC FSM continues to associate new LSAP-connection endpoints with the IrLAP connection. Likewise it also removes associations in response to LS_Disconnect.request primitives from LSAP-connections that have ceased to use the IrLAP connection. When there are no LSAP connections associated with a given IrLAP connection the ICC FSM may disconnect the inderlyiing IrLAP connection and return to the STANDBY state or it may hold the connection open in anticipation of future LSAP-connections.

In both the ACTIVE and the U-CONNECT state LS_Status.request primitives are serviced by the ICC FSM. On the first such request an IrLAP_Status.request is invoked. A set of all LSAP-connection endpoints requesting status is maintained and the resulting status is returned to the LCC FSM at all requesting LSAP-connection endpoints. Similarly, unsolicitited IrLAP_Status.indications are forwarded to the LCC FSMs of all associated LSAP connection endpoints.

3.5.3.3 Precise Description

3.5.3.3.1 IrLAP Connection Control State Transition Diagram



3.5.3.3.2 IrLAP Connection Control State Transition Table

State	Event	Action	Next State	
STANDBY	IrLAP_Connect.indication (srcDeviceAddress)	peerDevice=srcDeviceAddress IrLAP_Connect.response Associated=Ø Idle=Ø Connected = Connected ∪ {peerDeviceAddress}	ACTIVE	5
	IrLAP_Connect.confirm	Error	STANDBY	7
	IrLAP_Disconnect.indication	Error	STANDBY	7
	IrLAP_Status.indication	Error	STANDBY	7
	IrLAP_Status.confirm	Error	STANDBY	7
	IrLAP_Reset.indication	Error	STANDBY	7
	IrLAP_Reset.confirm	Error	STANDBY	7
	LS_Connect.request (dstDeviceAddress)	peerDevice=dstDeviceAddress IrLAP_Connect.request Associated = {lsap_ce} Idle=Ø Connected = Connected ∪ {peerDeviceAddress}	U-CONNECT	1

State	Event	Action	Next State	
	LS_Idle.request(mode=*)	Error /* No LSAP Connections */	STANDBY	7
	LS_Disconnect.request	Error	STANDBY	7
	LS_Status.request	Error	STANDBY	7
U-CONNECT	IrLAP_Connect.indication	IrLAP_Connect.response ∀ lsap_ce ∈ Associated LS_Connect.confirm /*Should never occur as IrLAP resolves IrLAP connection races */	ACTIVE	3
	IrLAP_Connect.confirm	∀ lsap_ce ∈ Associated LS_Connect.confirm	ACTIVE	3
	IrLAP_Disconnect.indication	∀ lsap_ce ∈ Associated LS_Disconnect.indication Associated=Ø Connected = Connected-{peerDeviceAddress}	STANDBY	2
	IrLAP_Status.indication	∀ lsap_ce ∈ Associated LS_Status.indication	U-CONNECT	8
	IrLAP_Status.confirm	∀ lsap_ce ∈ StatusPending LS_Status.confirm StatusPending=Ø	U-CONNECT	8
	IrLAP_Reset.indication ∨ LS_ForceDisconnect.request	Error /* Should not occur before the IrLAP connection is established */	U-CONNECT	8
	IrLAP_Reset.confirm	Error	U-CONNECT	8
	LS_Connect.request	Associated=Associated ∪ {lsap_ce}	U-CONNECT	8
	LS_Disconnect.request ∧ Associated={lsap_ce}	IrLAP_Disconnect.request Associated=Ø Idle=Ø Connected = Connected-{peerDeviceAddress}	STANDBY	2
	LS_Disconnect.request ∧ {lsap_ce} ⊂ Associated	Associated=Associated-{lsap_ce}	U-CONNECT	8
	LS_Status.request ∧ StatusPending = Ø	IrLAP_Status.request StatusPending={lsap_ce}	U-CONNECT	8
	LS_Status.request ∧ StatusPending ≠ Ø	StatusPending= StatusPending ∪ {lasp_ce}	U-CONNECT	8
	LS_Idle.request(mode=*)	**/* Left Pending */**	U-CONNECT	8
ACTIVE	IrLAP_Connect.indication	Error	ACTIVE	4
	IrLAP_Connect.confirm	Error	ACTIVE	4
	IrLAP_Disconnect.indication	∀ lsap_ce ∈ Associated LS_Disconnect.indication Associated=Ø Idle=Ø Connected = Connected-{peerDeviceAddress}	STANDBY	6
	IrLAP_Status.indication	∀ lsap_ce ∈ Associated LS_Status.indication	ACTIVE	4
	IrLAP_Status.confirm	∀ lsap_ce ∈ StatusPending LS_Status.confirm StatusPending=Ø	ACTIVE	4
	IrLAP_Reset.indication ∨ LS_ForceDisconnect.request	IrLAP_Disconnect.request ∀ lsap_ce ∈ Associated LS_Disconnect.indication(irLapReset) Associated=Ø Idle=Ø Connected = Connected-{peerDeviceAddress}	STANDBY	6
	IrLAP_Reset.confirm	Error	ACTIVE	4
	LS_Idle.request(mode=active)	Idle=Idle - {lsap_ce} CancelIdleMonitor	ACTIVE	4
	LS_Idle.request(mode=idle) ∧ |Associated| ≠ |Idle|+1	Idle=Idle ∪ {lsap_ce}	ACTIVE	4

State	Event	Action	Next State	
	LS_Idle.request(mode=idle) ∧ |Associated| = |Idle|+1	Idle=Idle ∪ {lsap_ce} StartIdleMonitor	ACTIVE	4
	LS_Connect.request	Associated=Associated ∪ {lsap_ce} LS_Connect.confirm CancelIdleMonitor	ACTIVE	4
	LS_Disconnect.request ∧ Associated={lsap_ce}	Associated=Ø Idle=Ø StartIdleMonitor	ACTIVE	4
	LS_Disconnect.request ∧ {lsap_ce} ⊂ Associated ∧ {lsap_ce} ⊄ Idle ∧ |Associated| ≠ |Idle| + 1	Associated=Associated-{lsap_ce}	ACTIVE	4
	LS_Disconnect.request ∧ {lsap_ce} ⊂ Associated ∧ {lsap_ce} ⊄ Idle ∧ |Associated| = |Idle| + 1	Associated=Associated-{lsap_ce} StartIdleMonitor	ACTIVE	4
	LS_Disconnect.request ∧ {lsap_ce} ⊂ Associated ∧ {lsap_ce} ⊂ Idle	Associated=Associated-{lsap_ce} Idle=Idle - {lsap_ce}	ACTIVE	4
	LS_Status.request ∧ StatusPending = Ø	IrLAP_Status.request StatusPending={lsap_ce}	ACTIVE	4
	LS_Status.request ∧ StatusPending ≠ Ø	StatusPending= StatusPending ∪ {lasp_ce}	ACTIVE	4
	IdleMonitorExpired ∧ |Associated| = |Idle|	∀ lsap_ce ∈ Associated LS_Disconnect.indication IrLAP_Disconnect.request Associated=Ø Idle= Ø Connected = Connected-{peerDeviceAddress}	STANDBY	6
	IdleMonitorExpired ∧ |Associated| ≠ |Idle|	/* A connection has become active */	ACTIVE	4

3.5.3.3.3 State Definitions

STANDBY.
The IrLAP connection does not exist.

U_CONNECT.
Service user implicit IrLAP connection request initiated as a result of trying to open an LSAP connection. Awaiting response from IrLAP.

ACTIVE.
IrLAP connection is active.

3.5.3.3.4 State Variables

Associated
Set variable of references to LSAP connection endpoints associated with this instance of the ICC FSM. Each instance of the ICC FSM maintains a distinct instance of this variable and it is shared with the Station Control FSM.

Connected
Set variable of IrLAP device addresses. Used to hold the device address of each IrLAP peer device connected to the local device. This variable is maintained by the all ICC FSMs and shared with the Station Control FSM.

Idle
Set variable of references to idle LSAP connection endpoints associated with this instance of the ICC FSM.

StatusPending
Set variable of references to LSAP-connection endpoints waiting a response to an earlier LM_Status.request. Each instance of the ICC FSM maintains a distinct instance of this variable.

lsap_ce
Variable containing a reference to the local LSAP connection endpoint associated with the current transition (where applicable). This variable is implictly set for each transition and is local to the instance of the ICC FSM.

peerDeviceAddress
Variable containing the IrLAP device address of the IrLAP peer associated with the IrLAP connection monitored by this instance of the ICC FSM. Each instance of the ICC FSM maintains a distinct instance of this variable.

3.5.3.3.5 Event Descriptions

Associated={lsap_ce}
The LSAP-Connection endpoint that invoked the event is the ONLY one associated with the underlying IrLAP-connection.

{lsap_ce} ì ***Associated***
The LSAP-Connection endpoint that invoked the event is NOT the ONLY LSAP-connection associated with the underlying IrLAP-connection.
A ⊂ **B** means: set **A** is a strict subset of **B**.

StatusPending „ ***Ø***
There are LSAP-connections awaiting the result of an LS_Status request directed at the underlying IrLAP-connection.

StatusPending = Ø
There are no LSAP-connections awaiting the result of an LS_Status request directed at the underlying IrLAP-connection.

IrLAP_Connect.confirm.
Event forwarded from Station Control indicating that the underlying IrLAP connection has been established.

IrLAP_Connect.indication.
Event forwarded from Station Control requesting that the underlying IrLAP connection be established.

IrLAP_Disconnect.indication.
Event forwarded from Station Control indicating that the underlying IrLAP connection has disconnected.

IrLAP_Reset.indication.
Event forwarded from Station Control.

IrLAP_Reset.confirm.
Event forwarded from Station Control.

IrLAP_Status.confirm.
Event forwarded from Station Control re: Unacked data.

IrLAP_Status.indication.
Event forwarded from Station Control.

LS_Connect.request.
Request forwarded from Station Sontrol to associate the LSAP-connection endpoint with an IrLAP connection.

LS_Disconnect.request.
Request forwarded from station control to end the association between an LSAP-connection and an IrLAP-connection.

LS_Status.request.
Request forwarded from station control to invoke IrLAP_Status.request on the underlying IrLAP-connection.

LS_ForceDisconnect.request
Request from Station Control FSM to forcibly disconnect an IrLAP connection.

LS_Idle.request(mode).
Internal event forwarded from station control to inform the IrLAP connection control that the mode of an associated LSAP endpoint has changed.

IdleMonitorExpired
The implementation dependant method of determining how long an IrLAP connection should remain if only idle LSAP connections are associated with it has determined that it is now time to close the IrLAP connection.

3.5.3.3.6 Action Descriptions

" *lsap_ce ¸ Associated*
Apply the following action(s) (shown indented) to all LSAP-connection endpoints associated with the undelying IrLAP-connection.

" *lsap_ce ¸ StatusPending*
Apply the following action(s) (shown indented) to all LSAP-connection endpoints that have request the status of the underlying IrLAP connection.

Associated = {lsap_ce}
Initialise the set of LSAP-connection endpoints associated with the underlying IrLAP connection with a reference to the LSAP-connection endpoint that caused the transition.

Associated = Associated ¨ {lsap_ce}
Insert a reference to the LSAP-connection endpoint that caused the transition in the set of LSAP-connection endpoints associated with (using) the underlying IrLAP connection.

Associated = Associated-{lsap_ce}
Remove the reference to the LSAP-connection endpoint that caused the transition from the set of LSAP-connection endpoints associated with (using) the underlying IrLAP connection

Associated = Ø
Empty the set of LSAP-connection endpoints associated with the underlying IrLAP connection.

Connected = Connected ¨ {peerDeviceAddress}
Insert the peer device address in the (per Station) set of device addresses connected (or partially connected) by IrLAP.

Connected = Connected - {peerDeviceAddress}
Remove the peer device address from the (per Station) set of device addresses connected (or partially connected) by IrLAP.

Idle = Idle ¨ {lsap_ce}
Insert a reference to the LSAP-connection endpoint that caused the transition in the set of LSAP-connection endpoints associated with (using) the underlying IrLAP connection but have been marked idle by the user.

Idle = Idle-{lsap_ce}
Remove the reference to the LSAP-connection endpoint that caused the transition from the set of idle LSAP-connection endpoints associated with (using) the underlying IrLAP connection

Idle = Ø
Empty the set of idle LSAP-connection endpoints associated with the underlying IrLAP connection.

StatusPending = {lsap_ce}
Initialise the set of LSAP-connection endpoints waiting the status of underlying IrLAP connection with a reference to the LSAP-connection endpoint that caused the transition.

StatusPending = StatusPending ¨ {lasp_ce}
Insert a reference to the LSAP-connection endpoint that caused the transition into the set of LSAP-connection endpoints awainting status from the underlying IrLAP connection.

StatusPending = Ø
Empty the set of LSAP-connection endpoints awaiting status from the underlying IrLAP connection.

Error
An unexpected or illegal transition has occured. These are internal to an IrLMP LM-MUX and are simply ignored. They result in no change in the state of the LCC FSM.

IrLAP_Connect.request
IrLAP Service Primitive that requests the establishment of an IrLAP connection.

IrLAP_Connect.response
IrLAP Service Primitive that accepts an incoming IrLAP connection reported via an IrLAP_Connect.indication.

IrLAP_Disconnect.request
IrLAP Service Primitive that requests the closure of the underlying IrLAP connection. This primitive may also be used to reject an IrLAP connection however IrLMP never uses this primitive for that purpose.

IrLAP_Status.request
IrLAP Service Primitive that requests the status of the underlying IrLAP connection. This primitive is used to report whether all outstanding data has been acknowledged at the IrLAP level.

LS_Connect.confirm
Internal Service Primitive that indicates the availability of the IrLAP connection to which an LSAP-connection is bound.

LS_Disconnect.indication
Internal Service Primitive that reports the disconnection of an underlying IrLAP connection to an LSAP-connection endpoint was using the IrLAP connection.

LS_Status.confirm
Pass through that conveys an IrLAP_Status.confirm to an LSAP-connection endpoint that awaits a response to an earlier LM_Status.request.

LS_Status.indication
Pass through that conveys an IrLAP_Status.indication to an LSAP-connection endpoint associated with the underlying IrLAP connection.

StartIdleMonitor
This action begins an implementation dependent method of determining how long the IrLAP connection control FSM should maintain the IrLAP connection when only idle LSAP connections are associated with it. The main aim of this monitor is to prevent an LSAP connection from keeping the IrLAP connection active for long periods of time even when the LSAP connection being marked idle. Several options are available when implementing the IdleMonitor process. The process could return instantly, thereby disconnecting the IrLAP connection as soon as all LSAP connections are marked idle. Alternatively, the process could never return and therefore maintain the IrLAP connection indefinitely. However, a more useful process could implement a simple timer function which, after a certain period of time, say 2 seconds, the monitor would expire and trigger the disconnection of the IrLAP connection. Another method would allow the IdleMonitor to monitor the quality of the IrLAP connection. If the quality begins to deteriorate, possibly due to the devices moving out of range, the IdleMonitor could trigger the disconnection of the IrLAP link and hence avoiding the delay required by IrLAP to recover from a broken link.

CancelIdleMonitor
Cancels the process started by StartIdleMonitor and hence prevents the occurrence of the associated event IdleMonitorExpired..

3.5.4 LSAP-Connection Control

3.5.4.1 Purpose

An instance LSAP-connection control FSM maintains the state of an LSAP-connection that terminates within the station. There is one instance of this FSM for each LSAP-connection endpoint within the station

The FSM also provides the LM_Idle service and participates in the establishment and enforcement of the LM-MUX exclusive mode.

3.5.4.2 Overview

An instance of this FSM is associated with each LSAP-connection endpoint within a station. It is initialized into the DISCONNECTED state.

Active opening of the LSAP-connection commences with the receipt of an LM_Connect.request primitive from the LSAP-connection endpoint. This causes invocation of an LS_Connect.request to associate the resulting LSAP-connection (if successfully established) with its supporting IrLAP connection. This is accompanied by a transition to SETUP-PENDING.

LS_Connect.request is serviced by station control which will attempt to establish an IrLAP connection. If a suitable IrLAP connection exists station control signals the availability of a suitable IrLAP connection by the invocation of an LS_Connect.confirm event at the FSM. This causes the FSM to send a Connect LM-PDU to its intended peer and transition to the SETUP state awaiting the return of a Connect Confirm LM-PDU from its peer. If Station control was unable to associate the LSAP-connection endpoint with an IrLAP connection it returns a LS_Disconnect.indication (reason=noIrLAPConnection). The FSM transitions back to the DISCONNECTED state and issues an LM_Disconnect.indication back to the service user.

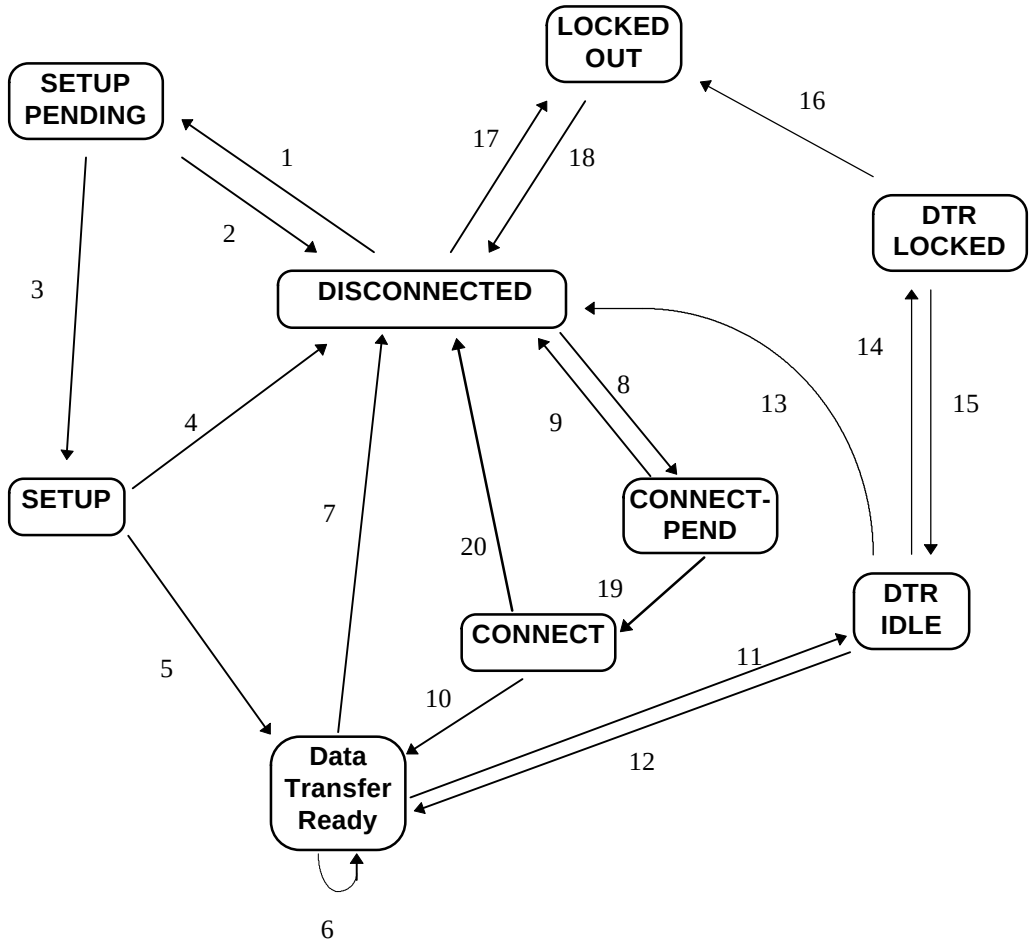
Upon receipt of a Connection Confirm LM-PDU the FSM issues an LM_Connect.confirm to the service user and transitions to the DTR state. User data may now be exchanged over the LSAP-connection through the use of LM_Data and LM_UData services. If a Connect LM-PDU arrives whilst the FSM is in the SETUP state a connection race occurs and the FSM transitions back to the DISCONNECTED state and issues an LM_Disconnect.indication(reason=connectionRace) to the service user. Likewise an LS_Disconnect.indication or the arrival of an Disconnect LM-PDU also cause the failure of the LSAP-connection.

Passive establishment of an LSAP-connection occurs when a Connect LM-PDU arrives at an FSM in the DISCONNECTED state. It first issues an LS_Connect.request to bind the LSAP-connection endpoint to the supporting IrLAP connection and transitions to the CONNECT-PEND state. Upon receipt of an LS_Connect.confirm (from station control) the incoming LSAP connection is signaled to the service user by an LM_Connect.indication and the state transitions to CONNECT awaiting a response from the LM-MUX client. If the user returns an LM_Disconnect.request then the FSM returns a Disconnect LM-PDU to its peer, reason=userRequest, issues an LS_Disconnect.request to dissociate itself from the supporting IrLAP connection and transitions to the DISCONNECTED state. If the user accepts the incoming LSAP connection by issuing an LM_Connect.response a Connect Confirm LM-PDU is sent to the peer FSM and the local FSM transitions to the DTR state. Once again user data may now be exchanged on the resulting LSAP connection through the use of LM_Data and LM_UData services.

The LM_Idle service is implemented by transitions between DTR and DTR-IDLE state. LM_Idle services may only be invoked in the DTR, DTR-IDLE and DTR-LOCKED states. When the station establishes exclusive mode operation on behalf of another LSAP-connection the FSM transitions to the DTR-LOCKED state. From this state the LSAP-connection may be disconnected with a resulting transition to LOCKED out (which serves to prevent new connections being established when the station is in exclusive mode).

3.5.4.3 Precise Description

3.5.4.3.1 LSAP Connection Control State Transition Diagram



3.5.4.3.2 LSAP Connection Control State Transition Table

State	Event	Action	Next State	
DISCONNECTED	LM_Connect.request (userData)	connectData=userData LS_Connect.request /*Open and Bind IrLAP Connection*/ StartWatchDogTimer	SETUP-PEND	1
	LM_Connect.response	Error	DISCONNECTED	
	LM_Disconnect.request	Error	DISCONNECTED	
	LM_Idle.request	Error	DISCONNECTED	
	LM_Data.request	Error	DISCONNECTED	
	LM_UData.request	Error	DISCONNECTED	
	LM_Status.request	Error	DISCONNECTED	
	LS_Connect.confirm	LS_Disconnect.request	DISCONNECTED	
	LS_LockOut.request (lockout=true)	LS_LockOut.confirm(lockout=true) /* Accept Lockout */	LOCKED-OUT	17
	LS_LockOut.request (lockout=false)	LS_LockOut.confirm(lockout=false) /* No Change */	DISCONNECTED	
	LS_Status.indication	Error	DISCONNECTED	
	LS_Status.confirm	Error	DISCONNECTED	

State	Event	Action	Next State	
	IrLAP_Data.indication (Data LM-PDU)	IrLAP_Data.request (Disconnect LM-PDU [reason=Disconnected]) /* Data delivered on a disconnected LSAP connection is rejected with an Disconnect LM-PDU */	DISCONNECTED	
	IrLAP_Data.indication (Connect LM-PDU[userData])	connectData=userData LS_Connect.request /* Bind to IrLAP Connection .indication delivered to LSAP User following LS_Connect.confirm */r	CONNECT-PEND	8
	IrLAP_Data.indication (Connect confirm LM-PDU)	IrLAP_Data.request (Disconnect LM-PDU [reason=Disconnected]) /* Connection confirmation delivered on a non existent LSAP-connection is rejected with a disconnect LM-PDU. */	DISCONNECTED	
	IrLAP_Data.indication (Disconnect LM-PDU)	Error	DISCONNECTED	
	WatchDogTimeOut	/* Ignore */	DISCONNECTED	
CONNECT-PEND	LM_Connect.request	LM_Disconnect.indication (reason=incomingConnection)	CONNECT-PEND	
	LM_Connect.response (userData)	Error /* No .indication issued yet */	CONNECT-PEND	
	LM_Disconnect.request	Error /* Not yet Bound to IrLAP connection */	CONNECT-PEND	
	LM_Idle.request	Error	CONNECT-PEND	
	LM_Data.request	Error	CONNECT-PEND	
	LM_UData.request	Error	CONNECT-PEND	
	LM_Status.request	Error /* Not yet Bound to IrLAP connection */	CONNECT-PEND	
	LS_Connect.confirm /*Bound to IrLAP Connection*/	LM_Connect.indication (connectData)	CONNECT	19
	LS_Disconnect.indication (reason)	LM_Disconnect.indication(reason)	DISCONNECTED	9
	LS_Status.indication(status)	Error /* Not yet Bound to IrLAP connection */	CONNECT-PEND	
	LS_Status.confirm(status)	Error /* Not yet Bound to IrLAP connection */	CONNECT-PEND	
	LS_LockOut.request (lockout=*)	LS_LockOut.confirm(lockout=false) /* Reject Lockout or no change */	CONNECT-PEND	
	IrLAP_Data.indication	Error	CONNECT-PEND	
CONNECT	LM_Connect.request	LM_Disconnect.indication (reason=incomingConnection)	CONNECT	
	LM_Connect.response (userData)	IrLAP_Data.request (Connect Confirm LM-PDU[userData], expedited=false)	DTR	10
	LM_Disconnect.request	IrLAP_Data.request (Disconnect LM-PDU [reason=userRequest], expedited=false) LS_Disconnect.request /* Unbind IrLAP connection */	DISCONNECTED	9
	LM_Idle.request	Error	CONNECT	
	LM_Data.request	Error	CONNECT	
	LM_UData.request	Error	CONNECT	
	LM_Status.request	LS_Status.request	CONNECT	
	LS_Connect.confirm	Error	CONNECT	
	LS_Disconnect.indication (reason)	LM_Disconnect.indication(reason)	DISCONNECTED	20
	LS_Status.indication(status)	LM_Status.indication(status)	CONNECT	

State	Event	Action	Next State	
	LS_Status.confirm(status)	LM_Status.confirm(status)	CONNECT	
	LS_LockOut.request (lockout=*)	LS_LockOut.confirm(lockout=false) /* Reject Lockout or no change */	CONNECT	
	IrLAP_Data.indication	Error	CONNECT	
DATA-TRANSFER-READY (DTR)	LM_Connect.request	Error	DTR	
	LM_Connect.response	Error	DTR	
	LM_Disconnect.request	IrLAP_Data.request (Disconnect LM-PDU [Reason=UserRequest], expedited=false) LS_Disconnect.request /* Unbind IrLAP connection */	DISCONNECTED	7
	LM_Idle.request(mode=idle) ∧ stationMode!=exclusive	LM_Idle.confirm (status=success,mode=idle) LS_Idle.request(mode=idle)	DTR-IDLE	11
	LM_Idle.request(mode=idle) ∧ stationMode=exclusive	LM_Idle.confirm (status=failed,mode=active)	DTR	
	LM_Idle.request(mode=active)	LM_Idle.confirm (status=success,mode=active)	DTR	
	LM_Data.request(userData)	IrLAP_Data.request (Data-LM-PDU[userData], expedited=false)	DTR	
	LM_UData.request(userData)	IrLAP_Data.request (Data-LM-PDU[userData], expedited=true)	DTR	
	LM_Status.request	LS_Status.request	DTR	
	LS_Connect.confirm	Error	DTR	
	LS_Disconnect.indication (reason)	LM_Disconnect.indication(reason)	DISCONNECTED	7
	LS_LockOut.request (lockout=*)	LS_LockOut.confirm(lockout=false) /* Reject Lockout or no change */	DTR	
	LS_Status.indication(status)	LM_Status.indication(status)	DTR	
	LS_Status.confirm(status)	LM_Status.confirm(status)	DTR	
	IrLAP_Data.indication (Data LM-PDU[userData], expedited=false)	LM_Data.indication(userData)	DTR	
	IrLAP_Data.indication (Data LM-PDU[userData], expedited=true)	LM_UData.indication(userData)	DTR	
	IrLAP_Data.indication (Connect LM-PDU)	IrLAP_Data.request (Disconnect LM-PDU[reason=halfOpen], expedited=false) LS_Disconnect.request LM_Disconnect.indication(reason=halfOpen)	DISCONNECTED	7
	IrLAP_Data.indication (Connect confirm LM-PDU)	Error	DTR	
	IrLAP_Data.indication (Disconnect LM-PDU [reason])	LS_Disconnect.request LM_Disconnect.indication(reason)	DISCONNECTED	7
	IrLAP_Data.indication (Disconnect LM-PDU)	Error	DTR	
	WatchDogTimeOut	/* Ignore */	DTR	
SETUP-PEND	LM_Connect.request	Error	SETUP-PEND	
	LM_Connect.response	Error	SETUP-PEND	
	LM_Disconnect.request	Error	SETUP-PEND	
	LM_Idle.request	Error	SETUP-PEND	
	LM_Data.request	Error	SETUP-PEND	
	LM_UData.request	Error	SETUP-PEND	
	LM_Status.request	Error	SETUP-PEND	

State	Event	Action	Next State	
	LS_Connect.confirm	IrLAP_Data.request (Connect LM-PDU[connectData], expedited=false) StartWatchDogTimer	SETUP	3
	LS_Disconnect.indication (reason)	LM_Disconnect.indication(reason) CancelWatchDogTimer	DISCONNECTED	2
	LS_LockOut.request (lockout=*)	LS_LockOut.confirm(lockout=false) /* Reject lockout or no change */	SETUP-PEND	
	LS_Status.indication(status)	LM_Status.indication(status)	SETUP-PEND	
	LS_Status.confirm(status)	LM_Status.confirm(status)	SETUP-PEND	
	IrLAP_Data.indication	Error /* Can't receive data - not 'bound' to an IrLAP connection */	SETUP-PEND	
	IrLAP_UnitData.indication	Error	SETUP-PEND	
	WatchDogTimeOut	LS_Disconnect.request	DISCONNECTED	2
SETUP	LM_Connect.request	Error	SETUP	
	LM_Connect.response	Error	SETUP	
	LM_Disconnect.request	Error	SETUP	
	LM_Idle.request	Error	SETUP	
	LM_Data.request	Error	SETUP	
	LM_UData.request	Error	SETUP	
	LM_Status.request	LS_Status.request	SETUP	
	LS_Connect.confirm	Error	SETUP	
	LS_Disconnect.indication (reason)	LM_Disconnect.indication(reason) CancelWatchDogTimer	DISCONNECTED	4
	LS_LockOut.request (lockout=*)	LS_LockOut.confirm(lockout=false) /* Reject lockout or no change */	SETUP	
	LS_Status.indication(status)	LM_Status.indication(status)	SETUP	
	LS_Status.confirm(status)	LM_Status.confirm(status)	SETUP	
	IrLAP_Data.indication (Data LM-PDU)	Error	SETUP	
	IrLAP_Data.indication (Connect LM-PDU)	/* No need to send Disconnect - peer will see matching Connect */ LS_Disconnect.request LM_Disconnect.indication (connectionRace) CancelWatchDogTimer	DISCONNECTED	4
	IrLAP_Data.indication (Connect confirm LM-PDU [userData])	LM_Connect.confirm(userData) CancelWatchDogTimer	DTR	5
	IrLAP_Data.indication (Disconnect LM-PDU [reason])	LS_Disconnect.request LM_Disconnect.indication(reason) CancelWatchDogTimer	DISCONNECTED	4
	WatchDogTimeOut	LS_Disconnect.request LM_Disconnect.indication (nonResponsivePeer)	DISCONNECTED	4
LOCKED-OUT	LM_Connect.request	LM_Disconnect.indication (reason=lockedOut)	LOCKED-OUT	
	LM_Connect.response	Error	LOCKED-OUT	
	LM_Disconnect.request	Error	LOCKED-OUT	
	LM_Idle.request	Error	LOCKED-OUT	
	LM_Data.request	Error	LOCKED-OUT	
	LM_UData.request	Error	LOCKED-OUT	
	LM_Status.request	Error	LOCKED-OUT	
	LS_Connect.confirm	Error	LOCKED-OUT	
	LS_Disconnect.indication	Error	LOCKED-OUT	
	LS_LockOut.request (lockout=true)	LS_LockOut.confirm(lockout=true)	LOCKED-OUT	
	LS_LockOut.request (lockout=false)	LS_LockOut.confirm(lockout=false)	DISCONNECTED	18
	LS_Status.indication	Error	LOCKED-OUT	
	LS_Status.confirm	Error	LOCKED-OUT	

State	Event	Action	Next State	
	IrLAP_Data.indication	Error	LOCKED-OUT	
	WatchDogTimeOut	/* Ignore */	LOCKED-OUT	
DTR-LOCKED	LM_Connect.request	Error	DTR-LOCKED	
	LM_Connect.response	Error	DTR-LOCKED	
	LM_Disconnect.request	IrLAP_Data.request (Disconnect LM-PDU [Reason=UserRequest], expedited=false) LS_Disconnect.request /* Unbind IrLAP connection */	LOCKED-OUT	16
	LM_Idle.request(mode=active)	LM_Idle.confirm (status=failed, mode=idle)	DTR-LOCKED	
	LM_Idle.request(mode=idle)	LM_Idle.confirm (status=success, mode=idle)	DTR-LOCKED	
	LM_Data.request	Error	DTR-LOCKED	
	LM_UData.request	Error	DTR-LOCKED	
	LM_Status.request	LS_Status.request	DTR-LOCKED	
	LS_Connect.confirm	Error	DTR-LOCKED	
	LS_Disconnect.indication (reason)	LM_Disconnect.indication(reason)	LOCKED-OUT	16
	LS_LockOut.request (lockout=true)	LS_LockOut.confirm(lockout=true)	DTR-LOCKED	
	LS_LockOut.request (lockout=false)	LS_LockOut.confirm(lockout=false) LM_Status.indication(lockout=false)	DTR-IDLE	15
	LS_Status.indication(status)	LM_Status.indication(status)	DTR-LOCKED	
	LS_Status.confirm(status)	LM_Status.confirm(status)	DTR-LOCKED	
	IrLAP_Data.indication (Data LM-PDU [userData], expedited=false)	LM_Data.indication(userData)	DTR-LOCKED	
	IrLAP_Data.indication (Data LM-PDU [userData], expedited=true)	LM_UData.indication(userData)	DTR-LOCKED	
	IrLAP_Data.indication (Connect LM-PDU)	IrLAP_Data.request (Disconnect LM-PDU[reason=halfOpen], expedited=false) LS_Disconnect.request LM_Disconnect.indication(reason=halfOpen)	LOCKED_OUT	16
	IrLAP_Data.indication (Connect confirm LM-PDU)	Error	DTR-LOCKED	
	IrLAP_Data.indication (Disconnect LM-PDU [reason])	LS_Disconnect.request LM_Disconnect.indication(reason)	LOCKED-OUT	16
	WatchDogTimeOut	/* Ignore */	DTR-LOCKED	
DTR-IDLE	LM_Connect.request	Error	DTR-IDLE	
	LM_Connect.response	Error	DTR-IDLE	
	LM_Disconnect.request	IrLAP_Data.request (Disconnect LM-PDU [Reason=UserRequest], expedited=false) LS_Disconnect.request /* Unbind IrLAP connection */	DISCONNECTED	13
	LM_Idle.request(mode=idle)	LM_Idle.confirm (status=success, mode=idle)	DTR-IDLE	
	LM_Idle.request(mode=active)	LM_Idle.confirm (status=success, mode=active) LS_Idle.request(mode=active)	DTR	12
	LM_Data.request	Error	DTR-IDLE	
	LM_UData.request	Error	DTR-IDLE	
	LM_Status.request	LS_Status.request	DTR-IDLE	
	LS_Connect.confirm	Error	DTR-IDLE	
	LS_Disconnect.indication (reason)	LM_Disconnect.indication(reason)	DISCONNECTED	13

State	Event	Action	Next State	
	LS_LockOut.request (lockout=true)	LS_LockOut.confirm(lockout=true) LM_Status.indication(lockout=true)	DTR-LOCKED	14
	LS_LockOut.request (lockout=false)	LS_LockOut.confirm(lockout=false) /* No Change */	DTR-IDLE	
	LS_Status.indication(status)	LM_Status.indication(status)	DTR-IDLE	
	LS_Status.confirm(status)	LM_Status.confirm(status)	DTR-IDLE	
	IrLAP_Data.indication (Data LM-PDU [userData], expedited=false)	LM_Data.indication(userData)	DTR-IDLE	
	IrLAP_Data.indication (Data LM-PDU [userData], expedited=true)	LM_UData.indication(userData)	DTR-IDLE	
	IrLAP_Data.indication (Connect LM-PDU)	IrLAP_Data.request (Disconnect LM-PDU[reason=halfOpen], expedited=false) LS_Disconnect.request LM_Disconnect.indication(reason=halfOpen)	DISCONNECTED	13
	IrLAP_Data.indication (Connect confirm LM-PDU)	Error	DTR-IDLE	
	IrLAP_Data.indication (Disconnect LM-PDU [reason])	LS_Disconnect.request LM_Disconnect.indication(reason)	DISCONNECTED	13
	WatchDogTimeOut	/* Ignore */	DTR-IDLE	

3.5.4.3.3 State Definitions

DISCONNECTED. LSAP Connection does not exist.

SETUP-PEND.
An LM_Connect.request has been received from the service user. A request has been sent to the Station Control FSM to set up the underlying IrLAP connection.

SETUP.
Station control has set up the underlying IrLAP connection. A request to open an LSAP connection has been transmitted to the peer LSAP-connection control FSM.

CONNECT-PEND.
An Connect LM-PDU has been received and an LS_Connect.request has been issued to Station Control to bind the LSAP-connectiion to the underlying IrLAP connection. A response is awaited from Station Control. In practice the response will be near instantaneous since the IrLAP connectiion must have been open in order for the Connect LM-PDU to have been received.

CONNECT.
A LM_Connect.indication has been posted to the registered service user. Awaiting response from service user.

DATA TRANSFER READY (DTR).
An LSAP-connection has been established. Ready for data transfer.

DTR IDLE.
The service user has indicated that the link is idle.

DTR LOCKED.
Another service user has brought the station into its exclusive control. This LSAP connection is currently locked out.

LOCKED OUT.
Another service user has the station in exclusive use. No other connections can become active.

3.5.4.3.4 Event Descriptions

LM_Connect.request(userData).
Invocation of LM_Connect.request by LM-MUX service user

LM_Connect.response(userData).
Invocation of LM_Connect.response by LM-MUX service user

LM_Data.request(userData).
Invocation of LM_Data.request by LM-MUX service user

LM_Disconnect.request.
Invocation of LM_Disconnect.request by LM-MUX service user

LM_Idle.request.
Invocation of LM_Idle.request by LM-MUX service user

LM_Status.request.
Invocation of LM_Status.request by LM-MUX service user

LM_UData.request(userData).
Invocation of LM_UData.request by LM-MUX service user

LS_Connect.confirm.
Event from Station Control completing association of LSAP and IrLAP connections.

LS_Disconnect.indication(reason).
Event from Station Control indicating absence of IrLAP connection to associate with an LSAP-connection or the ending of such an association due to the failure of the associated IrLAP connection.

LS_LockOut.request(lockout).
Request from Station Control as preluded epilogue to LM-MUX being placed in exclusive mode (lockout=true) or subsequent to a return to shared mode (lockout=false).

LS_Status.confirm(status).
A pass through of an IrLAP_Status.confirm received by station control on behalf of this LSAP-connection endpoint.

LS_Status.indication(status).
A pass through of unsolicited IrLAP_Status.indication primitives received by station control and forwarded to all LSAP-connection endpoints associated with the IrLAP connection.

IrLAP_Data.indication(LM-PDU,expedited=false).
Delivery of reliable Data, Connect, Connect Confirm and Disconnect LM-PDUs. LM-PDU parameters may be shown enclosed in [].

IrLAP_Data.indication(Data LM-PDU,expedited=true).
Delivery of unreliable Data LM-PDUs. Other LM-PDU types may not be delivered unreliably.

stationMode.
A flag used to prevent an exclusive LSAP-connection being marked idle. May take the values exclusive or shared.

WatchDogExpired
The LSAP-connection watchdog timer has expired. This timer runs whenever the LSAP-Connection Control FSM is waiting for a response from either the LM-MUX user, local station control or its peer LSAP-Connection Control FSM. An implementation may effectively disable the watchdog by regarding it as having infinite duration. It is intended to catch non-responsive client, station control or peer entities resulting from implementation errors or the mutation of LM-PDUs not detected any of the guards applied within IrLAP and below. Where implemented there is once such timer for each instance of the LSAP-connection control FSM.

3.5.4.3.5 Action Descriptions

Error.
Indicates an illegal or unexpected event - In the case of LM-MUX service primitives the invocation mechanism of a practical implementation may report the error to the LM-MUX service user. In the case of IrLMP internal events and IrLAP service primitives, these may be silently ignored. In all cases the error results in no change in the state of the FSM.

LM_Connect.confirm(userData).
Delivery of LM_Connect.confirm to LM-MUX service user.

LM_Connect.indication(userData).
Delivery of LM_Connect.indication to LM-MUX service user.

LM_Data.indication(userData).
Delivery of LM_Data.indication to LM-MUX service user.

LM_Disconnect.indication(reason).
Delivery of LM_Disconnect.indication to LM-MUX service user.

LM_Idle.confirm(status, mode).
Delivery of LM_Idle.confirm to LM-MUX service user.

LM_Status.confirm(status).
Delivery of LM_Status.confirm to LM-MUX service user.

LM_Status.indication(status).
Delivery of LM_Statuse.confirm to LM-MUX service user. This reports both unsolicited IrLAP_Status.indications and transitions between DTR-IDLE and DTR-LOCKED.

LM_UData.indication(userData).
Delivery of LM_UData.indication to LM-MUX service user.

LS_Connect.request.
Request station control to associate the LSAP-connection endpoint with an IrLAP connection.

LS_Disconnect.request.
Requests station control to end the association between an LSAP-connection and an IrLAP-connection. Station control may subsequently close the IrLAP-connection.

LS_LockOut.confirm(lockout).
Response to LS_LockOut.request from station control. The request succeeds if the returned 'lockout' value matches that of the corresponding request[3].

[3] **NB.** This action is now redundant as it is ignored by Station Control. Station Control examines the state of the LCC FSM to ensure that the corresponding LS_LockOut.request will succeed.

LS_Status.request.
Requests station control to invoke IrLAP_Status.request on the underlying IrLAP-connection. Station control aggregates requests distributes confirms.

LS_Idle.request(mode)
Informs the station control that the service user has changed the mode of this LSAP connection to either active or idle.

IrLAP_Data.request(LM-PDU,expedited).
Requests transmission of Data, Connect, Connect Confirm and Disconnect LM-PDUs. LM-PDU parameters may be shown enclosed in []. For Data LM-PDUs expedited may be set true. For all other LM-PDUs expedited is set false.

StartWatchDogTimer
Start or restart the watchdog timer. It is recommended that this timer have a duration of at least 20 seconds however, the exact timer length is implementation dependent.

CancelWatchDogTimer
Stop the LSAP-connection watchdog timer.

4. Information Access Service

Each IrDA device provides an Information Access Service (IAS). The IAS maintains information about the services provided by this IrDA device and also provides operations for remotely accessing the information base on another device. This information is needed so that clients on a remote device can find configuration information needed in order to access a service. One such piece of information that a client application or a transport protocol must supply is the destination LSAP selector for sending frames through the multiplexer.

This information is held in a number of objects in the information base, providing a simple, uniform way for services to advertise their presence and any information needed to access them. The information model defines the external conceptual view of the information held by an IAS. It defines the operations used to access the information and the format of transmitted data. This specification does not dictate the internal organization of an implementation that is used to meet this specification nor does this specification describe how information is registered with the local IAS. The Information Access Service does not control or mandate the information held except for the “Device” object as described in section 5.

Figure 6 outlines the organization of components for each IrLAP connection:

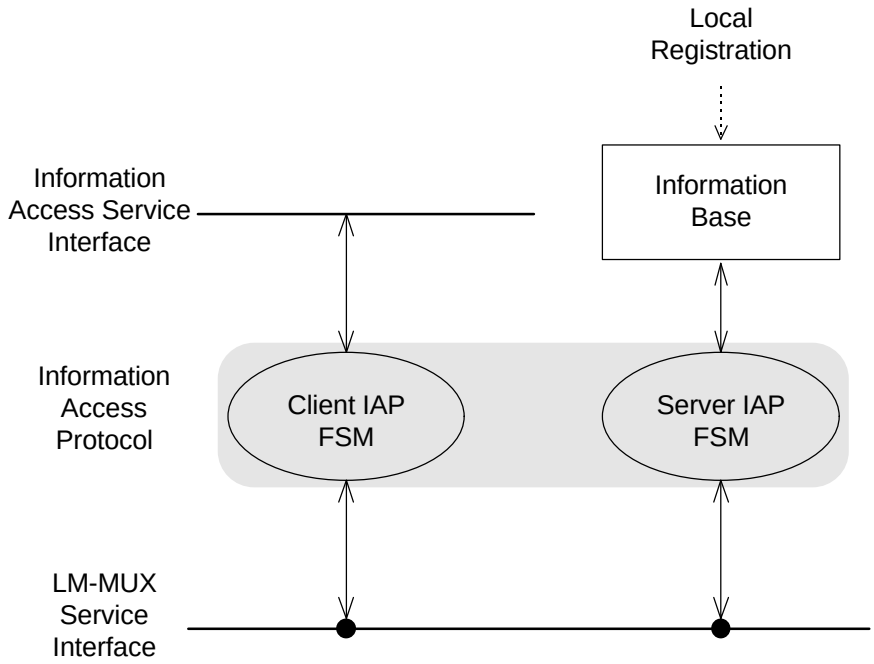


Figure 6. Internal Organization of the Information Access Service

The Information Base holds the data about the local services being offered to other IrDA devices. There is a simple protocol for one IrDA device to access the information base of another.

The Information Access Protocol (IAP) is the means by which two IAS entities communicate. It is a command/response protocol where each operation has results, including an indication of the success or failure. The server is always found at the same location within any IrDA device, at LSAP selector zero. The protocol uses the reliable data transfer of the IrLAP. There is one instance of server IAP finite state machine for each device that can be contacted and one instance per contactable device of the client IAP finite state machine if this device provides the query primitives.

4.1 Information Model

Each service, including all other clients of the multiplexer, may provide information for an object in the information base. Each object in the Information Base has a class name for the object, an identifier which uniquely specifies the object within the device and a number of attributes. There may be several objects of the same class. Each attribute consists of a name that identifies the storage slot within its object and a typed value. Values are typed according to a fixed set of base types. Compound types are not supported. The example figure shows a possible configuration with three objects, the distinguished “device” object and two services, named “Foo” and “Bar”.

Object 0: Class “Device”

“DeviceName”	“MyDevice”
“IrDASupport”	Binary data

Object 1: Class “Foo”

“Attribute 1”	5
“Attribute 2”	“String”
“Attribute 3”	“Binary”

Object 7: Class “Bar”

“Attribute 1”	7
“Attribute 4”	“String”
“Attribute 3”	“Binary”

Figure 7. Example Information Base

A service designer will define the attributes that objects of the service class can or must provide. It is the responsibility of the information provider to ensure that any required attributes are provided. The IAS does not enforce any class specific constraints.

There is one distinguished object, with identifier 0 (zero), of class “Device”, that is always present. It contains details about this device such as its name.

4.2 Service Primitives

The Information Access Protocol (IAP) provides operations to identify all the objects currently in the information base, to access attributes within an object and to find all the attribute names of a given object.

In addition, a fast access mechanism is provided that returns the value for an attribute in a particular class; if there is more than one object of this service class, then a list of attribute values is returned. This operation is particularly useful when there is only ever one object of a given class in a device.

The IAS only provides for reading the value of an attribute and does not permit setting the value of an attribute on a remote device. This specification does not define how objects are locally registered or manipulated.

Only the `LM_GetValueByClass` service is required in all implementations. See section 6 on minimal implementations.

4.2.1 LM_GetInfoBaseDetails

```
LM_GetInfoBaseDetails.request(address)
LM_GetInfoBaseDetails.confirm(number of objects, max. object id)
```

address	32 bit address of remote device
number of objects	Number of objects in the remote information base
max. object id	Highest object identifier in use at the time of the call

This service returns information about the information base so that a client can choose appropriate parameters for the `LM_GetObjects` call.

This service is optional.

4.2.2 LM_GetObjects

```
LM_GetObjects.request(address, first id, max. list, class name)
LM_GetObjects.confirm(next id, list of (id, num. attributes, class name))
```

address	32 bit address of remote device
first id	Object identifier for the first object in the returned list
max. list	Upper bound on the length of the list to be returned
class name	Name of class; if a zero length string, all classes are searched
next id	Object identifier of the object after the last one in the list
id	Object identifier of this list item
num. attributes	Number of attributes for this object
class name	Object's class name

A client can find out the class name, the identifier and the number of attributes for a range of object identifiers. This is done by specifying an object identifier and the length of the information list required. The next object identifier to use is also returned. A restriction to a single class name may be specified to find all objects of a single class name. If a null class name is provided, then information about all registered classes is returned.

By repeatedly calling this primitive, specifying the returned 'next' identifier as the first identifier argument in the subsequent call, a client will get details of every object in the information base assuming that no local changes are made to the remote information base. The list returned may be shorter then the requested maximum length. If a class name was provided to restrict the search in the `.request` primitive, then the class name is of zero length in the associated `.confirm`.

This service is optional.

4.2.3 LM_GetValue

```
LM_GetValue.request(address, id, attribute name1, [attribute name2, ...])
LM_GetValue.confirm(List of attribute values)
```

address	32 bit address of remote device
id	Object identifier
attribute name	Attribute whose value is required
value	Attribute value

Access one or more attributes of a specified object. A value of type "missing" is returned if the named attribute does not exist.

This service is optional.

4.2.4 LM_GetValueByClass

```
LM_GetValueByClass.request(address, class name, attribute name)
LM_GetValueByClass.confirm(list of (object id, attribute value))
```

address	32 bit address of remote device
class name	The class name scopes the search for objects
attribute name	Attribute whose value is required
object id	Object identifier of this list item
value	Attribute value

Access the all the values of a named attribute in objects of a given class name. A list is always returned, even if there is only one returned value. The class name may not be the null string.

This service is required in all IAP servers.

4.2.5 LM_GetObjectInfo

```
LM_GetObjectInfo.request(address, id)
LM_GetObjectInfo.confirm(lowest slot, highest slot, num. slot)
```

address	32 bit address of remote device
id	Object identifier
lowest slot	First slot currently in use for this object
highest slot	Last slot currently in use for this object
num. slot	Total number of slots currently in use

The operation `LM_GetObjectInfo` returns information that is useful when calling the service `LM_GetAttributeNames`. It returns the highest and lowest attribute slot currently occupied as well as the total number of slots in use.

This service is optional.

4.2.6 LM_GetAttributeNames

```
LM_GetAttributeNames.request(address, id, first slot, number of names)
LM_GetAttributeNames.confirm(next slot, List of (attribute name, type))
```

address	32 bit address of remote device
id	Object identifier
first slot	Slot number of the first requested attribute name
number of names	Maximum number of attribute names to return
next slot	Slot number of slot after the last one in the returned list
attribute name	Name of an attribute for the object
type	Type of the attribute

`LM_GetAttributeNames` enables a client to find every attribute of an object. This is done in series of calls that return a list of attribute names. By calling `LM_GetAttributeNames` using the returned next slot number from the previous call it made, a client will get every attribute of an object provided no changes are made to the object. The list returned may be shorted than the number requested.

This service is optional.

4.3 Elements of Procedure

This section describes the transmission format for each item in the object model.

4.3.1 Class Names

Class names are transmitted as an untyped octet sequence:

Length	Class Name
1 octet	"Length" octets

where length is an unsigned 8 bit quantity. The maximum length of a class name is 60 octets.

4.3.2 Object Identifier

An object identifier is a 16 bit unsigned number transmitted most significant byte first

Identifier
2 octets

4.3.3 Attributes

An attribute is a name-value pair. The name is a length encoded sequence of octets. The value is a typed field, with a length field if the type is not of fixed length, and a sequence of octets comprising the actual value. An object can have a maximum of 256 attributes.

4.3.3.1 Attribute Names

Attribute names are transmitted as an untyped octet sequence:

Length	Attribute Name
1 octet	"Length" octets

where length is an unsigned 8 bit quantity. The maximum length of an attribute name is 60 octets. Attribute names are scoped by the object containing them. An object must not have two attributes of the same name. Octets are transmitted in sequence.

4.3.3.2 Attribute Values

Values types are shown in Table 6. Some value types allow variable length values. These contain a length field.

Type	Type Identifier	Length	Description
Missing	0	Fixed : zero	Indicates error in requested attribute
Integer	1	Fixed : 4 octets	32 bit signed integer in Internet order
Octet sequence	2	Variable 16 bit length field	A sequence of octets up to 1024 octets
User String	3	Variable 8 bit length field	A sequence of characters with character set information. Maximum length is 256 characters.

Table 6. IrLMP Attribute Value Types

4.3.3.2.1 Missing

Type = 0
1 octet

The missing type indicates that a requested attribute does not exist.

4.3.3.2.2 Integer

Type = 1	Integer
1 octet	4 octets

The integer type is transmitted with the most significant byte. Only 32 bit signed (2's complement) integers are supported.

4.3.3.2.3 Octet Sequence

Type = 2	Length	Sequence
1 octet	2 octets	"Length" octets

An octet string is a variable length sequence of 8 bit units with no implied meaning to the octets. An octet string has a length field of 16 bits interpreted as an unsigned integer. The maximum length of an octet sequence is 1024 octets. The length field is transmitted most significant byte first. Octets are transmitted in sequence order, that is, the first octet in the sequence is transmitted first. An Octet Sequence may have zero length (length value of zero).

4.3.3.2.4 User String

Type = 3	Char. Set	Length	Characters
1 octet	1 octet	1 octet	"Length" octets

A User String is a variable length value that is intended for presentation in human readable form. Attributes values of type User String have international character support so that it can displayed to the user. See [ISO8859]. The length is measured in octet units (not characters) so that unknown character sets can be skipped. The maximum length of a User String is 255 octets as given by the length field. A User String may have zero length (length value of zero).

Char Set Code	Meaning
0	ASCII
1	ISO-8859-1
2	ISO-8859-2
3	ISO-8859-3
4	ISO-8859-4
5	ISO-8859-5
6	ISO-8859-6
7	ISO-8859-7
8	ISO-8859-8
9	ISO-8859-9
0xFF = 255	UNICODE

Table 7. IrLMP Character Code Values

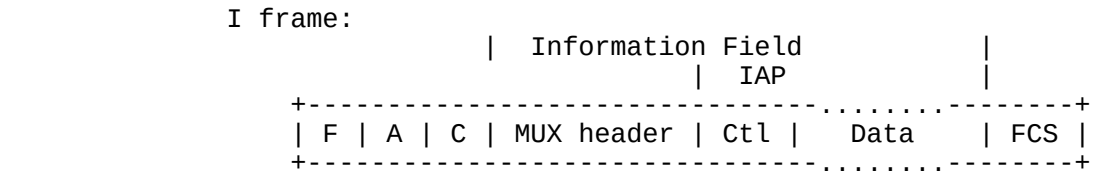
It should be noted that a device may not support the specified character set. In this case, it is up to the displaying device to display the string as best it can. UNICODE uses 16 bit characters.

4.3.4 Lists

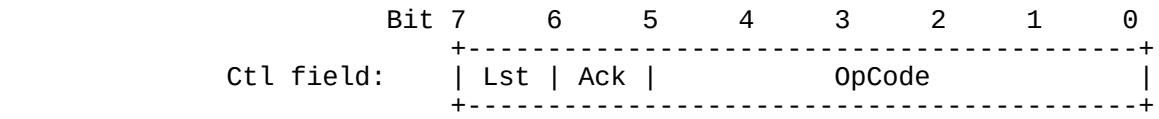
Several of the operations on the information base return a list of results. All lists consist of elements of the same kind. Such a list is transferred by specifying the number of items to be transferred by the operations. Each item is transmitted in the form defined for the element with no additional type information (see section 4.3.6). A list item may be a tuple of base items (Class names; Object Identifiers; Attribute names; Attribute values)..

4.3.5 IAP Frame Formats

The IAP uses the information field of IrLAP I frames. The following description of the frame format is in byte order.



The control (Ctl) byte is laid out as:



The Lst bit indicates whether this is the last frame in a multiframe command or result. It is 1 for the last (or only) frame in a command or in results. The Data fields of multiframe commands or results are concatenated to form operation frames described in section 4.3.6. Operation frames may be segmented in this way at any octet boundary.

It is required that IAS queries, i.e. command type operation frames, be sent in the minimum number of Data LM-PDUs possible. As an example it is not permitted to allow an IAS query of length 50 bytes to be send in two Data LM-PDUs. In this case the query must not be segmented and therefore will be transmitted in a single Data LM-PDU. This means that an IAS entity does not have to buffer IAS queries if it is known that no valid query for this IAS entity will be greater than that contained in a single frame. Such an IAS entity must be able to handle multiple frame queries but they can assume that valid queries will fit in a minimum number of frames.

The Ack bit indicates whether this is acknowledging a frame of type specified by the other fields. If the Ack bit is set, then there is no data (arguments or results) in the frame. The last frame in a multiframe command need not be explicitly acknowledged with a frame with the Ack bit set but instead implicitly by the return of the results. The last result frame need not be explicitly acknowledged with a frame with the Ack bit set but instead implicitly by transmission of the first frame of a new command.

4.3.6 Operation Frame Formats

All information base access operations use the Information Access Protocol. Each service primitive corresponds to a single call or reply by IAP. The formats of the frames are described by giving the op code, the arguments, and results. Arguments are packed into IAP frames with no padding. Each result starts with a return code and then zero or more results. Results are packed into frames directly after the status code. The status code is in the first octet of the first frame of the results.

All multi-octet integer values are transmitted most significant byte first.

4.3.6.1 LM_GetInfoBaseDetails

Op Code:	1
Arguments:	None
Return code:	0 (Success: results as shown)
	0xFF (Unsupported optional operation: no other results)
Results:	16 bit unsigned integer (number of objects)
	16 bit unsigned integer (largest object identifier)

4.3.6.2 LM_GetObjects

Op Code:	2
Arguments:	16 bit unsigned integer (first id)
	16 bit unsigned integer (max. length of result list)
	8 bit unsigned integer (length of class name: may be zero)
	“Length” octets (class name)
Return Code:	0 (Success: results as shown)
	0xFF (Unsupported optional operation: no other results)
Results:	16 bit unsigned integer (next identifier after list end)
	16 bit unsigned integer (list length)
	List of
	16 bit unsigned integer (object identifier)
	8 bit unsigned integer (number of attributes)
	8 bit unsigned integer (length of octet sequence: may be zero)
	“Length” Octets (class name)

4.3.6.3 LM_GetValue

Op Code:	3
Arguments:	16 bit unsigned integer (object identifier)
	16 bit unsigned integer (list length)
	List of
	8 bit unsigned integer (length of octet sequence)
	“Length” Octets (attribute name)
Return Code:	0 (Success: results as shown)
	1 (No such object: no other results)
	2 (One or more attribute names do not exist: results as shown. Attribute name in error will have a value type of “missing”)
	3 (Attribute name list too long: List length result field give maximum accepted length).
	0xFF (Unsupported optional operation: no other results)
Results:	16 bit unsigned integer (list length)
	List of (only for return code 0)
	Values encoded as described in section 4.3.3.2

4.3.6.4 LM_GetValueByClass

Op Code:	4	
Arguments:	8 bit unsigned integer (class name length)	
	“Length” Octets (class name)	
	8 bit unsigned integer (attribute name length)	
	“Length” Octets (attribute name)	
Return Code:	0	(Success: results as shown)
	1	(No such class: no other results)
	2	(No such attribute: no other results)
Results:	16 bit unsigned integer (list length)	
	List of	
	16 bit unsigned integer (object identifier)	
	Value encoded as described in section 4.3.3.2	

4.3.6.5 LM_GetObjectInfo

Op Code:	5	
Arguments:	16 bit unsigned integer (object identifier)	
Return Code:	0	(Success: results as shown)
	1	(No such object: no other results)
	0xFF	(Unsupported optional operation: no other results)
Results:	16 bit unsigned integer (lowest slot)	
	16 bit unsigned integer (highest slot)	
	16 bit unsigned integer (number of slots in use)	

4.3.6.6 LM_GetAttributeNames

Op Code:	6	
Arguments:	16 bit unsigned integer (object identifier)	
	16 bit unsigned integer (first slot)	
	8 bit unsigned integer (number of names to fetch)	
Return Code:	0	(Success: results as shown)
	0xFF	(Unsupported optional operation: no other results)
Results:	16 bit unsigned integer (next slot in use after last list item)	
	16 bit unsigned integer (list length)	
	List of	
	8 bit unsigned integer (length of attribute name)	
	“Length of name” Octets (attribute name)	
	8 bit unsigned integer (value type)	

4.4 Description of Procedure: IAP

4.4.1 Description

This descriptive text provides an overview of the Information Access Protocol. The finite state machines should be taken as definitive. An implementation is not required to implement the finite state machines, only to conform to their external behaviour.

Like any other client protocol running over LSAP-connections, the IAP is responsible for its own flow control. The acknowledgement frames are used as a low level flow control mechanism. As such, the IAP ensures that only one unacknowledged frame is outstanding at any one time for each connection.

There are two finite state machine per IrLAP link; one for initiating calls, one for processing calls. These two machines are independent of one another. Each has an LSAP-connection so there is one connection from the client machine on one device to the server machine on another device. There may also be a separate connection for a client on the other device to communicate with the server on the first device. There is at most one call in progress between a particular pair of client and server. There is one finite state machine for the server and one for the client (where present) for each address that this device can currently communicate with.

Client Engine

The client protocol engine consists of an FSM to connect to the particular address and an FSM to process an individual call. The state “S-Call” in the outer FSM is used to catch disconnection indications from any of the states in the inner machine.

Any call invoked while the previous call is still in progress is simply kept waiting, at the implementation writers discretion, until the engine returns to its starting state. Once the connection is established, the machine pauses in state “S-Wait-for-Call” in the inner FSM until a call request is received.

If there is a single frame (that is, the arguments for the call can be contained in a single IrLAP data frame) then the S-Call state machine moves from state “S-Make-Call” to state “S-Outstanding” where it awaits the reply. If there is more than one frame then the send state machine proceeds from state “S-Make-Call” to state “S-Calling” where the call engine waits for an acknowledgement of each frame before sending the next. When the frame with the LST bit is sent, the machine moves to state “S-Outstanding”.

In state “S-Outstanding” the call engine can process an optional acknowledgement of the last (or only) frame in a call as well as process the first frame in the results. An implementation of the server may choose to send the optional acknowledgement for the last frame in the command if it expects the processing of the call to take a long time.

If the first frame of the reply is the only frame then the machine proceeds from state “S-Outstanding” to state “S-MakeCall” or “S-Wait-for-Call”. If there several frames in the reply, the machine moves to state “S-Replying” where each incoming frame is acknowledged. Note that the final or only frame of the results may be acknowledged or it may be implicitly acknowledged by the first frame of the next call.

Server Engine

The server FSM manages connections between devices. The state “R-Call” in the outer FSM is used to catch disconnection indications from any of the states in the inner machine.

Once a connection is established, the receive machine waits in state “R-Waiting” until the first frame in a call arrives. It makes the LSAP-connection active. then, if this is the only frame, the machine moves to state “R-Execute”, otherwise it moves to state “R-Receiving” where each frame (except, optionally, the last) is acknowledged, causing the next frame in the call to be sent. When the last frame is seen, the machine moves to state “R-Execute”.

When the call has been executed, the receive machine can either send a single frame of results and proceed directly to state “R-Waiting” or it can send the first frame of a multiframe reply and move to state “R-Returning”. In this state, as each frame of the results is acknowledged, the next frame is sent. When the last frame is sent the machine moves to state “R-Waiting”. There is an optional acknowledgement of the final or only frame of the reply.

4.4.2 Notes and Notation

The IAP uses any quality of service for a connection that the multiplexer providers. No client data is supplied or expected in the connection or disconnection service primitive. QoS parameters, client data parameters and connection end points are omitted in the state transition tables for clarity.

F(lst:L, ack:A, op:O, args) describes a frame where the control byte is set according to the values for `lst`, `ack` bits and the `OpCode`.

Recv: F(lst:L, ack:A, op:O, args) is a `LM_Data.indication` of a frame matching the given description.

Send: F(lst:L, ack:A, op:O, args) is a `LM_Data.request` of a frame of the given description

`FL[args]`: a frame list of arguments

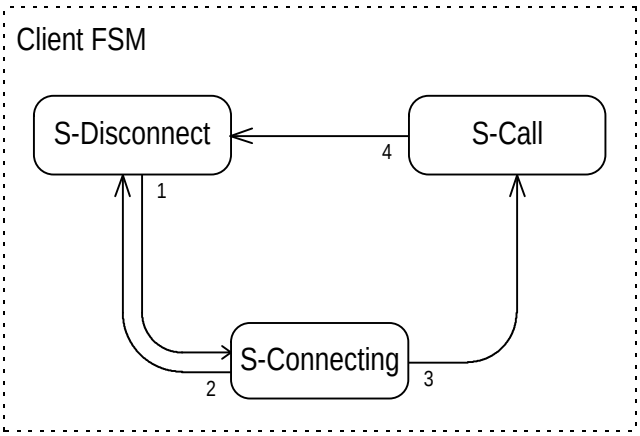
`FL[res]`: a frame list of results

[address, selector]: An LSAP-ID specifying a particular LSAP.

Events that are not recognized in a particular state are assumed to remain pending until any masking flag is modified or a transition to is made to a state where they can be recognized.

4.4.3 Client Finite State Machine

4.4.3.1 State Transition Diagram



4.4.3.2 State Transition Table

State	Event	Action	Next State	
S-Disconnect	call-request(addr, op, args)	LM_connect.request([addr, 0])	S-Connecting	1
S-Connecting	LM_disconnect.indication()	abort-calls(addr)	S-Disconnect	2
	LM_connect.confirm([addr,0])	call-request(addr, op, args)	S-Call	3
S-Call	LM_disconnect.indication()	abort-calls(addr)	S-Disconnect	4

4.4.3.3 State Definitions

S-Disconnect. The device has no LSAP connection to a particular remote device.

S-Connecting. The device is waiting for the LSAP connection to be established.

S-Call. The device can process calls to a specific remote device. Whenever the LSAP connection is disconnected, this state catches that event and clears up.

4.4.3.4 Event Descriptions

call-request(addr, op, args). Request to invoke the operation op on the remote device specified by the address given.

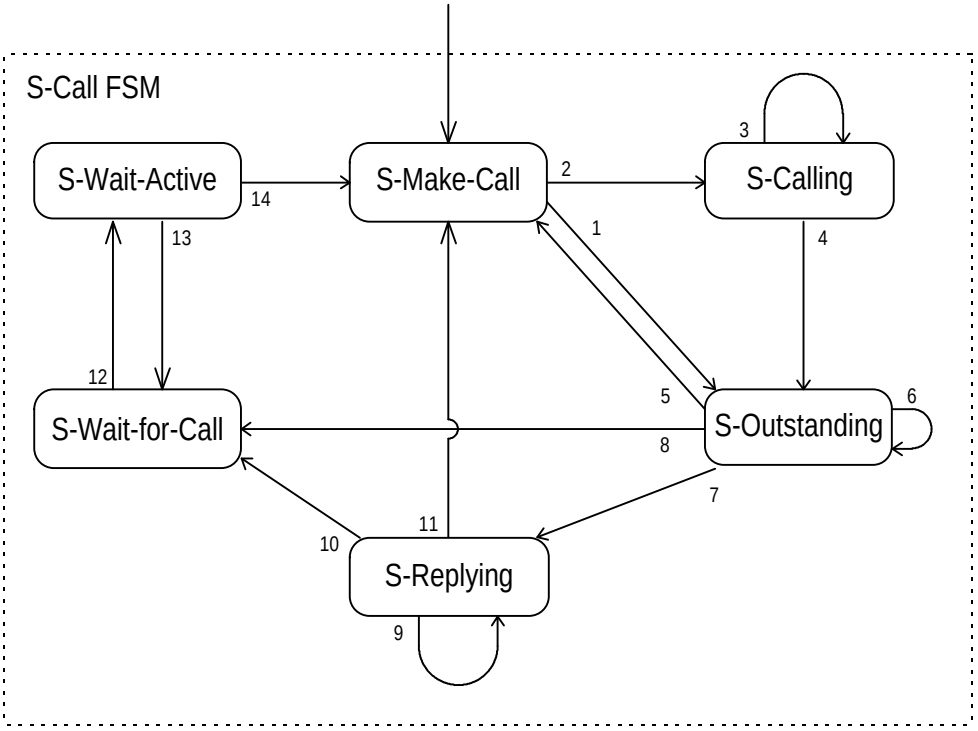
4.4.3.5 Action Descriptions

call-request(addr, op, args). Requeue the call request to be processed later.

abort-calls(address). Abort all executing and current calls to this address.

4.4.4 S-Call Finite State Machine

4.4.4.1 State Transition Diagram



4.4.4.2 State Transition Table

State	Event	Action	Next State	
S-Make-Call	call-request(addr, op, args) ∧ length(FL[args]) = 1	Send: F(lst:1, ack:0, op:op, all args)	S-Outstanding	1
	call-request(addr, op, args) ∧ length(FL[args])> 1	Send: F(lst:0, ack:0, op:op, first args)	S-Calling	2
S-Calling	Recv: F(lst:0, ack:1, op:op) ∧ ! last-arg-frame	Send: F(lst:0, ack:0, op:op, more args)	S-Calling	3
	Recv: F(lst:0, ack:1,op:op) ∧ last-arg-rame	Send: F(lst:1, ack:0, op:op, last args)	S-Outstanding	4
S-Outstanding	Recv: F(lst:1, ack:0, op:op) ∧ call-pending	call-confirm(results) [See Note 1]	S-Make-Call	5
	Recv: F(lst:1, ack:0, op:op) ∧ ! call-pending	Send: F(lst:1, ack:1,op:op) call-confirm(results) LM_Idle.request(idle)	S-Wait-for-Call	8
	Recv: F(lst:0, ack:0, op:op, res)	Send: F(lst:0, ack:1, op:op) store-results	S-Replying	7
	Recv: F(lst:1, ack:1, op:op)	[See note 1]	S-Outstanding	6
S-Replying	Recv: F(lst:0, ack:0, op:op, res)	Send: F(lst:0, ack:1, op:op) store-results	S-Replying	9
	Recv: F(lst:1, ack:0, op:op, res) ∧ ! call-pending	Send: F(lst:1, ack:1, op:op) [See note 2] call-confirm(results) LM_Idle.request(idle)	S-Wait-for-Call	10
	Recv: F(lst:1, ack:0, op:op, res) ∧ call-pending	call.confirm(results)	S-Make-Call	11
S-Wait-for-Call	LM_Idle.confirm(success, idle) ∧ call-request(addr, op, args) /* Transition to Idle ALWAYS succeeds */	LM_Idle.request(active) call-request(addr, op, args)	S-Wait-Active	12
S-Wait-Active	LM_Idle.confirm(refuse, idle)	abort-call	S-Wait-for-Call	13
	LM_Idle.confirm(success, active)	check-qos	S-Make-Call	14

Note 1: The acknowledgement of the last frame in the command is optional. A device may use the first frame of the response as an implicit acknowledgement.

Note 2: This is the acknowledgement of the last frame of a response. It is optional. The first frame of the next command is an implicit acknowledgement.

4.4.4.3 State Definitions

S-Make-call. The device is processing the beginning of a call

S-Calling. The device is sending a multiple frame command.

S-Outstanding. The device is waiting for the response to a command.

S-Replying. The device is collecting a multiple part response.

S-Wait-for-Call. The device is waiting for a call-request primitive.

S-Wait-Active. The device is waiting for its connection to become active.

4.4.4.4 Event Descriptions

call-request(address, op, FL[args]). Request for an operation to be performed.

last-arg-frame. True if there is only one frame of arguments in the current call

call-pending. True if the local implementation has queued an operation, pending the completion of the current one.

4.4.4.5 Action Descriptions

call-confirm(results). Notify complete call with results.

store-results. Store the partial results contained in the current frame so that all the partial results can be gathered together when the last frame of results is received.

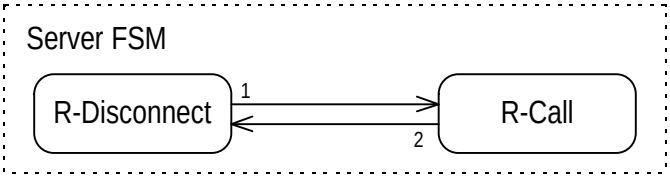
check-qos. Check the quality of service parameters and adjust any lists of frames accordingly.

call-request(addr, op, args). Requeue the call request to be processed later.

abort-call. Abort the current call to this address. The LSAP-connection can not be obtained. An implementation may instead choose to wait for the LSAP-connection to become available.

4.4.5 Server Finite State Machine

4.4.5.1 State Transition Diagram



4.4.5.2 State Transition Table

The end point for service primitives is always in LSAP [this address, 0].

State	Event	Action	Next State	
R-Disconnect	LM_connect.indication ([addr,ep])	LM_connect.response ([addr,ep]) LM_idle.request(idle)	R-Call	1
R-Call	LM_disconnect.indication()	abort-calls(addr)	R-Disconnect	2

4.4.5.3 State Descriptions

R-Disconnect. The device has no LSAP connection to a particular remote device.

R-Call. The device is ready to process incoming calls.

4.4.5.4 Event Descriptions

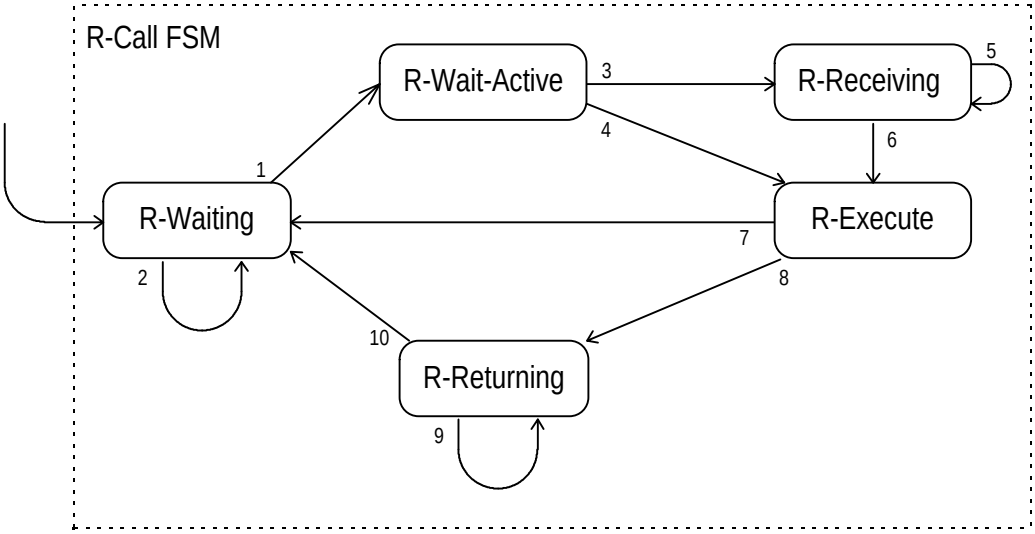
None

4.4.5.5 Action Descriptions

abort-calls(address). Abort all executing and pending calls from this address.

4.4.6 R-Connect Finite State Machine

4.4.6.1 State Transition diagram



4.4.6.2 State Transition Table

State	Event	Action	Next State	
R-Waiting	Recv: F(lst:?, ack:0, op:op, args) ∧ LM_Idle.confirm(success, idle) /* Transition to Idle ALWAYS succeeds */	LM_Idle.request(active)	R-Wait-Active	1
	Recv: F(lst:1, ack:1, op:op)	[See note 1]	R-Waiting	2
R-Wait-Active	LM_Idle.confirm(success, active) ∧ lst-is-0	Send: F(lst:0, ack:1, op:op) store-args	R-Receiving	3
	LM_Idle.confirm(success, active) ∧ lst-is-1	call-indication(op,args)	R-Execute	4
	LM_idle.confirm(refuse, idle)	[Note 3]		
R-Receiving	Recv: F(lst:0, ack:0, op:op, args)	Send: F(lst:0, ack:1, op:op) store-args	R-Receiving	5
	Recv: F(lst:1, ack:0, op:op, last args)	call-indication(op,arguments) Send: F(lst:1, ack:1, op:op) [See note 2]	R-Execute	6
R-Execute	call-response(FL[results]) ∧ length(FL[results]) = 1	Send: F(lst:1, ack:0, op:op, results) LM_Idle.request(idle)	R-Waiting	7
	call-response(FL[results]) ∧ length(FL[results]) > 1	Send: F(lst:0, ack:0, op:op, first res)	R-Returning	8
R-Returning	Recv: F(lst:0, ack:1, op:op) ∧ ! last-frame-of-results	Send: F(lst:0, ack:0, op:op, results)	R-Returning	9
	Recv: F(lst:0, ack:1, op:op) ∧ last-frame-of-results	Send: F(lst:1, ack:0, op:op, results) LM_Idle.request(idle)	R-Waiting	10

Note 1: This is an acknowledgement of the last frame of a response. It is optional. It may not occur, and instead be replaced by the first frame of the next command.

Note 2: Sending of an acknowledgement of the last frame of a command. It is optional. Implementations should issue this if the execution of the call will take an exceptionally long period of time. The subsequent first frame of the response will acknowledge the last frame of the command if this frame is not sent.

Note 3: This should not occur because the other end of the connection should be active in order to send the last frame. It would wait, with the LSAP-connection active, for the reply.

4.4.6.3 State Definitions

R-Waiting. The device is waiting for the first frame of a command. The LSAP-connection is idle while in this state even though the LM_idle.confirm(success, idle) event is only consumed when the state is left.

R-Wait-Active. The device is waiting for the connection to become active.

R-Receiving. The device is gathering the arguments of a multiple frame command.

R-Execute. The device is performing the operation.

R-Returning. The device is sending a multiple frame response.

4.4.6.4 Event Descriptions

call-response(FL[results]). Results of the current operation to be returned.

lst-is-0. The lst bit in the current call is 0.

lst-is-1. The lst bit in the current call is 1.

last-frame-of-results. True if there is just one more frame if results to send; false if there is more than one frame of results still to be sent.

4.4.6.5 Action Descriptions

call-indication(op, args). Collect arguments together and pass the call off to be executed. The execution of the call on the information base is an implementation issue.

store-args. Store the partial arguments contained in the current frame so that all the partial arguments can be gathered together when the last frame of arguments is received.

5. Appendix A: Required Object and Attributes

The following sections describe the object and attributes that must be supported for IrDA compliance.

5.1 Device Object

An instance of “Device” object class must be present in every IrDA device. The object contains general information about the device including, but not limited to the device’s name and level of IrLMP support. The sections below describe the required attributes of the “Device” object class. The names of required attributes are case-sensitive and must appear exactly as indicated (without the quotes).

5.1.1 Device Name Attribute

The “DeviceName” attribute is a user string (see section 4.3.3.2.4).

5.1.2 IrLMP Support Attribute

The “IrLMPSupport” attribute is an octet sequence (see section 4.3.3.2.3) and includes an indication of the major IrLMP version a device is running, and which optional IAS and LM-MUX features are supported. The format of the attribute is shown below.

IrLMP Version #	IAS Support	LM-MUX Support
1 Octet	n Octets	m Octet

The current IrLMP Version number is 1, encoded as 0x01.

Both the IAS and LM-MUX support fields are extensible. If the most significant bit of an octet in one of these fields is set then the field continues in the next octet. For this verision of IrLMP both these fields occupy a single octet. The meaning of assigned bits in the first octet of each field will NOT be changed in successive versions of this specification. Meaning may be given to currently unassigned bits and of course further octets may be added in future versions. Unassigned bits should be transmitted as zero and ignored on reception.

The IAS Support field is currently defined as follows:-

Byte 1	
Bit	Function
0	**GetInfoBaseDetails**
1	**GetObjects**
2	**GetValue**
3	Unassigned
4	**GetObjectInfo**
5	**GetAttributeNames**
6	Unassigned
7	**Extension**

Table 8. IAS Support Bit Assigments

If the corresponding bit is set then the IAS server within the device responds to an IAS request of the given type with a response of the corresponding type. If the bit is cleared the request is not supported by the IAS server and an unsupported resonse will be issued if such a request is received.

Byte 1	
Bit	Function
0	**Exclusive Mode**
1	**Role Exchange**
2	**ConnectionlessData**
3	Unassigned
4	Unassigned
5	Unassigned
6	Unassigned
7	**Extension**

Table 9. LM-MUX Support Bit Assigments

If the corresponding is set then the LM-MUX entity is prepared to honour transitions to exclusive mode, request for primary/secondary role exchange and deliver connectionless data as appropriate. If the corresponding bit is clear the LM-MUX entity will alway refuse transitions to exclusive mode, refuse primary/secondary role exchanges and does not support delivery of connectionless data, as appropriate.

Consequently both IAS and LM-MUX support field for a minimal IrLMP implmentation carry the single octet value 0x00.

5.2 Attributes for use in Service Object Class Definitions

This section defines two attributes that are intended for use by those defining object classes that represent services that are either directly attached as LM-MUX clients, or indirectly attached via a transport entity.

Use of these attributes is not mandatory, but their use is strongly encouraged in those circumstances where an attribute is required for the same purpose as these attributes are defined.

Attribute Name	Value Type	Description
IrDA:IrLMP:InstanceName or in hex: 0x49-72-44-41-3a-49-72-4c-4d-50-3a-49-6e-73-74-61-6e-63-65-4e-61-6d-65	User String	A user string that may be used to distinguish one instance of an object class from another. The string may be presented to the user in cases where there is a choice between two (or more) otherwise identical object instances (and hence the services instances that they represent). Where an IrDA device supports only a single instance of a given service, this attribute may be absent from an object instance; may be null (zero length); or may duplicate the device name held in the Device object instance.
IrDA:IrLMP:LsapSel or in hex: 0x49-72-44-41-3a-49-72-4c-4d-50-3a-4c-73-61-70-53-65-6c	Integer	An integer value in the range 0x00 through 0x6f that represent the LSAP-SEL value through which the service represented by an object instance may be accessed. This is primarily intended for use in objects that describe services that are directly attached to LM-MUX. The definition of any intervening transport entities is expected to include an attribute definition for identifying the endpoints provided by that entity eg. the TinyTP attribute IrDA:TinyTP:LsapSel.

6. Appendix B: Minimal Implementation

The following sections outline the minimum requirements for IrDA Link Management compliance. As indicated in section 5, the IrLMP version information reflects the support provided. Additionally, for the IAS commands and the exclusive mode command, the remote entity will respond with a status of NotSupported, when it cannot honor a request due to a minimal implementation.

6.1 Minimum Service Class Primitives.

The required service class primitives for the LM-MUX are :

- LM_DiscoverDevices
- LM_Connect
- LM_Disconnect
- LM_Status
- LM_Data
- LM_UData

The required service class primitives for the IAS are:

- LM_GetValueByClass

6.2 Optional Service Class Primitives.

The optional service class primitives for the LM-MUX are :

- LM_Sniff
- LM_Idle
- LM_AccessMode
- LM_ConnectionlessData
- [Multipoint support] [4]

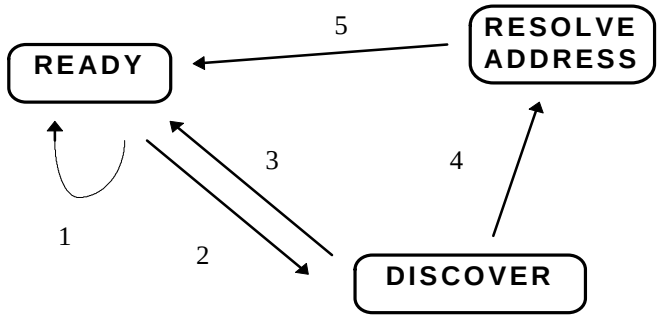
The optional service class primitives for the IAS are:

- LM_GetInfoBaseDetails
- LM_GetObjects
- LM_GetValue
- LM_GetObjectInfo
- LM_GetAttributeNames

[4] Multipoint behavior is currently not supported by IrLAP.

6.3 Minimal Station Control

6.3.1 Station Control State Transition Diagram



6.3.2 Minimal Station Control State Transition Table.

State	Event	Action	Next State	
READY	IrLAP_Connect.indication	Forward [IrLAP_Connect.indication];	READY	1
		IrLAP_Disconnect.request /* No resources to accept connection */	READY	1
	IrlAP_Connect.confirm	Forward [IrLAP_Connect.confirm]	READY	1
	IrLAP_Disconnect.indication	Forward [IrLAP_Disconnect.indication]	READY	1
	IrLAP_Status.confirm	Forward [IrLAP_Status.confirm]	READY	1
	IrLAP_Status.indication	Forward [IrLAP_Status.indication]	READY	1
	IrLAP_Reset.indication	Forward [IrLAP_Reset.indication]	READY	1
	IrLAP_Reset.confirm	Forward [IrLAP_Reset.confirm]	READY	1
	IrLAP_Discover.indication(Log)	/* Accumulate in CacheLog */ CacheLog = CacheLog ∪ Log	READY	1
		/* Replace CacheLog */ CacheLog = Log	READY	1
	IrLAP_Discover.confirm	Error /* No outstanding request */	READY	1
	IrLAP_NewAddress.confirm	Error /* No outstanding request */	READY	1
	IrLAP_Primary.indication ∧ Connected = ∅	Error /* No IrLAP connection */	READY	1
	IrLAP_Primary.indication ∧ Connected ≠ ∅	IrLAP_Primary.response(deny=true) /* Disallow the swap */	READY	1
	IrLAP_Primary.confirm	Error /* No outstanding request */	READY	1
	LS_Connect.request(deviceAddress) ∧ ((deviceAddress ∈ Connected) ∨ (Connected = ∅))	Forward [LS_Connect.request]	READY	1
	LS_Connect.request(deviceAddress) ∧ deviceAddress ∉ Connected ∧ Connected ≠ ∅ ∧ IdleIrLAPConnections = ∅	LS_Disconnect.indication (noIrLAPConnection)	READY	1
	LS_Connect.request(deviceAddress) ∧ deviceAddress ∉ Connected ∧ Connected ≠ ∅ ∧ IdleIrLAPConnections ≠ ∅	∀ IrLAP Connections ∈ IdleIrLAPConnections Forward [LS_ForceDisconnect.request] Forward [LS_Connect.request]	READY	1
	LS_Disconnect.request	Forward [LS_Disconnect.request]	READY	1
	LS_Status.request	Forward [LS_Status.request]	READY	1
	LM_AccessMode.request (mode=exclusive)	LM_AccessMode.confirm (status=localFailure,mode=multiplexed)	READY	1
	LM_AccessMode.request (mode=multiplexed)	LM_AccessMode.confirm (status=success,mode=multiplexed)	READY	1

State	Event	Action	Next State	
	IrLAP_Data.indication (AccessMode Request LM-PDU [mode=exclusive])	IrLAP_Data.request (AccessMode Confirm LM-PDU [status=unsupported mode=multiplexed], expedited=false)	READY	1
	IrLAP_Data.indication (AccessMode Request LM-PDU [mode=multiplexed])	Error /* Not in exclusive mode */	READY	1
	IrLAP_Data.indication (AccessMode Confirm LM-PDU)	Error /* No outstanding request */	READY	1
	LM_ConnectionlessData.request(data) ∧ Connected = ∅	IrLAP_UnitData.request (Data LM-PDU [DLSAP-SEL=0x70, SLSAP-SEL=0x70, data]) LM_ConnectionlessData.confirm (status=success)	READY	1
	LM_ConnectionlessData.request(data) ∧ Connected ≠ ∅	IrLAP_Data.request (Data LM-PDU [DLSAP-SEL=0x70, SLSAP-SEL=0x70, data] , expedited=true]) LM_ConnectionlessData.confirm (status=success)	READY	1
	LM_DiscoverDevices.request ∧ Connected = ∅	IrLAP_Discover.request	DISCOVER	2
	LM_DiscoverDevices.request ∧ Connected ≠ ∅	LM_DiscoverDevices.confirm (status=cache,CacheLog);	READY	1
	LM_Sniff.request(option=start)	LM_Sniff.confirm (status=refused,deviceAddress=null)	READY	1
	LM_Sniff.request(option=cancel)	Error /* Not Sniffing! */	READY	1
DISCOVER	IrLAP_Connect.indication	IrLAP_Disconnect.request /* reject the connection attempt */	DISCOVER	
	IrlAP_Connect.confirm	Error /* No pending IrLAP Connections */	DISCOVER	
	IrLAP_Disconnect.indication	Error /* No IrLAP connections */	DISCOVER	
	IrLAP_Status.confirm	Error /* No IrLAP connections */	DISCOVER	
	IrLAP_Status.indication	Error /* No IrLAP connections */	DISCOVER	
	IrLAP_Reset.indication	Error /* No IrLAP connections */	DISCOVER	
	IrLAP_Reset.confirm	Error	DISCOVER	
	IrLAP_Discover.indication(Log)	/* Ignore */	DISCOVER	
	IrLAP_Discover.confirm(Log) ∧ AddressConficts(Log) = ∅	CacheLog = Log; LM_DiscoverDevices.confirm (status=newLog, Log)	READY	3
	IrLAP_Discover.confirm(Log) ∧ AddressConficts(Log) ≠ ∅	Conflicts = AddressConflicts(Log); CacheLog = Log CacheLog = CacheLog -Conflicts; ConflictAddresses = ExtractAddresses(Conflicts); resolveAddress = ConflictAddresses[0]; ConflictAddresses = ConflictAddresses - {resolveAddress}; IrLAP_NewAddress.request (resolveAddress);	RESOLVE ADDR	4
	IrLAP_NewAddress.confirm	Error /* No Outstanding Request */	DISCOVER	
	IrLAP_Primary.indication	Error /* No IrLAP Connections */	DISCOVER	
	IrLAP_Primary.confirm	Error /* No outstanding request */	DISCOVER	
	IrLAP_Data.indication (AccessMode Request LM-PDU)	Error /* No IrLAP Connections */	DISCOVER	
	IrLAP_Data.indication (AccessMode Confirm LM-PDU)	Error /* No IrLAP Connections */	DISCOVER	
	LM_AccessMode.request	Error /* No IrLAP connection */	DISCOVER	
	LS_Disconnect.request	Error /* No IrLAP connections */	DISCOVER	

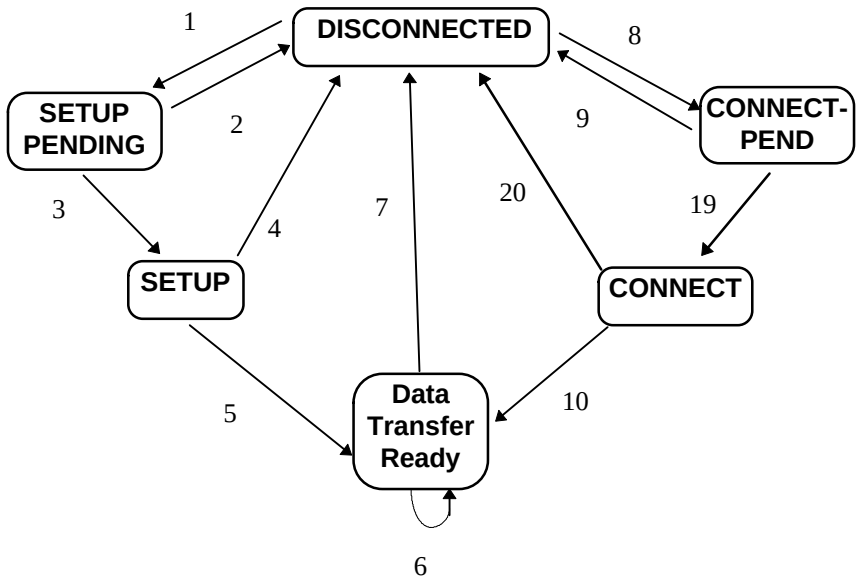
State	Event	Action	Next State	
	LS_Status.request	Error /* No IrLAP connections */	DISCOVER	
	IrLAP_Data.indication	Error /* No IrLAP connection */	DISCOVER	
	LS_Connect.request, LM_ConnectionlessData.request, LM_DiscoverDevices.request, LM_Sniff.request	***/* Left pending */***	***DISCOVER***	
RESOLVE ADDR	IrLAP_Connect.indication	IrLAP_Disconnect.request; /* Reject the connection attempt */	RESOLVE ADDR	
	IrlAP_Connect.confirm	Error /* No pending IrLAP Connections */	RESOLVE ADDR	
	IrLAP_Disconnect.indication	Error /* No IrLAP connections */	RESOLVE ADDR	
	IrLAP_Status.confirm	Error /* No IrLAP connections */	RESOLVE ADDR	
	IrLAP_Status.indication	Error /* No IrLAP connections */	RESOLVE ADDR	
	IrLAP_Reset.indication	Error /* No IrLAP connections */	RESOLVE ADDR	
	IrLAP_Reset.confirm	Error	RESOLVE ADDR	
	IrLAP_Discover.indication(Log)	/* Ignore */	RESOLVE ADDR	
	IrLAP_Discover.confirm(Log)	Error /* No outstanding request */	RESOLVE ADDR	
	IrLAP_NewAddress.confirm(Log) ∧ AddressConflicts (Log ∪ CacheLog)=∅ ∧ ConflictAddresses = ∅	CacheLog = CacheLog ∪ Log; LM_DiscoverDevices.confirm (status=newLog, CacheLog);	READY	5
	IrLAP_NewAddress.confirm(Log) ∧ AddressConflicts (Log ∪ CacheLog) = ∅ ∧ ConflictAddresses ≠ ∅	CacheLog = CacheLog ∪ Log; resolveAddress = ConflictAddresses[0]; ConflictAddresses = ConflictAddresses - {resolveAddress}; IrLAP_NewAddress.request (resolveAddress)	RESOLVE ADDR	
	IrLAP_NewAddress.confirm(Log) ∧ AddressConflicts (Log ∪ CacheLog) ≠ ∅ ∧ ConflictAddresses = ∅	Conflicts = AddressConflicts (CacheLog ∪ Log); CacheLog = CacheLog - Conflicts; LM_DiscoverDevices.confirm (status=newLog,CacheLog);	READY	5
	IrLAP_NewAddress.confirm(Log) ∧ AddressConflicts (Log ∪ CacheLog) ≠ ∅ ∧ ConflictAddresses ≠ ∅	Conflicts = AddressConflicts (CacheLog ∪ Log); CacheLog = CacheLog - Conflicts; resolveAddress = ConflictAddresses[0]; ConflictAddresses = ConflictAddresses - {resolveAddress}; IrLAP_NewAddress.request (resolveAddress)	RESOLVE ADDR	
	IrLAP_Primary.indication	Error /* No IrLAP Connections */	RESOLVE ADDR	
	IrLAP_Primary.confirm	Error /* No outstanding request */	RESOLVE ADDR	
	IrLAP_Data.indication (AccessMode Request LM-PDU)	Error /* No IrLAP Connections */	RESOLVE ADDR	
	IrLAP_Data.indication (AccessMode Confirm LM-PDU)	Error /* No IrLAP Connections */	RESOLVE ADDR	
	LM_AccessMode.request	Error /* No IrLAP connection */	RESOLVE ADDR	
	LS_Disconnect.request	Error /* No IrLAP connections */	RESOLVE ADDR	
	LS_Status.request	Error /* No IrLAP connections */	RESOLVE ADDR	
	IrLAP_Data.indication	Error /* No IrLAP connection */	RESOLVE ADDR	
	LS_Connect.request, LM_ConnectionlessData.request, LM_DiscoverDevices.request, LM_Sniff.request	***/* Left pending */***	***RESOLVE ADDR***	

6.4 Minimal IrLAP Link Connection Control

This remains unchanged.

6.5 Minimal LSAP-Connection Control

6.5.1 LSAP-Connection Control State Transition Diagram



6.5.2 Minimal LSAP-Connection Control State Transition Table

State	Event	Action	Next State	
DISCONNECTED	LM_Connect.request (userData)	connectData=userData LS_Connect.request /*Open and Bind IrLAP Connection*/ StartWatchDogTimer	SETUP-PEND	1
	LM_Connect.response	Error	DISCONNECTED	
	LM_Disconnect.request	Error	DISCONNECTED	
	LM_Idle.request	Error /* Not Supported */r	DISCONNECTED	
	LM_Data.request	Error	DISCONNECTED	
	LM_UData.request	Error	DISCONNECTED	
	LM_Status.request	Error	DISCONNECTED	
	LS_Connect.confirm	LS_Disconnect.request	DISCONNECTED	
	LS_Status.indication	Error	DISCONNECTED	
	LS_Status.confirm	Error	DISCONNECTED	
	IrLAP_Data.indication (Data LM-PDU)	IrLAP_Data.request (Disconnect LM-PDU [reason=Disconnected]) /* Data delivered on a disconnected LSAP connection is rejected with an Disconnect LM-PDU */	DISCONNECTED	
	IrLAP_Data.indication (Connect LM-PDU[userData])	connectData=userData LS_Connect.request /* Bind to IrLAP Connection .indication delivered to LSAP User following LS_Connect.confirm */	CONNECT-PEND	8

State	Event	Action	Next State	
	IrLAP_Data.indication (Connect confirm LM-PDU)	IrLAP_Data.request (Disconnect LM-PDU [reason=Disconnected]) /* Connection confirmation delivered on a non existent LSAP-connection is rejected with a disconnect LM-PDU. */	DISCONNECTED	
	IrLAP_Data.indication (Disconnect LM-PDU)	Error	DISCONNECTED	
	WatchDogTimeOut	/* Ignore */	DISCONNECTED	
CONNECT-PEND	LM_Connect.request	LM_Disconnect.indication (reason=incomingConnection)	CONNECT-PEND	
	LM_Connect.response (userData)	Error /* No .indication issued yet */	CONNECT-PEND	
	LM_Disconnect.request	Error /* Not yet Bound to IrLAP connection */	CONNECT-PEND	
	LM_Idle.request	Error /* Not Supported */r	CONNECT-PEND	
	LM_Data.request	Error	CONNECT-PEND	
	LM_UData.request	Error	CONNECT-PEND	
	LM_Status.request	Error /* Not yet Bound to IrLAP connection */	CONNECT-PEND	
	LS_Connect.confirm /*Bound to IrLAP Connection*/	LM_Connect.indication (connectData)	CONNECT	19
	LS_Disconnect.indication (reason)	LM_Disconnect.indication(reason)	DISCONNECTED	9
	LS_Status.indication(status)	Error /* Not yet Bound to IrLAP connection */	CONNECT-PEND	
	LS_Status.confirm(status)	Error /* Not yet Bound to IrLAP connection */	CONNECT-PEND	
	IrLAP_Data.indication	Error	CONNECT-PEND	
CONNECT	LM_Connect.request	LM_Disconnect.indication (reason=incomingConnection)	CONNECT	
	LM_Connect.response (userData)	IrLAP_Data.request (Connect Confirm LM-PDU[userData], expedited=false)	DTR	10
	LM_Disconnect.request	IrLAP_Data.request (Disconnect LM-PDU [reason=userRequest], expedited=false) LS_Disconnect.request /* Unbind IrLAP connection */	DISCONNECTED	9
	LM_Idle.request	Error /* Not Supported */r	CONNECT	
	LM_Data.request	Error	CONNECT	
	LM_UData.request	Error	CONNECT	
	LM_Status.request	LS_Status.request	CONNECT	
	LS_Connect.confirm	Error	CONNECT	
	LS_Disconnect.indication (reason)	LM_Disconnect.indication(reason)	DISCONNECTED	20
	LS_Status.indication(status)	LM_Status.indication(status)	CONNECT	
	LS_Status.confirm(status)	LM_Status.confirm(status)	CONNECT	
	IrLAP_Data.indication	Error	CONNECT	
DATA-TRANSFER-READY (DTR)	LM_Connect.request	Error	DTR	
	LM_Connect.response	Error	DTR	
	LM_Disconnect.request	IrLAP_Data.request (Disconnect LM-PDU [Reason=UserRequest], expedited=false) LS_Disconnect.request /* Unbind IrLAP connection */	DISCONNECTED	7
	LM_Idle.request	Error /* Not Supported */r	DTR	

State	Event	Action	Next State	
	LM_Data.request(userData)	IrLAP_Data.request (Data-LM-PDU[userData], expedited=false)	DTR	
	LM_UData.request(userData)	IrLAP_Data.request (Data-LM-PDU[userData], expedited=true)	DTR	
	LM_Status.request	LS_Status.request	DTR	
	LS_Connect.confirm	Error	DTR	
	LS_Disconnect.indication (reason)	LM_Disconnect.indication(reason)	DISCONNECTED	7
	LS_Status.indication(status)	LM_Status.indication(status)	DTR	
	LS_Status.confirm(status)	LM_Status.confirm(status)	DTR	
	IrLAP_Data.indication (Data LM-PDU[userData], expedited=false)	LM_Data.indication(userData)	DTR	
	IrLAP_Data.indication (Data LM-PDU[userData], expedited=true)	LM_UData.indication(userData)	DTR	
	IrLAP_Data.indication (Connect LM-PDU)	IrLAP_Data.request (Disconnect LM-PDU[reason=halfOpen], expidited=false) LS_Disconnect.request LM_Disconnect.indication(reason=halfOpen)	DISCONNECTED	7
	IrLAP_Data.indication (Connect confirm LM-PDU)	Error	DTR	
	IrLAP_Data.indication (Disconnect LM-PDU [reason])	LS_Disconnect.request LM_Disconnect.indication(reason)	DISCONNECTED	7
	IrLAP_Data.indication (Disconnect LM-PDU)	Error	DTR	
	WatchDogTimeOut	/* Ignore */	DTR	
SETUP-PEND	LM_Connect.request	Error	SETUP-PEND	
	LM_Connect.response	Error	SETUP-PEND	
	LM_Disconnect.request	Error	SETUP-PEND	
	LM_Idle.request	Error /* Not Supported */r	SETUP-PEND	
	LM_Data.request	Error	SETUP-PEND	
	LM_UData.request	Error	SETUP-PEND	
	LM_Status.request	Error	SETUP-PEND	
	LS_Connect.confirm	IrLAP_Data.request (Connect LM-PDU[connectData], expedited=false) StartWatchDogTimer	SETUP	3
	LS_Disconnect.indication (reason)	LM_Disconnect.indication(reason) CancelWatchDogTimer	DISCONNECTED	2
	LS_Status.indication(status)	LM_Status.indication(status)	SETUP-PEND	
	LS_Status.confirm(status)	LM_Status.confirm(status)	SETUP-PEND	
	IrLAP_Data.indication	Error /* Can't receive data - not 'bound' to an IrLAP connection */	SETUP-PEND	
	IrLAP_UnitData.indication	Error	SETUP-PEND	
	WatchDogTimeOut	LS_Disconnect.request	DISCONNECTED	2
SETUP	LM_Connect.request	Error	SETUP	
	LM_Connect.response	Error	SETUP	
	LM_Disconnect.request	Error	SETUP	
	LM_Idle.request	Error /* Not Supported */r	SETUP	
	LM_Data.request	Error	SETUP	
	LM_UData.request	Error	SETUP	
	LM_Status.request	LS_Status.request	SETUP	
	LS_Connect.confirm	Error	SETUP	
	LS_Disconnect.indication (reason)	LM_Disconnect.indication(reason) CancelWatchDogTimer	DISCONNECTED	4
	LS_Status.indication(status)	LM_Status.indication(status)	SETUP	
	LS_Status.confirm(status)	LM_Status.confirm(status)	SETUP	

State	Event	Action	Next State	
	IrLAP_Data.indication (Data LM-PDU)	Error	SETUP	
	IrLAP_Data.indication (Connect LM-PDU)	/* No need to send Disconnect - peer will see matching Connect */ LS_Disconnect.request LM_Disconnect.indication (connectionRace) CancelWatchDogTimer	DISCONNECTED	4
	IrLAP_Data.indication (Connect confirm LM-PDU [userData])	LM_Connect.confirm(userData) CancelWatchDogTimer	DTR	5
	IrLAP_Data.indication (Disconnect LM-PDU [reason])	LS_Disconnect.request LM_Disconnect.indication(reason) CancelWatchDogTimer	DISCONNECTED	4
	WatchDogTimeOut	LS_Disconnect.request LM_Disconnect.indication (nonResponsivePeer)	DISCONNECTED	4

7. Appendix C: Examples

The following sections illustrate how IrLMP is intended to work.

7.1 Top Level Client/Server Example

IrLMP provides the mechanisms that enable 'ad-hoc' communication between clients of the Link Management Multiplexer (LM-MUX). For client/server styles of interaction an LM-MUX client in one station (an IrDA device) will behave as a service provider and offer some service e.g., Printing, Fax Transmission, File Transfer and/or access and so forth.

A service provider will advertise the existence of a particular service by creating an instance of a object in the Information Base managed by the Link Management Information Access Service (LM-IAS). The class of the object instance defines the nature of the service being offered. The attributes attached to the object instance carry information that is useful to a potential client of the service: firstly in determining whether the offered service meets the requirements of the service client; and secondly by indicating how the service client can contact the service provider.

A service client that is seeking a particular type of service provider must first engage in XID Discovery. This will result in a list of IrLAP device addresses with corresponding device nicknames and service hints. The nicknames and service hints are used by the service client to refine the list of device address in order to minimize number of potential service providers it examines in more detail. Detailed examination of potential service providers is performed by examining the LM-IAS Information Base of the potential provider(s). The service client application may interact with the end user to resolve which of a number of similar service providers should be chosen.

Once the service client has determined which service provider it wishes to use then it attempts to make contact with it in the manner described by the corresponding LM-IAS object instance. For the purposes of this example it is assumed that both service client and service provider are direct clients of LM-MUX. The LM-IAS service is itself an example of a service that is a direct client of LM-MUX. Other connection methods are available, for example the connection between service client and service provider may be established by the use of a transport protocol rather than through direct interaction with LM-MUX.

In summary a normal client/server interaction proceeds as follows:

At the service client, an LM-MUX client (e.g., a portable PC):

- An LSAP is retrieved
- The LSAP is bound locally
- `LM_DiscoverDevices.request` is issued (and .confirm is awaited)
- `LM_GetAttributeByClass.request` is issued (and .confirm is awaited)
- `LM_Connect.request` is issued (and .confirm is awaited)
- `LM_Data.request` and `LM_Data.indication` are exchanged (multiple times) between service client and service provider to exchange data over the resulting LSAP connection.
- `LM_Disconnect.request` is issued

At the service provider, an LM-MUX client (e.g., a Fax Machine):

- An LSAP is retrieved
- The LSAP is bound locally
- An object is added to the information base
- Attributes of the new object, including connection information, are added to the information base.

- `LM_DiscoverDevices.indication` is received which indicates the service providing device is in the process of being discovered.
- `GetValueByClass.indication` is received and appropriate values are returned
- `LM_Connect.indication` is received and `LM_Connect.response` is returned to accept the LSAP-connection.
- `LM_Data.request` and `LM_Data.indication` are exchanged (multiple times) between service client and service provider to exchange data over the resulting LSAP connection.
- `LM_Disconnect.indication` is received to inform the service provider that the LSAP-connection has closed.

7.2 LSAP-Connection Examples

Each of the following subsections contains figures. The first simply shows the exchange of service primitives at the interface between an LM-MUX client and LM-MUX itself. The second expand on the first by should the exchange of LM-PDUs between peer LM-MUX entities and the invocation of IrLAP service primitives.

7.2.1 Accepted Connection

The initiating LM-MUX client invokes **LM_Connect.request** at its local LSAP-connection endpoint. An **LM_Connect.indication** is received at the intended peer LM-MUX client. The responding client accepts the LSAP-connection by invoking **LM_Connect.response** which results in the invocation of an **LM_Connect.confirm** at the initiating LM-MUX client.

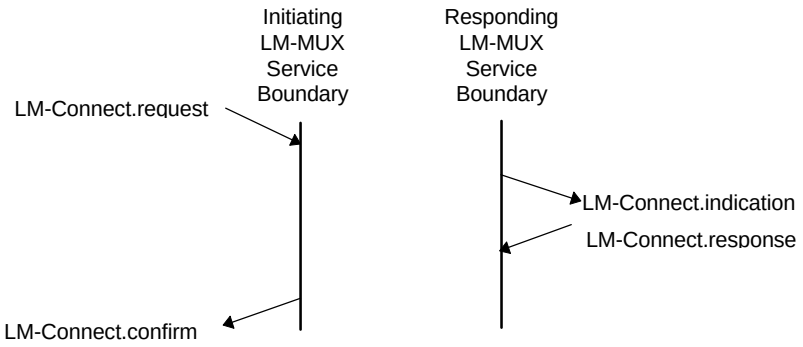


Figure 8. Accepted Connection

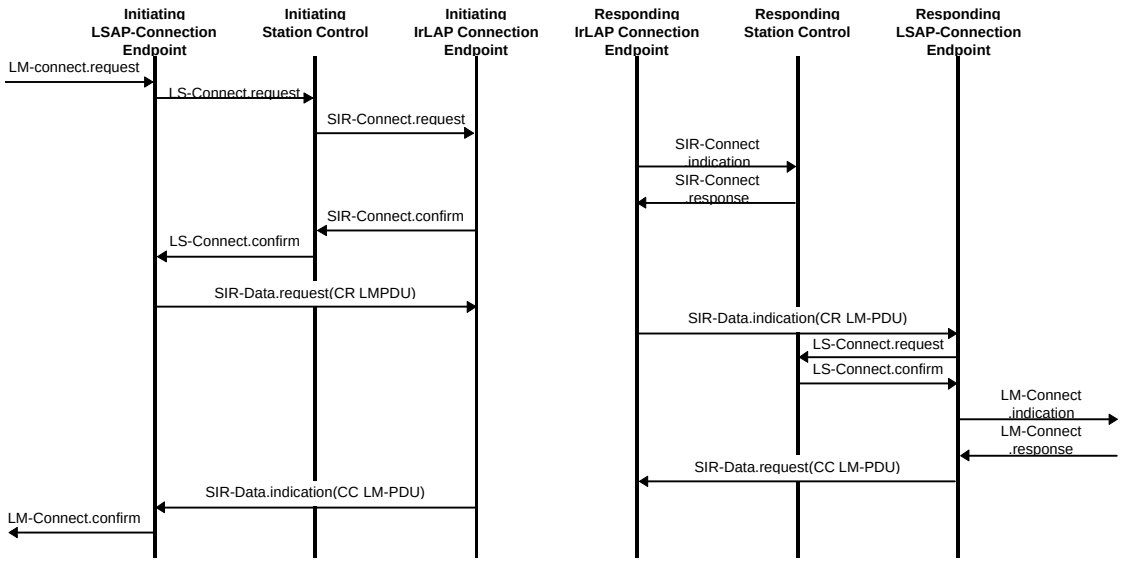


Figure 9. Accepted Connection - Detailed Primitive and PDU Exchange

For LSAP-connections established over an existing IrLAP connection the four IrLAP_Connect primitives should be deleted from the previous diagram.

7.2.2 Connection Rejection

The initiating LM-MUX client invokes **LM_Connect.request** at its local LSAP-connection endpoint. An **LM_Connect.indication** is received at the intended peer LM-MUX client. The responding client rejects the LSAP-connection by invoking **LM_Disconnect.request**.

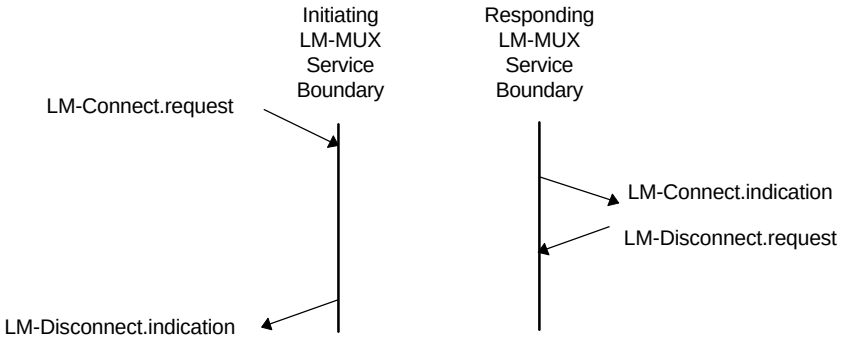


Figure 10. Rejected Connection

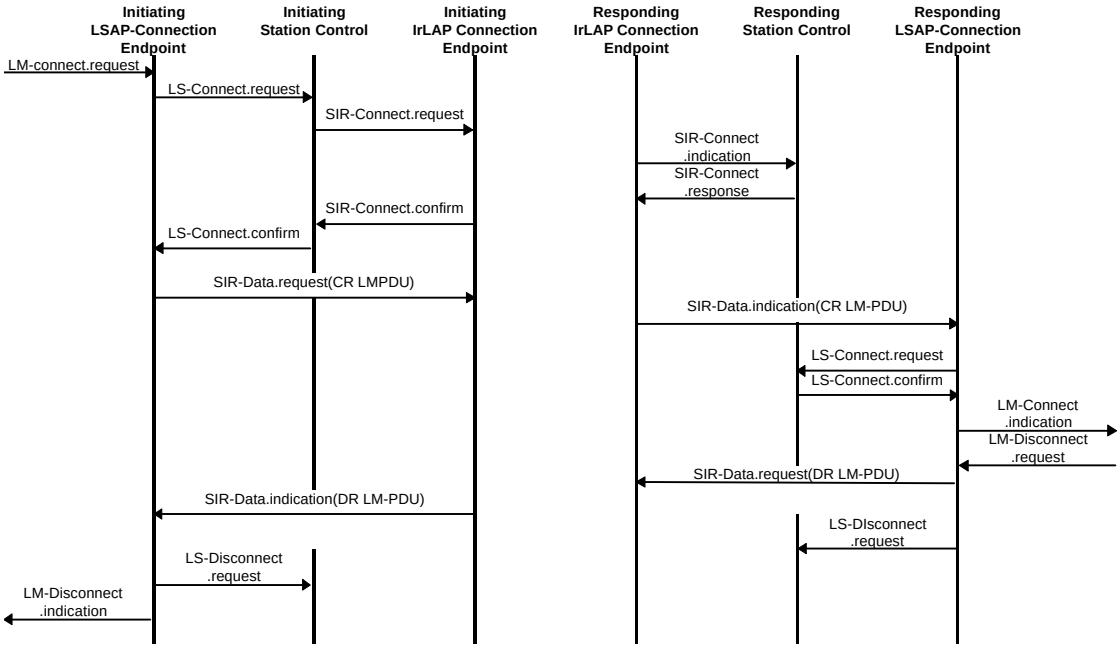


Figure 11. Rejected Connection - Detailed Primitive and PDU Exchange

7.2.3 Race Condition

A connection race occurs both ends of a connection attempt to establish the connection at the same time. IrLMP LSAP-connections are defined such that both connection attempts fail and reason reported in the resulting LM_Disconnect.indication indicates that the failure was caused by this race.

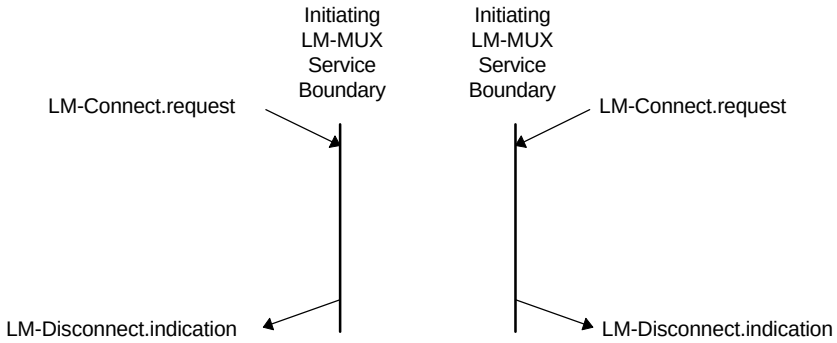


Figure 12. Connection Race

The figure below assumes that an IrLAP connection has already been established. IrLAP resolves IrLAP-connection races an yields a single IrLAP connection. Because the LM_Connect primitives carry user data it is not possible to resolve a LSAP-connection race in the same way. The connection race shown is perfectly symmetrical, however in practice the window in which a race can occur is between the **LM_Connect.request** and the corresponding **LM_Connect.confirm**.

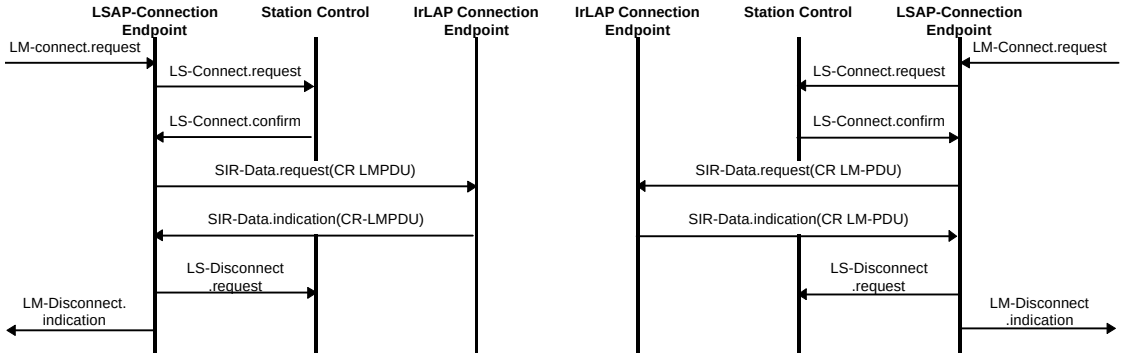


Figure 13. Connection Race - Detailed Primitive and PDU Exchange

7.2.4 Failure to Establish IrLAP Connection

The initiating LM-MUX client invokes **LM_Connect.request** at its local LSAP-connection endpoint. Since the there is a failure to establish the underlying IrLAP connection the responding LM-MUX client is completely unaware of the connection attempt.

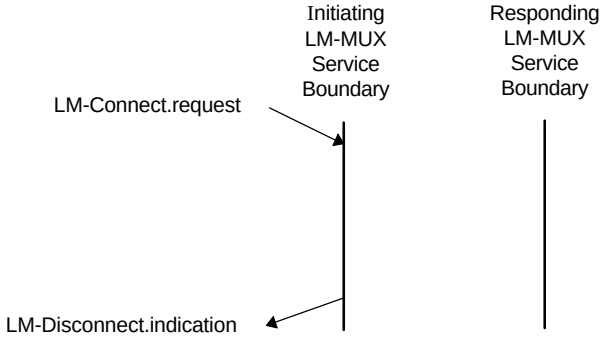


Figure 14. IrLAP-Connection Establishment Failure

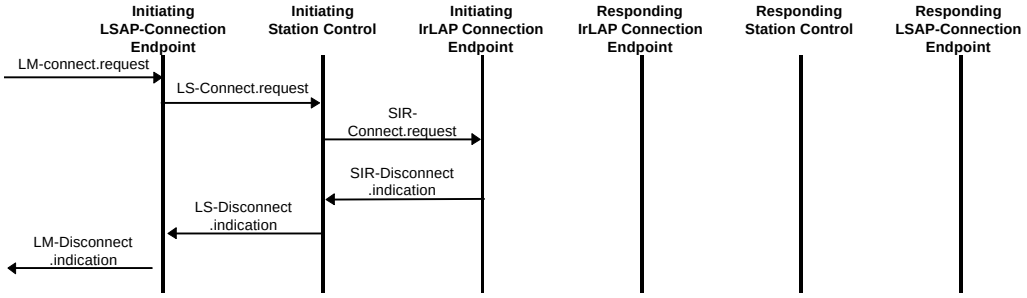


Figure 15. IrLAP-Connection Establishment Failure - Detailed Primitive and PDU Exchange