LASTport/Disk Architecture Specification

Order Number: EK–LASTD–AS–001

March 1992

This document describes the LASTport/Disk architecture and protocol. The architecture specifies usage of the protocol in the context of a local area network (LAN) client-server model.

The LASTport/Disk message protocol is described in sufficient detail to facilitate interoperable implementations.

Revision/Update Information: This is a new document.

Software Version: LAD Version 3.1

This document was prepared using VAX DOCUMENT, Version 2.0.

Contents

Tables

Preface

Intended Audience

This document addresses implementers of the LASTport/Disk architecture and programmers porting the architecture to various environments. Readers are expected to be familiar with local area network (LAN) concepts and transport protocols.

Document Structure

This document contains three chapters and four appendixes.

- Chapter 1 presents an overview of the LASTport/Disk architecture.
- Chapter 2 describes the interfaces to the user application and the LASTport Association layer.
- Chapter 3 describes LASTport/Disk Version 3.1 message formats.
- Appendix A describes LASTport/Disk Version 3.0 message formats.
- Appendix B describes LASTport/Disk Version 2.0 message formats.
- Appendix C explains mapping LASTport/Disk Version 3.0 and later to LASTport Version 3.1.
- Appendix D lists global constants, ranges, and values.

Associated Documents

For information on the LASTport Transport, which is required for LASTport/Disk operation, refer to the *LASTport Architecture Specification* document.

Conventions

Table 1 lists typographical conventions used in this document. Table 2 lists architecture format types. All numeric values are decimal unless specified otherwise.

Table 1 Typographic Conventions

Convention	Meaning
Initial Caps	Names of messages, such as Solicit Request, or names of architecturally controlled variables, such as Service Name.
italics	Names of events, such as *AppConnect* or *AsnTransComplete*, that are generated or processed by LAST, LAD, or user applications.
UPPERCASE	Names of message fields, such as CUR_PRTCL_VER or STATUS.
MixedCase	Names of processes, such as AsnClientRcv; names of routines and variables in process pseudo-code examples, such as GetNextService or dataRequestLength; names of timers, such as SolicitResponseTimer or AsnConnectResponseTimer.
TAZIOR	Transmit As Binary Zeros And Ignore On Receipt. Specifying TAZIOR in message format fields allows Engineering Change Order (ECO) extensions to the protocol without changing major version numbers.

Table 2 Architecture Format Types

Type	Meaning
Bit mask	A variable-length field (in bits), in which each bit has an assigned meaning. Bit value 1 indicates "true" or "do" relative to the assigned bit meaning.
Bytes (8 bits) Words (16 bits) Longwords (32 bits)	These fields describe either signed or unsigned integers. Unsigned integers can range in value from 0 to 255 (bytes), 65,535 (words) and 2**32–1 (longwords). Signed integers are specified in two's complement form; bytes range from –128 to 127, words from –32,768 to 32,767 and longwords from –2**31 to 2*31–1.
Name	Unsigned byte-counted string containing the characters described in Section 1.3.5.
Text string	Unsigned byte-counted string containing the characters described in Section 1.3.4.

1 Overview of LASTport/Disk Architecture

The LASTport/Disk architecture is media independent. It specifies a simple, efficient method of accessing named collections of logical blocks, called virtual disks, in a heterogeneous system environment. Client systems see the virtual disks as though they are locally attached devices. The LASTport/Disk server supports a directory service of locally resident virtual disks that enables client systems to find and access these disks. Device control mechanisms facilitate allocation of drives and their storage devices.

The architecture provides structures to accomplish the following kinds of tasks:

- Let a user of one of the major operating system **platforms** allocate a publicly accessible drive (by a coffee stand, for instance), load the media of choice, and access the media from that platform (such as a VMS system or an MS–DOS system)
- Let a facility manager offer to all users publicly accessible media that users can access and share from heterogeneous system platforms
- Specify system-independent and system-dependent naming conventions concurrently for both media and disk drives
- Create named virtual disks (on read-write media) “on demand” for use by heterogeneous client systems

This chapter introduces the LASTport/Disk architecture. Topics include

- Disk service management
- Protocol messages
- Message formats
- Message processing
- Revision control

1.1 Disk Service Management

The LASTport/Disk architecture provides the following disk service management facilities:

- Name Spaces and Name Space Handlers
- Load balancing
- Disk cache policies and attributes
- Connect/preserve semantics
- Create/connect semantics

Sections 1.1.1 through 1.1.5 discuss these facilities.

1.1.1 Name Spaces and Name Space Handlers

The architecture defines a set of typed **Name Spaces** that contain **Service Names**. Users access media by specifying these Service Names, which are the names of virtual disks. Multiple virtual disks or a single disk can be mapped to a single physical medium. Conversely, a single virtual disk can span multiple physical media.

Each Name Space has potentially unique capabilities that are implemented by **Name Space Handlers**. For instance, one handler might define a Name Space for the International Standards Organization (ISO) 9660 media format, and a different handler might define a Name Space for "scratch" (page file) storage applications.

The service and message definitions available to Name Space Handlers enable each handler to offer the following facilities independently:

- Solicitation services to determine the physical location of virtual disks and drives
- Directory services appropriate to the Name Space
- Device-handling services
- Storage allocation policies

The architecture requires implementations to provide local services to manage the following:

- Allocation and deallocation of drives
- Allocation and deallocation of available storage
- The database of named servers, drives, and media

The architecture also requires each server implementation to provide nonvolatile storage. This storage is used to save the states of the server and the local media on the server, so that these states can be restored consistently after server failures.

Each Name Space Handler independently uses these common services to implement its own policies. Thus, a free drive could be allocated for ISO 9660 use or for use as a page file (but not for both concurrently). If the drive is allocated by a client but never accessed, the server implementation is expected to detect and respond appropriately to this condition — that is, to make the drive available again. Similarly, the server implementation is expected to handle cases in which page file storage is allocated but never used.

The LASTport/Disk architecture allows multiple types of operating-system-specific media to be present concurrently in a network and on a server. The architecture also provides each client system with a view of the media subset that is of interest to that client system. Because the architecture provides a default view that is normal for a particular operating system (for instance, the VMS operating system), VMS clients can, by default, access On-Disk Structure Level 2 (ODS-2) media without preventing VMS users from accessing "foreign" media.

1.1.2 Load Balancing

When multiple identical read-only media are available concurrently on different physical drives and different physical servers, the architecture provides a mechanism to balance the load across these drives and servers when connections are made and balance requests at run time.

Thus, when publicly served media are offered on a server that fails or becomes so overloaded with requests that it becomes unusable, the architecture provides mechanisms to prevent these failures from affecting users — as long as identical media are still available to client systems.

The LASTport/Disk protocol provides static load balancing across duplicate media within an extended LAN. When client systems request access to a service, they select among duplicate services by selecting the service with the highest rating. A higher rating indicates a more readily available service for client connections; a lower rating indicates lower availability. Whenever a service is offered more than once on the LAN network, clients always select the service with the highest rating,

By default, the server system dynamically calculates a rating based on the load for each service. For disks, the dynamic rating for each service is calculated and updated regularly (for example, every 10 seconds). The following algorithm is used to calculate the dynamic rating for disks, within the LASTport/Disk protocol range of 0 to 65535:

$$\text{Rating} = (32768 * \%CPU\_FREE) + \\ (32767 * ((0.9 * last\_calc\_srvd\_load) + (0.1 * new\_calc\_srvd\_load)))$$

The first term, called the server load factor, is an instantaneous look at the CPU percentage, weighted over one-half the rating scale.

The second term, called the service load factor, is 90 percent of the previously calculated load factor plus 10 percent of the current calculated load factor, weighted over one-half the rating scale. The previously calculated load factor is then replaced with the current calculation of the service load factor. This produces a moving weighted average that is calculated for each service at each timer interval. The formula for the *new_calc_srvd_load* term is as follows:

$$new\_calc\_srvd\_load = \frac{1000 - \min(1000,\ \text{I/Os to service in last 10 sec})}{1000}$$

As a result, a load is imposed on the service based on the number of block reads in the past 10 seconds. The term *last_calc_srvd_load* is then updated with the new service load factor calculated.

1.1.3 Disk Cache Policies and Attributes

While the LASTport/Disk protocol defines cache policies and attributes, the server is not obligated to implement any caching policies. Note that distributed systems that attempt to hint about policies rely on these assumptions when making tradeoffs. For example, a client might set the readahead attribute for a particular service knowing that future reads to that service will always be sequential.

1.1.4 Create/Connect Semantics

Version 3.1 servers allow clients to allocate and name virtual disks dynamically. Servers respond to Solicit Request and Connect Request messages to inform clients that storage is available.

Clients use the CREATE/CONNECT_SERVICE option to allocate temporary or permanent disk space. Examples of uses for temporary disk space are page files, LAT service names, and scratch disk space. This space is usually allocated with a delete on disconnect policy.

1.1.5 Connect/Preserve and Update Semantics

Version 3.1 servers offer a special service, which helps the client to preserve the state of data on the virtual disk. When a client connects to a read/write service with the CONNECT/PRESERVE_SERVICE bit set in the CONNECT_MODIFIER field of the Connect Request message (see Table 3–8), the server preserves the data on the virtual disk until the client issues an update request (see Table 3–19).

The server maintains a master copy of the virtual disk and applies all writes to a temporary copy. When the update request is issued, the server performs an atomic operation that remaps the temporary copy of the virtual disk to the master copy. This feature allows clients to perform non-atomic updates while guaranteeing that the state of the data remains consistent. For example, if a client writes data to a virtual disk, and the connection is lost before the update request is issued, all write operations are void.

Each time the client issues an update request, the server copies the temporary copy of the virtual disk to the master copy before allowing writes to continue. Note that such an update operation can be expensive in terms of CPU utilization and I/O bandwidth.

An alternative to making a temporary copy of the master disk is to implement a journaling scheme that preserves the atomicity of update operations. Regardless of the update algorithm, the server must guarantee the semantics of the update operation through all failure modes.

1.2 Protocol Messages

The LASTport/Disk message protocol consists of a wire protocol and specific rules governing the transmission and reception of wire protocol messages. The protocol syntax is extensible; specifically, it allows for future implementation of security policies.

LASTport/Disk protocol messages are structured as request messages and their corresponding response messages. A client system sends a request message to a server, which executes a transaction one or more times and then sends an appropriate response message to the client. The server never generates unsolicited messages.

The LASTport/Disk protocol requires that each server transaction be idempotent relative to itself and commutative relative to other concurrently executing transactions. This requirement ensures that transactions can be executed multiple times while the server maintains a consistent global view of the data.

Servers therefore utilize minimal memory for network buffering. The system application protocol can be implemented in a small number of messages when errors do not occur, and it can perform efficient error recovery, often providing improved availability.

The LASTport/Disk protocol defines the following messages:

- Resource management messages

  Solicit Request (message format)
  Solicit Response (message format)
  Solicit Summary Request (message format)
  Solicit Summary Response (message format)

- Connection management messages

  Connect Request (message format)
  Connect Response (message format)
  Disconnect Request (reason, reason descriptor)
  Disconnect Response (reason, reason descriptor)

- Data exchange and control messages

  Data Request (data)
  Data Response (data)
  Parameter Request (message format)
  Parameter Response (message format)

Solicit messages locate services to which connections are established using the Connect messages. Data can be sent until either the user or the system sends a Disconnect Request message.

1.3 Message Formats

Message formats are specific to each message type. They encapsulate information about Name Spaces, Service Names, device types, access modes, and other information required to find disks on the network and **connect** to them.

The LASTport/Disk message protocol specifies the contents of each message format so that the necessary receive-processing information (such as version control, Name Spaces, and so forth) is present. The message formats can be extended for specific Name Space processing using **Parameter Codes**.

The message formats are both flexible and extensible. Features like the following make them useful across multiple system platforms:

- Name Spaces
- Service Names and Service Instances
- Parameter Lists
- Name Strings
- Server Names

1.3.1 Name Spaces

Message formats contain a field for a Name Space type. The SERVICE_CLASS field is a 16-bit positive integer that describes the use and underlying file structure for the disk being serviced.

The LASTport/Disk client is responsible for interpreting the on-disk structure of a virtual disk. If a client connects to a virtual disk with a foreign on-disk structure, unpredictable results might occur. The LASTport/Disk protocol uses Name Spaces to partition services offered to its clients. For example, MS–DOS virtual disk services (named collections of logical blocks) exist in a well-defined Name Space, visible only to MS–DOS clients. Thus, Name Spaces are aligned with operating system on-disk structures but are not architecturally limited to this use. Clients can bind to virtual disks in multiple Name Spaces; for example, a VMS client could connect to a virtual disk in the ODS-2 Name Space and to a virtual disk in the ISO 9660 Name Space.

The LASTport/Disk protocol provides the global Name Spaces defined in Table 1–1.

Table 1–1 Global Name Spaces

Value	Description
0, 1	Reserved for use by Digital Equipment Corporation.
2	MS_DOS. The on-disk structure used by the MS–DOS operating system.
3	ODS_2. The on-disk structure used by the VMS and RSX operating systems.
4	ISO_9660. The CDROM on-disk structure used by the International Standards Organization.
5	ULTRIX. The on-disk structure used by the ULTRIX operating system, which is identical to that used by Berkeley UNIX Version 4.2.
6	HIGH_SIERRA. The on-disk structure used by personal computers.
7 through 10	VXT. The on-disk structure used by windowing terminals for dynamic page files and configuration files.
11	HFS. The on-disk structure used by Apple Computer, Inc. for Macintosh II computers.
12 through 65,534	Reserved for use by Digital and by Digital's licensees, as assigned by Digital Equipment Corporation.
65,535	Wildcard Name Space flag.

1.3.2 Service Names and Service Instances

An architecture model for heterogeneous systems must accommodate the needs of both platform-specific and platform-independent naming. The LASTport/Disk protocol does this by specifying a Service Name . This Service Name can be defined independently by each Name Space. LASTport/Disk Service Names typically contain names of media or of physical drives. The Service Names are system independent.

When a user requests a service, the Service Name is conveyed in the SERVICE_NAME field of the Solicit Request message. (Some Solicit Request messages specify a null Service Name, such as when the user specifies a device instead of a service.) Servers translate the requested Service Name and return a **Service Instance** in the Solicit Response message. For example, a client might **solicit** with the service name CD_DOC, and the server could return CD_DOC_V2 as the Service Instance. Implementations must copy the Service Instance into the corresponding Connect Request message SERVICE_INSTANCE field. Thus, clients connect to a Service Instance and not to a Service Name.

A Connect Response message might also contain a Service Instance that can be used in subsequent Solicit Request messages.

If a server creates a service but does not provide a Service Instance, the server generates one. Because most remotely created services must be unique, the server creates a name with two variable fields. To ensure that the name is unique, the first field is the server's network media access control (MAC) address. The second field is a 2-byte modulo counter that ensures uniqueness of Service Instances. The two variable fields are preceded by a fixed field. The two variable fields are preceded by a fixed field (VXT), and all fields are separated by an underscore. An example of a server-generated name is VXT_08002B010203_0058.

1.3.3 Parameter Lists

For extensibility purposes, the LASTport/Disk architecture defines **Parameter Lists**, which are designed to allow ECO extensions to the message formats or Name Space–specific extensions. Each service message provides for the inclusion of Parameter Lists. Parameter Lists are formed from Parameter Code, Parameter Length, and Parameter Data fields. Parameter Codes are of two types: global and user (Name Space).

Both global and user Parameter Code assignments are allocated in the range of 1 through 255. The protocol definition contains the global Parameter Codes and their interpretation, as well as the assignment of the user Parameter Codes. Each Name Space independently defines how user Parameter Codes are to be interpreted.

1.3.4 Text Strings

Some Parameter Codes contain fields that represent textual or descriptive information. Valid characters present in these fields are 16 (2/0) to 70 (7/14) and 90 (10/10) to 253 (15/15). A text string length is encoded as an unsigned byte count followed by the textual information.

1.3.5 Name Strings

Solicit Request messages and some Parameter Codes convey architecturally specified names called **Name Strings**. Name Strings are limited to the characters listed in Table 1–2

Table 1–2 Name String Characters

Character Codes (decimal)	Meaning
32–127	All printable C0 characters (SP through DEL).
160–255	All printable C1 characters.
1	Wildcard character. Matches any number of any characters in the current character position.
2	Wildcard character. Matches exactly one character in the current character position.

A Name String length is encoded as an unsigned byte count followed by the character defined in Table 1–2. All characters in Name Strings are uppercased before they are compared. Uppercasing means subtracting the value 32 from the lowercase letters in the range 97 through 122, and subtracting 32 from the character set in the range 224 through 255.

Because characters are uppercased before comparison, lowercase characters are for display purposes only. If two different systems use the same Name String, but one Name String is in lowercase and the other in uppercase, solicitation might result in unintended services being bound.

Only those terminals that support the international character set can specify Name Strings using that character set.

The LASTport/Disk protocol permits wildcard operations for services offered on an extended local area network (LAN). If a command includes a wildcard string, all objects matching the wildcard string are used in the specified operation. For example, a value of 1 or 2 in a Service Name indicates that the string can match multiple Service Names at the server.

1.3.6 Server Names

Servers are required to initialize and use a unique **Server Name** within the network in which the server offers service. Because the physical address of each network **data link port** is guaranteed to be unique, that address can be used reliably to form a unique server name. The recommended procedure is to compose the name from the network address derived from the algorithm described in the Ethernet specification document. The node name can be formed by combining the facility code and the Ethernet address with the hyphens removed. When servers are not directly attached to an Ethernet, a similar algorithm should be used to construct a unique Server Name.

For example, each InfoServer system is assigned a name (Name String) that uniquely identifies the unit on an extended LAN. This name is also used as a LAT management service that is advertised to the LAN network.

When the InfoServer system is shipped, the InfoServer software assigns each InfoServer unit a unique name. The default name is in the form LAD_*xxxxxxxxxxxx*, where *xxxxxxxxxxxx* is the hexadecimal ASCII representation of the InfoServer system's network adapter address; for instance, LAD_08002B15009F.

The network adapter address is transmitted in canonical form: bytes are transmitted from left to right with the least significant bit of each byte transmitted first. For example, the address 08-00-2B-15-00-9F is transmitted as follows:

```
0001 0000 0000 0000 1101 0100 1010 1000 0000 0000 1111 1001
```

Using the Name String characters defined in Table 1–2, implementers can create a Server Name that is meaningful in a particular environment.

1.4 Message Processing

LASTport/Disk message processing facilities include the following:

- Access modes
- Service parameters
- Service passwords
- Solicit Response policies

Sections 1.4.1 through 1.4.4 describe these facilities. Table 1–3 summarizes response policies for Solicit Request messages. Table 1–4 provides examples of Solicit Request messages and client-supplied values.

1.4.1 Access Modes

A local manager at the server sets access mode policies for objects such as devices and virtual disks. The server uses the RSP_ACCESS_MODE field in the Connect Response message to indicate which access modes are granted, either for a specific access request by a client, or for a default access mode when the client has not specified one.

When a device or storage medium is being allocated to a specific client as the result of a successful solicitation process, the client can specify “allow read-write” access in the ACCESS_MODE_MASK field. In this case, the server applies the requested access to the allocated resource. Additionally, the client can specify the maximum number of readers and writers for the allocated resource. A value of 0 indicates that the access is turned off, and a value of 1 indicates exclusive access for that mode. If server policy does not allow the specified mode, the server returns an appropriate error in the Connect Response message. To indicate that a successful connection has been established to an object, the server returns relevant session and device information in the Connect Response message.

1.4.2 Service Parameters

Generally, service parameters can be modified only for remotely created services. At creation time, parameters can be modified with the Connect Request message, and during an existing session they can be modified with the Parameter Request message. When a connection is established to an existing service that has not been created remotely, only the following parameters can be modified:

- REQUEST_READ_ACCESS (in the ACCESS_MODE_MASK field)
- REQUEST_WRITE_ACCESS (in the ACCESS_MODE_MASK field)
- SRVC_PASSW

The ACCESS_MODE_MASK field allows the SRVC_PASSW (service password) to be verified when connecting for read-only, write-only, or read-write sessions.

1.4.3 Service Passwords

Service passwords are set using a local management interface or by LASTport/Disk messages. If a service (for instance, a virtual disk) is protected with a password, the client must specify the correct password in Connect Request and Parameter Request messages to receive a response. If the password is invalid, or if a message does not specify a password when one is required, the server takes no action requested in the message and responds that access has been denied.

The protocol supports password protection for virtual disk services. Several modes of protection are available. The protocol defines a bitmask that instructs the server to check the password for read, write, or read-write connections. The Connect Request message REQ_ACCESS_MODE_MASK and Connect Response message RSP_ACCESS_MODE_MASK fields have bits that define the access modes. A single access mode can be password protected, while another mode has free access. For example, a network manager could allow any user to connect to a particular service with read-only access and require a password for write access to the same service. This mechanism controls the ability to update the data without hindering read access.

The LASTport/Disk protocol has two password parameters:

- The service password parameter (SRVC_PASSW), which is used to protect an individual service. If this parameter is null, the service is not protected, and no password is needed to access it.

  When a new, protected service is created, the password for the service is found in the service password parameter. A client connecting to the service must specify this password. Similarly, when modifying an existing service that is password protected, the client must specify the password for that service. (During a password-protected session, a valid password must be present.) This password can control read, write, or read-write access.
- The new service password parameter (NEW_SRVC_PASSW), which is used to change the service password. The password can be changed only if the current password is valid. The service password cannot be changed when connecting to an existing password-protected service.

A password-encoding parameter (PASSW_ENCODE) specifies how a service password is encoded in LASTport/Disk messages. Digital plans to allow encoding of distributed authentication and authorization policies in these messages using new Parameter Codes.

While the LASTport/Disk architecture does not specify password and data encryption, the architecture can accommodate these functions.

1.4.4 Solicit Response Policies

A client can construct Solicit Request messages in a variety of combinations to meet the needs of a user request. The LASTport/Disk architecture defines the policies that servers implement when they respond to these requests.

Solicit Request messages can be directed to a single node using a physical address or to all nodes using a multicast address. The Solicit Request message can include a Server Name and a Service Name. The message can also contain other information, such as device name or device type, that further qualifies the responses that servers provide.

When servers respond to a Solicit Request message, they always provide information about a single Service Instance in the Solicit Response message. If the server needs to return information for more than one service, it does so by sending multiple responses. Each response contains information about a single Service Instance.

Servers must **dally** responses to Solicit Request messages. When client systems construct Solicit Request messages, the format of the request can cause many or all servers on the extended LAN to respond with information. If all servers responded without delay, the client network adapter could be flooded with these responses, and some information could be lost. To prevent such adapter flooding, the client specifies the value of the RESPONSE_TIMER field in the Solicit Request message that it constructs. Servers must dally responses by waiting a random amount of time, from 0 seconds up to the client's RESPONSE_TIMER value, before transmitting the Solicit Response message. If the client specifies a timer value of 0, no dally is required, and servers respond to the Solicit Request message immediately.

Table 1–3 defines the policies that servers can implement in responding to Solicit Request messages. A null name means that no name is provided in the Solicit Request message.

Table 1–3 Response Policies for Solicit Request Messages

Destination Service Name	Destination Node Name	Solicit Request Message Multicasted	Solicit Request Message Physically Addressed
Null	Null	Any node can respond with node information and information for all services in requested Name Space.	Addressed node may respond with node information and information for all services in requested Name Space.
Null	Not null	Named node must respond with node information and information for all services in requested Name Space.	If node name is correct, node must respond with node information and information for all services in requested Name Space.
Not null	Null	If node offers service, it must respond with node information and service information; otherwise no response is sent.	If node offers service, it must respond with node information and service information; otherwise return error message “node does not offer service.”
Not null	Not null	If node offers service and node name is correct, node must respond with node and service information; otherwise, return error message “node does not offer service.”	If node offers service and node name is correct, node must respond with node and service information; otherwise, return error message “node does not offer service.”

The LASTport/Disk architecture defines a flexible set of solicitation message formats that meet a variety of possible user requests. Specific service, node, or device information can be directed to a single server. Table 1–4 lists examples of possible user requests and shows how a client would format a Solicit Request message to meet those requests.

Table 1–4 Examples of Solicit Requests and Client-Supplied Values

User Request	Client-Supplied Values

Show service CONDIST	Service-name = CONDIST
	Node-name = Null
	Device-name = Null
	Device-type = Null
	Address = Multicast

Show all services offered by server FRED	Service-name = Null
	Node-name = FRED
	Device-name = Null
	Device-type = Null
	Address = Physical or multicast

Show service ONLINE_DOC only on RRD42 compact disk drives	Service-name = ONLINE_DOC
	Node-name = Null
	Device-name = Null
	Device-type = RRD42
	Address = Multicast

Show all RRD40 compact disk drives on the network	Service-name = Null
	Node-name = Null
	Device-name = Null
	Device-type = RRD40
	Address = Multicast

Show all services offered by node FRED on device PORT_1	Service-name = Null
	Node-name = FRED
	Device-name = PORT_1
	Device-type = Null
	Address = Physical or multicast

Show whether node FRED offers service CONDIST on device PORT_1	Service-name = CONDIST
	Node-name = FRED
	Device-name = PORT_1
	Device-type = Null
	Address = Physical or multicast

1.5 Revision Control

The LASTport/Disk protocol is under revision control. When any LASTport/Disk message is received, it contains an explicit current major version identification, or it is bound to a session known to correspond to a current major protocol revision.

In LASTport/Disk messages, the current major version field (CUR_PRTCL_VER) specifies that the message received or transmitted is encoded under that version. Two other fields, low major version (LOW_PRTCL_VER) and high major version (HIGH_PRTCL_VER), indicate the oldest and newest major versions of the protocol that the transmitter is capable of encoding.

In general, implementations solicit using the high major version set equal to the current major version. New server implementations thus see old client implementations using an old protocol major version, and “speak down” to them. Old server implementations successfully ignore new client implementations, unless the new client has a mechanism to solicit using older major versions. A LASTport/Disk implementation must discard, without response, any message with the current version and ECO outside of its own high and low range.

A server can respond to a Solicit Request message or a Connect Request message by specifying the highest common protocol version and ECO between the client and the server. Once the server has selected a common version and ECO level in its Connect Response message, all association-related messages exception solicitation messages must use that version and ECO level for the connection.

For each minor version ECO within a major version, backward-compatible changes are made that do not affect interoperability with a previous version. When a major version is first implemented, the value of the CUR_PRTCL_ECO field must be 0. In successive releases, this value might be changed to 1, and then to 2, and so on. Old implementations ignore the value, and new implementations offer enhanced capabilities transparently to the older implementations.

LASTport/Disk Version 3.1 clients can implement a subset of the protocol. For example, a client might not require the dynamic storage creation capability. In this case, the client could omit the code to construct the variant of the Connect Request message that instructs the server to create storage. Because the client is the master in the protocol, the client defines the subset of features that it requires.

LASTport/Disk Version 3.1 servers must process all request message variants. However, some servers may not be able to satisfy all Version 3.1 requests. For example, a client might solicit for additional storage, and a particular Version 3.1 server might not have read/write media.

3 LASTport/Disk Messages

This chapter describes the following LASTport/Disk Version 3.1 messages:

- Solicit Request
- Solicit Response
- Solicit Summary Request
- Solicit Summary Response
- Connect Request
- Connect Response
- Parameter Request
- Parameter Response
- Data Request
- Data Response
- Disconnect Request
- Disconnect Response

Figures 3–1 through 3–10 illustrate message formats. Accompanying tables describe message fields in detail.

Because the LASTport/Disk wire protocol is meant to operate independently of lower-level transport protocols, the message formats are represented as procedure calls whose fields are bit streams, shown right to left. The right-most bit is the first bit transmitted on the wire (little-endian). Each field is an integer multiple of eight bits. The equals sign (=) is used in Figures 3–1 through 3–10 to indicate fields of varying or indeterminate length.

3.1 Version 3.1 Solicit Request Message

The Solicit Request message is used to solicit a particular service or set of services from servers offering LASTport/Disk virtual disks on the LAN. The message is supplied as a parameter to the LAST SolClientEvent process with the *SolReqMsgSend* event. The LASTport Solicitation layer transmits the message in accordance with the LASTport Name Service model.

Figure 3–1 shows the message format for a Solicit Request message. Table 3–1 describes message fields.

Figure 3–1 Version 3.1 Solicit Request Message Format

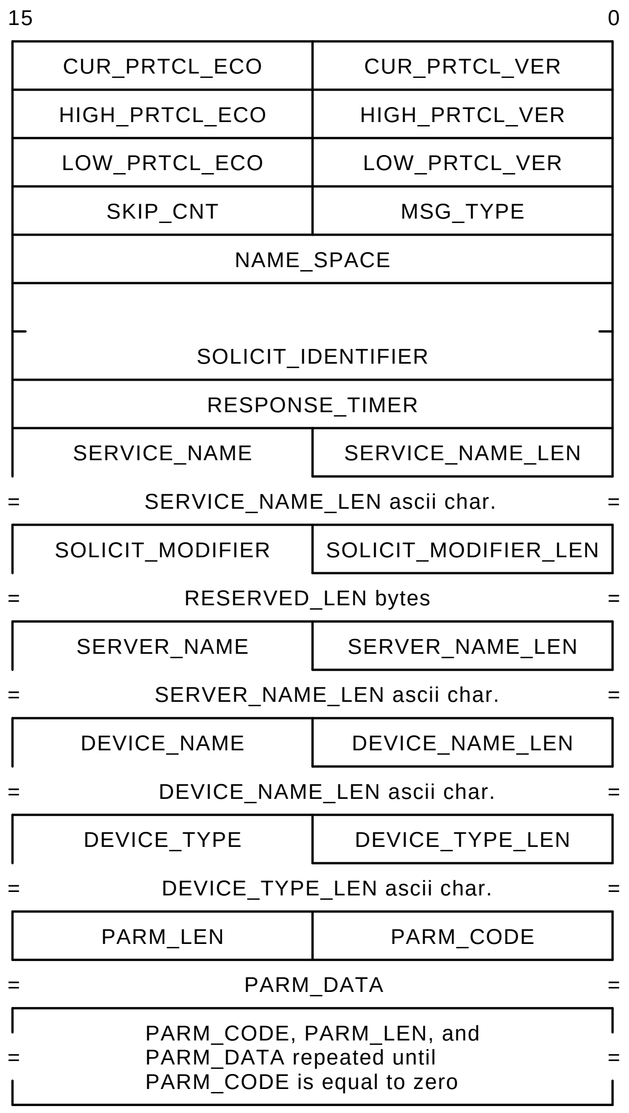

Table 3–1 Version 3.1 Solicit Request Message Fields

Name	Length	Description
CUR_PRTCL_VER	1 byte	Protocol version of this message (current version is 3.1).
CUR_PRTCL_ECO	1 byte	ECO level of CUR_PRTCL_VER for this message (current ECO is 0).
HIGH_PRTCL_VER	1 byte	Highest protocol version supported by node.
HIGH_PRTCL_ECO	1 byte	Highest ECO level supported by node.
LOW_PRTCL_VER	1 byte	Lowest protocol version supported by node.
LOW_PRTCL_ECO	1 byte	Lowest ECO level supported by node.
MSG_TYPE	1 byte	The value 8 (Solicit Request message).
SKIP_CNT	1 byte	The number of times that the client has seen a particular service. A client Solicit Request message can require more than one response from a server. This field indicates to the server how many service instances to “skip over” before returning the next response. Normally this field is set to zero on an initial solicitation and contains the SERVICE_INST_CNT field from a server Solicit Response message when a server indicates that there is more information available.
NAME_SPACE	2 bytes unsigned	The Name Space being solicited. Name Spaces are described in Table 1–1.
SOLICIT_IDENTIFIER	4 bytes unsigned	Identifier produced by the soliciting node that uniquely identifies the Solicit Request message. The soliciting node uses this identifier to correlate corresponding Solicit Response message(s).
RESPONSE_TIMER	2 bytes unsigned	Response timer (seconds) that starts when the Solicit Request message is sent. The soliciting node uses this timer to time out when waiting for Solicit Response messages. Servers should use this value as a maximum for the local response timer.
SERVICE_NAME_LEN	1 byte	Length of the next field. A byte containing the length of the SERVICE_NAME field in bytes. A value of 0 indicates that a particular service is not being solicited.
SERVICE_NAME	SERVICE_NAME_LEN bytes	Service name. An array of ASCII characters describing the name of a particular service being solicited. A server responds to a Solicit Request message only if it offers the service name specified. Service name characters are constrained as described in Section 1.3.5.

(continued on next page)

Table 3–1 (Cont.) Version 3.1 Solicit Request Message Fields

Name	Length	Description
SOLICIT_MODIFIER_LEN	1 byte	Length of the next field. A byte containing the length of the SOLICIT_MODIFIER field in bytes. A value of 0 indicates that no SOLICIT_MODIFIER is provided.
SOLICIT_MODIFIER	SOLICIT_MODIFIER_LEN bytes	Solicit modifier. A bit mask that enables the client to reduce the number of solicit responses by limiting the scope of the solicitation. The meaning of the bits (when set) is: • bit 0 = FREE_STORAGE. When set, indicates that the client is soliciting for the amount of free disk space that is available for creating services. • bit 1 = PERM_FREE_STORAGE. When set, indicates that the client is soliciting for services that support permanent free storage. Permanent free storage means that data created in free disk space is valid across server reboots. • bits 2 to n = TAZIOR.
SERVER_NAME_LEN	1 byte	Length of the next field. A byte containing the length of the SERVER_NAME field in bytes. A value of 0 indicates that no particular server is being solicited.
SERVER_NAME	SERVER_NAME_LEN bytes	Server name. An array of ASCII characters describing the name of a particular server being solicited. A server responds to a Solicit Request message only if its server name matches the name supplied in this field. Server name characters are constrained as described in Section 1.3.5.
DEVICE_NAME_LEN	1 byte	Length of the next field. A byte containing the length of the DEVICE_NAME field in bytes. A value of 0 indicates that a particular device is not being solicited.
DEVICE_NAME	DEVICE_NAME_LEN bytes	Device name. An array of ASCII characters describing a particular device on a server. Servers respond with information for the named device only if it exists. Device names are constrained as described in Section 1.3.5.

(continued on next page)

Table 3–1 (Cont.) Version 3.1 Solicit Request Message Fields

Name	Length	Description
DEVICE_TYPE_LEN	1 byte	Length of the next field. A byte containing the length of the DEVICE_TYPE field in bytes. A value of 0 indicates that no special device type is being requested.
DEVICE_TYPE	DEVICE_TYPE_LEN bytes	Device type. An array of ASCII characters describing a particular device type being requested. Servers respond only if they offer the service on the specified device type. Device type strings are implementation specific, and are constrained as described in Section 1.3.5.
PARM_CODE	1 byte	A parameter code. Parameter codes are defined in Table 3–2.
PARM_LEN	1 byte	The length of the following field in bytes. Note that the PARM_LEN field does not include the code and length bytes in the count.
PARM_DATA	PARM_LEN bytes	Parameter data.

Table 3–2 Version 3.1 Solicit Request Message Parameter Codes

Code	Length	Name	Description
0			Denotes the end of a parameter list.
1	4 bytes	SOL_FREE_BLOCKS	An integer specifying a free block threshold for which a server should respond. Clients that require a minimum amount of free blocks of storage for creating services use this to limit the number of Solicit Responses. Servers that do not have more than SOL_FREE_BLOCKS of available storage left do not respond.
2	1 byte	SOL_DEVICE_CLASS	A code indicating a particular type of device that is being solicited. The following codes are defined: 0 = Fixed Disk 1 = Tape 2 = Printer 3 = Reserved 4 = WORM 5 = CDROM 6 = CDROM jukebox 7 = Magneto-optical disk 8 = Magneto-optical jukebox 9 = Removable 3.5-inch floppy 10 = Removable 5.25-inch floppy 11 = Removable disk 12–255 = Reserved
3–127			Reserved for Digital use.
128–255			Reserved licensee, assigned by Digital.

3.2 Version 3.1 Solicit Response Message

The Solicit Response message is used to answer a request for information about a particular service or set of services. The message is supplied as a parameter to the LASTport SolServerEvent process with the *SolRspMsgSend* event.

Figure 3–2 shows the message format for a Solicit Response message. Table 3–3 describes message fields.

Figure 3–2 Version 3.1 Solicit Response Message Format

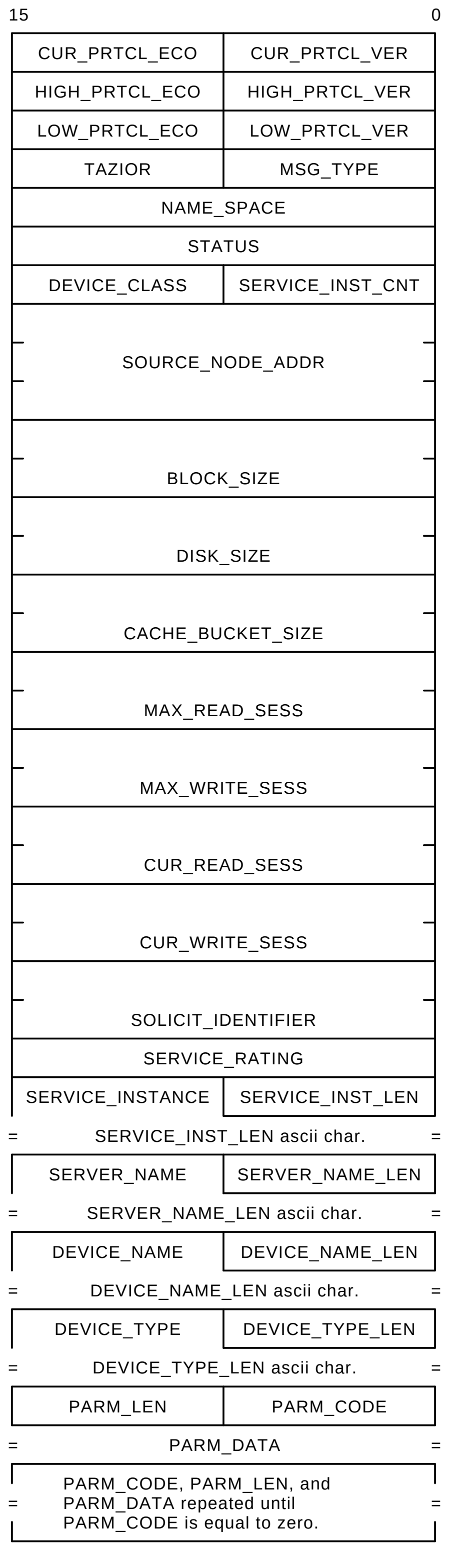

Table 3–3 Version 3.1 Solicit Response Message Fields

Name	Length	Description
CUR_PRTCL_VER	1 byte	Protocol version of this message (current version is 3.1). A responding node is allowed to respond using the highest version number supported by both the soliciting and the responding node.
CUR_PRTCL_ECO	1 byte	ECO level of CUR_PRTCL_VER for this message (current ECO is 0). A responding node is allowed to respond using the highest ECO level supported by both the soliciting and the responding node.
HIGH_PRTCL_VER	1 byte	Highest protocol version supported by node.
HIGH_PRTCL_ECO	1 byte	Highest ECO level supported by node.
LOW_PRTCL_VER	1 byte	Lowest protocol version supported by node.
LOW_PRTCL_ECO	1 byte	Lowest ECO level supported by node.
MSG_TYPE	1 byte	The value 9 (Solicit Response message).
TAZIOR	1 byte	Transmit as zero and ignore on receipt.
NAME_SPACE	2 bytes unsigned	The Name Space of the described service instance descriptor. Name Spaces are described in Table 1–1.
STATUS	2 bytes signed	Response status. A positive value indicates success status. Negative values indicate an error. Status code values are defined separately for each service class. Status code values are as follows: 1 = Success 2 = Success, more information available 3 = Success, end of list 0, –1, –2 = Reserved –3 = No such object
SERVICE_INST_CNT	1 byte	The instance number of the service information being returned. A client solicit can require more than one response from a server. This field indicates the number of times a server has responded, including the current response. A client uses this value in the SKIP_CNT field of a Solicit Request message when more information is being requested.

(continued on next page)

Table 3–3 (Cont.) Version 3.1 Solicit Response Message Fields

Name	Length	Description
DEVICE_CLASS	1 byte	A code indicating the type of device on which this service instance is being offered. The following codes are defined: 0 = Fixed Disk 1 = Tape 2 = Printer 3 = Reserved 4 = WORM 5 = CDROM 6 = CDROM jukebox 7 = Magneto-optical disk 8 = Magneto-optical jukebox 9 = Removable 3.5-inch floppy 10 = Removable 5.25-inch floppy 11 = Removable disk 12–255 = Reserved
SOURCE_NODE_ADDR	6 byte field	Transmitting node must fill this field with a value equal to the datalink source address of the named server node. Receiving node must reference this field and ignore the datalink source address of message.
BLOCK_SIZE	4 bytes unsigned	An integer specifying the number of bytes per disk block. However, only the value 512 is currently supported.
DISK_SIZE	4 bytes unsigned	An integer specifying the number of blocks of storage for a virtual disk. A value of 0 is illegal.
CACHE_BUCKET_SIZE	4 bytes unsigned	An integer specifying the number of bytes that the server uses to fill each “bucket” in its cache. Clients can use this value to determine read patterns which would cause “cache-striping” across multiple servers when run-time load balancing is used. A value of 0 is illegal.
MAX_READ_SESS	4 bytes unsigned	An integer specifying the maximum number of simultaneous readers that can be opened to this service.
MAX_WRITE_SESS	4 bytes unsigned	An integer specifying the maximum number of simultaneous writers that can be opened to this service.
CUR_READ_SESS	4 bytes unsigned	An integer specifying the current number of read sessions opened to this service.
CUR_WRITE_SESS	4 bytes unsigned	An integer specifying the current number of write sessions opened to this service.
SOLICIT_IDENTIFIER	4 bytes unsigned	Identifier produced by the soliciting node that uniquely identifies the solicitor’s Solicit Request message. This value is equal to the SOLICIT_IDENTIFIER field of the received Solicit Request message that is being responded to by this Solicit Response message.
SERVICE_RATING	2 bytes unsigned	A rating of the associated service instance. ratings change dynamically based on server load.

(continued on next page)

Table 3–3 (Cont.) Version 3.1 Solicit Response Message Fields

Name	Length	Description
SERVICE_INST_LEN	1 byte	Length of next field. A byte containing the length of the SERVICE_INSTANCE field in bytes. A value of 0 is illegal.
SERVICE_INSTANCE	SERVICE_INST_LEN bytes	Service Instance. An array of ASCII characters describing an instance of the requested service name. Clients use this field in Connect Request messages to establish a connection to a server. These characters are constrained as described in Section 1.3.5.
SERVER_NAME_LEN	1 byte	Length of the next field. A byte containing the length of the SERVER_NAME field in bytes. A value of 0 is illegal.
SERVER_NAME	SERVER_NAME_LEN bytes	Server name. An array of ASCII characters describing the name of a particular server being solicited. These characters are constrained as described in Section 1.3.5.
DEVICE_NAME_LEN	1 byte	Length of the next field. A byte containing the length of the DEVICE_NAME field in bytes. A value of 0 indicates that information is not being returned for a particular device.
DEVICE_NAME	DEVICE_NAME_LEN bytes	Device name. An array of ASCII characters describing the name assigned to a particular port by a server. These characters are constrained as described in Section 1.3.5.
DEVICE_TYPE_LEN	1 byte	Length of the next field. A byte containing the length of the DEVICE_TYPE field in bytes. A value of 0 indicates that no device type description is provided.
DEVICE_TYPE	DEVICE_TYPE_LEN bytes	Device type. An array of ASCII characters describing a particular device type being requested. Device type strings are implementation specific, and are constrained as described in Section 1.3.5. Table [TBS] lists globally defined DEVICE_TYPE names.
PARM_CODE	1 byte	A parameter code. Parameter codes are defined in Table 3–4.
PARM_LEN	1 byte	The length of the following field in bytes. Note that the parameter length no longer includes the code and length bytes in the count.
PARM_DATA	PARM_LEN bytes	Parameter data.

Table 3–4 Version 3.1 Solicit Response Message Parameter Codes

Code	Size (bytes)	Name	Description
0			Denotes the end of a parameter list.
1	2	RSP_ACCESS_MODE	Bit mask indicating the access mode for the connection. Servers fill in this field based on the requested access mode from the client or can indicate a default access mode selected by the server if no particular access mode was requested by the client. Table 3–12 defines the meaning of the bits.
2	4	FREE_BLOCKS	An integer specifying the number of free blocks of storage available for creating services. This is the total number of free blocks that the server has allocated for this device.
3	4	INSTANCE_QUOTA	An integer specifying the maximum number blocks that an individual service instance can use. This is a server specified parameter that applies to all service instances.
4–127			Reserved for Digital use.
128–255			Reserved for Name Space–specific usage.

3.3 Version 3.1 Solicit Summary Request Message

The Solicit Summary Request message is used to solicit a summary of services from servers offering LASTport/Disk virtual disks on the LAN. The message is supplied as a parameter to the LASTport SolClientEvent process with the *SolReqMsgSend* event. The LASTport Solicitation layer transmits the message in accordance with the LASTport Name Service model.

Figure 3–3 shows the message format for a Solicit Summary Request message. Table 3–5 describes message fields.

Figure 3–3 Version 3.1 Solicit Summary Request Message Format

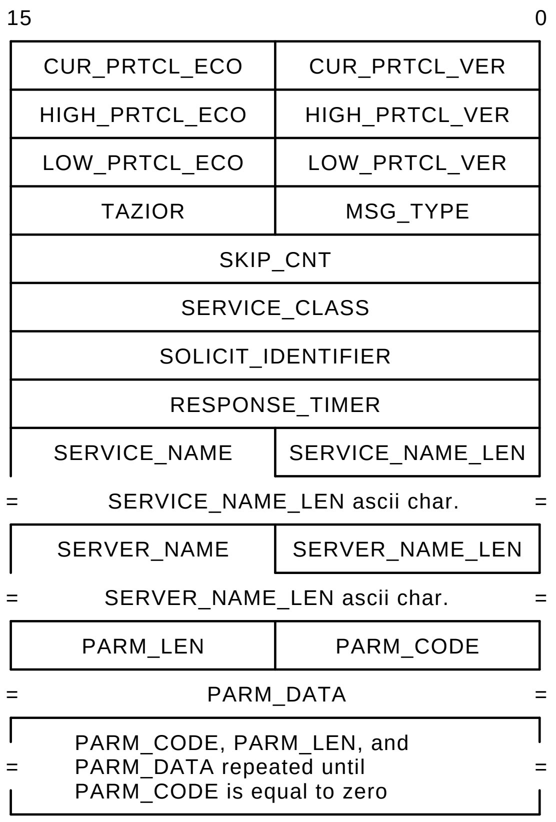

Table 3–5 Version 3.1 Solicit Summary Request Message Fields

Name	Length	Description
CUR_PRTCL_VER	1 byte	Protocol version of this message (current version is 3.1).
CUR_PRTCL_ECO	1 byte	ECO level of CUR_PRTCL_VER for this message (current ECO is 0).
HIGH_PRTCL_VER	1 byte	Highest protocol version supported by node.
HIGH_PRTCL_ECO	1 byte	Highest ECO level supported by node.
LOW_PRTCL_VER	1 byte	Lowest protocol version supported by node.
LOW_PRTCL_ECO	1 byte	Lowest ECO level supported by node.
MSG_TYPE	1 byte	The value 14 (Solicit Summary Request message).
TAZIOR	1 byte	Transmit as zero and ignore on receipt.
SKIP_CNT	2 bytes	The number of times that the client has seen a particular service. A client solicit can require more than one response from a server. This field indicates to the server how many service instances to “skip over” before returning the next response. Normally this field is set to zero on an initial solicitation, and contains the SERVICE_INST_CNT field from a server Solicit Summary Response message when a server indicates that there is more information available.
SERVICE_CLASS	2 bytes unsigned	The Service Class (Name Space) of service being solicited.
SOLICIT_IDENTIFIER	4 bytes unsigned	Identifier produced by the soliciting node that uniquely identifies the Solicit Request message. The soliciting node uses this identifier to correlate corresponding Solicit Response message(s).
RESPONSE_TIMER	2 bytes unsigned	Response timer (seconds) that starts when the Solicit Request message is sent. The soliciting node uses this timer to time out when waiting for Solicit Response messages. Servers use this value as a maximum for the local response timer.
SERVICE_NAME_LEN	1 byte	Length of the next field. A byte containing the length of the SERVICE_NAME field in bytes. A value of 0 indicates that a particular service is not being solicited.
SERVICE_NAME	SERVICE_NAME_LEN bytes	Service name. An array of ASCII characters describing the name of a particular service being solicited. A server responds to a Solicit Request message only if it offers the service name specified. Service name characters are constrained as described in Section 1.3.5.
SERVER_NAME_LEN	1 byte	Length of the next field. A byte containing the length of the SERVER_NAME field in bytes. A value of 0 indicates that no particular server is being solicited.

(continued on next page)

Table 3–5 (Cont.) Version 3.1 Solicit Summary Request Message Fields

Name	Length	Description
SERVER_NAME	SERVER_NAME_LEN bytes	Server name. An array of ASCII characters describing the name of a particular server being solicited. A server responds to a Solicit Request message only if its server name matches the name supplied in this field. Server name characters are constrained as described in Section 1.3.5.
PARM_CODE	1 byte	A parameter code. Parameter codes are defined in Table 3–6.
PARM_LEN	1 byte	The length of the following field in bytes. Note that the parameter length no longer includes the code and length bytes in the count.
PARM_DATA	PARM_LEN bytes	Parameter data.

Table 3–6 Version 3.1 Solicit Summary Request Message Parameter Codes

Code	Description
0	Denotes the end of a parameter list.
1–127	Reserved for Digital use.
128–255	Reserved for Name Space–specific usage.

3.4 Version 3.1 Solicit Summary Response Message

The Solicit Summary Response message is used to answer a Solicit Summary Request message and is supplied as a parameter to the LASTport SolServerEvent process with the *SolRspMsgSend* event.

Figure 3–4 shows the message format for a Solicit Summary Response message. Table 3–7 describes message fields.

Figure 3–4 Version 3.1 Solicit Summary Response Message Format

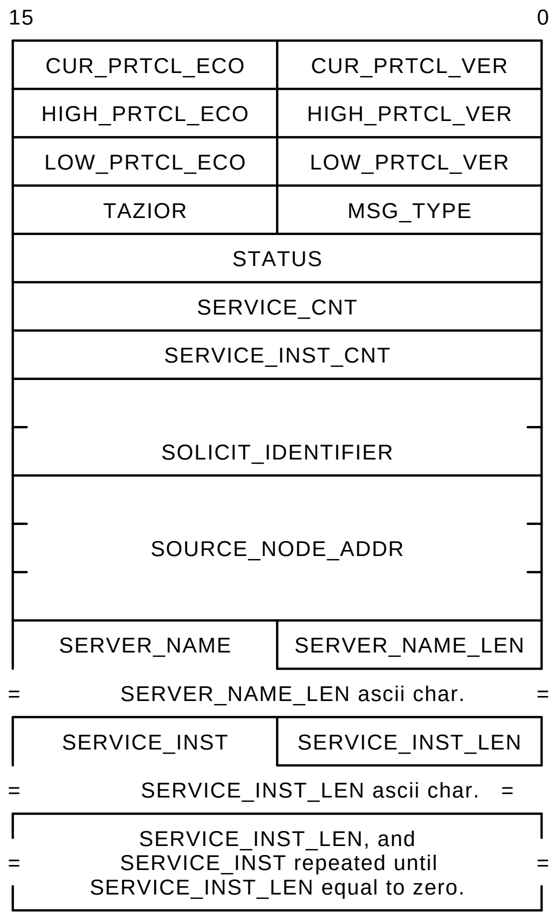

Table 3–7 Version 3.1 Solicit Summary Response Message Fields

Name	Length	Description
CUR_PRTCL_VER	1 byte	Protocol version of this message (current version is 3.1). A responding node is allowed to respond using the highest version number supported by both the soliciting and the responding node.
CUR_PRTCL_ECO	1 byte	ECO level of CUR_PRTCL_VER for this message (current ECO is 0). A responding node is allowed to respond using the highest ECO level supported by both the soliciting and the responding node.
HIGH_PRTCL_VER	1 byte	Highest protocol version supported by node.
HIGH_PRTCL_ECO	1 byte	Highest ECO level supported by node.
LOW_PRTCL_VER	1 byte	Lowest protocol version supported by node.
LOW_PRTCL_ECO	1 byte	Lowest ECO level supported by node.
MSG_TYPE	1 byte	The value 15 (Solicit Summary Response message).
TAZIOR	1 byte	Transmit as zero and ignore on receipt.
STATUS	2 bytes signed	Response status. A positive value indicates success status. Negative values indicate an error. Status code values are defined separately for each service class. Status code values are as follows: 1 = Success 2 = Success, more information available. 3 = Success, end of list. 0, –1, –2 = Reserved –3 = No such object
SERVICE_CNT	2 bytes	The number of services included in this message.
SERVICE_INST_CNT	1 byte	The instance number of the service information being returned. A client solicit can require more than one response from a server. This field indicates the number of times a server has responded, including the current response. A client uses this value in the SKIP_CNT field of a Solicit Request message when more information is being requested.
SOLICIT_IDENTIFIER	4 bytes unsigned	Identifier produced by the soliciting node that uniquely identifies the solicitor's Solicit Summary Request message. This value is equal to the SOLICIT_IDENTIFIER field of the received Solicit Summary Request message that is being responded to by this Solicit Summary Response message.
SOURCE_NODE_ADDR	6 bytes	Transmitting node must fill this field with a value equal to the datalink source address of the named server node. Receiving node must reference this field and ignore the datalink source address of message.

(continued on next page)

Table 3–7 (Cont.) Version 3.1 Solicit Summary Response Message Fields

Name	Length	Description
SERVER_NAME_LEN	1 byte	Length of the next field. A byte containing the length of the SERVER_NAME field in bytes. A value of 0 is illegal.
SERVER_NAME	SERVER_NAME_LEN bytes	Server name. An array of ASCII characters describing the name of a particular server being solicited. These characters are constrained as described in Section 1.3.5.
SERVICE_INST_LEN	1 byte	Length of next field. A byte containing the length of the SERVICE_INSTANCE field in bytes. A value of 0 is illegal.
SERVICE_INSTANCE	SERVICE_INST_LEN bytes	Service Instance. An array of ASCII characters describing an instance of the requested service name. Clients use this field in Connect Request messages to establish a connection to a server. These characters are constrained as described in Section 1.3.5.

3.5 Version 3.1 Connect Request Message

The Connect Request message is used to initiate access to a virtual disk. The message is supplied as a parameter to the LASTport AsnClientRcv process with the *AsnConnect* event. The message is also used as data in the LASTport Connect Request message.

Figure 3–5 shows the message format for a Connect Request message. Table 3–8 describes message fields. Table 3–9 defines the Connect Request message ACCESS_MODE_MASK bits; Table 3–10 defines the message parameter codes.

Figure 3–5 Version 3.1 Connect Request Message Format

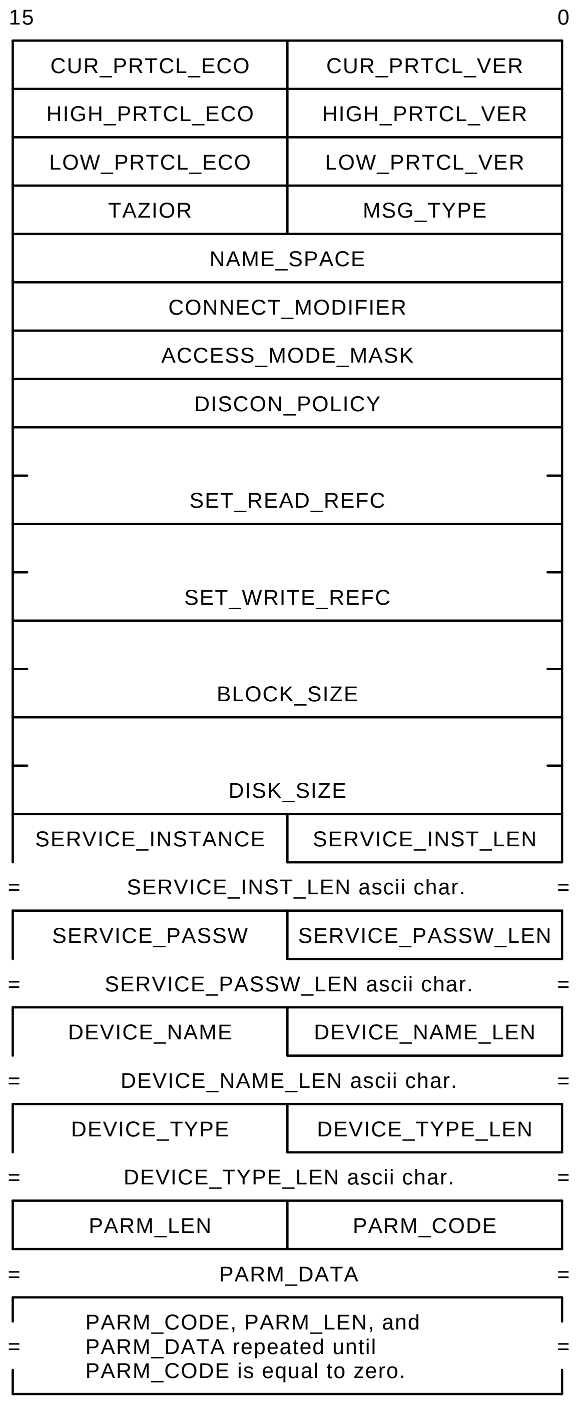

Table 3–8 Version 3.1 Connect Request Message Fields

Name	Length	Description
CUR_PRTCL_VER	1 byte	Protocol version of this message (current version is 3.1).
CUR_PRTCL_ECO	1 byte	ECO level of CUR_PRTCL_VER for this message (current ECO is 0).
HIGH_PRTCL_VER	1 byte	Highest protocol version supported by node.
HIGH_PRTCL_ECO	1 byte	Highest ECO level supported by node.
LOW_PRTCL_VER	1 byte	Lowest protocol version supported by node.
LOW_PRTCL_ECO	1 byte	Lowest ECO level supported by node.
MSG_TYPE	1 byte	The value 10 (Connect Request message).
TAZIOR	1 byte	Transmit as zero and ignore on receipt.
NAME_SPACE	2 bytes unsigned	The Name Space of the service instance being requested. Name Spaces are described in Table 1–1.
CONNECT_MODIFIER	2 byte bitmask	This field indicates the type of connection being requested. If no bits are set then the connection request is for an existing service (that was previously configured). The meaning of the bits (when set) is as follows: • bit 0 = CREATE/CONNECT_SERVICE. When set, allows the remote creation of a virtual disk (at connection time). • bit 0 = CONNECT/PRESERVE_SERVICE. When set, preserves the service until an update request is issued (see Table 3–19). • bits 2–15 = TAZIOR.
ACCESS_MODE_MASK	2 byte bitmask	Bit mask indicating which attributes should be modified via this access. A value of 0, in most cases, indicates that the server should use its default policy or value. The bits “enabling set access” are only meaningful if a service is being created (that is, the Create/Connect Service bit is set in the CONNECT_MODIFIER). Table 3–9 defines the bits.

(continued on next page)

Table 3–8 (Cont.) Version 3.1 Connect Request Message Fields

Name	Length	Description
DISCON_POLICY	2 byte bitmask	Bit mask specifying the requested disconnect policy that the server should provide. This field is meaningful only if ENABLE_DISCON_POLICY bit is set in the ACCESS_MODE_MASK. If bit is not set, this field is TAZIOR. The meaning of the bits (when set) is as follows: • bit 0 = TIMEOUT. When set, indicates that the service is deleted when a time interval (specified in CONN_TIMEOUT) has expired with no further connections to the service. (If CONN_TIMEOUT is set to zero, the service is deleted when disconnected.) When clear, indicates that the service is never deleted on disconnect (that is, the CONN_TIMEOUT is infinite). • bit 1 = ZERO_DISCONN. When set, indicates that the virtual disk associated with this service is initialized on disconnect. All the data on the virtual disk is cleared. • bits 2-15—TAZIOR.
SET_READ_REFC	4 bytes unsigned	The read reference count is an integer specifying the maximum number of simultaneous readers allowed for a particular service. This field has meaning only when the ENABLE_SET_READ_REFC bit is set in the ACCESS_MODE_MASK. If bit is not set, this field is TAZIOR. A value of 1 indicates exclusive read access.
SET_WRITE_REFC	4 bytes unsigned	The write reference count is an integer specifying the maximum number of simultaneous writers allowed for a particular service. This field has meaning only when the ENABLE_SET_WRITE_REFC bit is set in the ACCESS_MODE_MASK. If the bit is not set, this field is TAZ. A value of 1 indicates exclusive write access.
BLOCK_SIZE	4 bytes unsigned	An integer specifying the number of bytes per disk block. If no integer is specified, a default block size for the disk is used. This field has meaning only when the ENABLE_BLOCK_SIZE bit is set in the ACCESS_MODE_MASK. If the bit is clear, this field must be zero. Currently, only a size of 512 is supported.
DISK_SIZE	4 bytes unsigned	An integer specifying the number of blocks of storage for a virtual disk. This field only has meaning when the ENABLE_DISK_SIZE bit is set in the ACCESS_MODE_MASK. If not then this field is TAZIOR.
SERVICE_INST_LEN	1 byte	Length of next field. A byte containing the length of the SERVICE_INSTANCE field in bytes. A value of 0 is illegal.

(continued on next page)

Table 3–8 (Cont.) Version 3.1 Connect Request Message Fields

Name	Length	Description
SERVICE_INSTANCE	SERVICE_INST_LEN bytes	Service Instance. An array of ASCII characters describing the instance of the requested service name. Clients fill in this field from the received field in the Solicit Response message (or other naming service). These characters are constrained as described in Section 1.3.5.
SERVICE_PASSW_LEN	1 byte	Length of the next field. A byte containing the length of the SERVICE_PASSW field in bytes. A value of 0 indicates that no password is provided.
SERVICE_PASSW	SERVICE_PASSW_LEN bytes	Service password. An array of ASCII characters used to pass the service password to a server. A server rejects a connection to a password protected service if the client does not supply the correct password.
DEVICE_NAME_LEN	1 byte	Length of the next field. A byte containing the length of the DEVICE_NAME field in bytes. A value of 0 indicates that a particular device is not being requested.
DEVICE_NAME	DEVICE_NAME_LEN bytes	Device name. An array of ASCII characters describing a particular device on a server. Servers respond with information only for the named device, if it exists. Device names are constrained as described in Section 1.3.5.
DEVICE_TYPE_LEN	1 byte	Length of the next field. A byte containing the length of the DEVICE_TYPE field in bytes. A value of 0 indicates that no special device type is being requested.
DEVICE_TYPE	DEVICE_TYPE_LEN bytes	Device type. An array of ASCII characters describing a particular device type being requested. Servers complete the connect request only if they offer the service on the specified device type. Device type strings are implementation specific, and are constrained as described in Section 1.3.5. Table [TBS] lists globally defined DEVICE_TYPE names.
PARM_CODE	1 byte	A parameter code. Parameter codes are defined in Table 3–10.
PARM_LEN	1 byte	The length of the following field in bytes. Note that the parameter length no longer includes the code and length bytes in the count.
PARM_DATA	PARM_LEN bytes	Parameter data.

Table 3–9 Version 3.1 Connect Request Message ACCESS_MODE_MASK Bits

Bit	Name	Description
0	REQUEST_READ_ACCESS	When set, indicates that the client is requesting read access for this service. This function can be requested when creating a service or when connecting to an existing service.
1	ENABLE_SET_READ_REFC	When set, indicates that the maximum number of client connections allowed for read access is specified in SET_READ_REFC. This function can only be used when creating a service.
2	REQUEST_WRITE_ACCESS	When set, indicates that the client is requesting write access for this service. This function can be requested when creating a service or when connecting to an existing service.
3	ENABLE_SET_WRITE_REFC	When set, indicates that the maximum number of client connections allowed for write access is specified in SET_WRITE_REFC.This function can only be used when creating a service.
4	ENABLE_BLOCK_SIZE	When set, indicates that the number of bytes per block is specified in BLOCK_SIZE. This function can only be used when creating a service.
5	ENABLE_DISK_SIZE	When set, indicates that the number of blocks per virtual disk is specified in DISK_SIZE. This function can only be used when creating a service.
6	ENABLE_DISCON_POLICY	When set, indicates that the disconnect policy that the server provides is specified in DISCON_POLICY. This function can be used only when creating a service.
7	DISABLE_PASSW_READ	When set, indicates that the password is *not* required for read access. This function can only be used when creating a service.
8	DISABLE_PASSW_WRITE	When set, indicates that the password is *not* required for write access. This function can be used only when creating a service.
9	RATING	When clear, indicates that the rating for this service is calculated dynamically. When set, indicates that the rating for this service is static. A rating from 0 to 65535 is used for load balancing across redundant services. If the parameter STATIC_RATING is not provided, a default value is used. This function can be used only when creating a service.
10	RESERVED	Reserved.
11	PERM_FREE_STORAGE	When set, this indicates that the client is requesting services that support permanent free storage. Permanent free storage means that data created in free disk space is valid across re-boots of the server.
12–15		TAZIOR.

Table 3–10 Version 3.1 Connect Request Message Parameter Codes

Code	Length	Name	Description
0			Denotes the end of a parameter list.
1	4 bytes	CONN_TIMEOUT	Indicates the amount of time, in seconds, that this service exists with no connections. If no connections are made to this service in CONN_ TIMEOUT number of minutes, the service is deleted. Servers time out connections to reclaim shared resources from a node that can have hung or crashed. This parameter is meaningful only if the disconnect policy TIMEOUT is selected (in the DISCON_POLICY bitmask). If the disconnect policy TIMEOUT is selected and this parameter is not specified, a default value is used.
2	n bytes	NEW_SERVICE_PASSW	An array of ASCII characters used to modify the service password. The SRVC_PASSW field in this message MUST be valid (or null) for the server password to be modified.
3	2 bytes	STATIC_RATING	Used to set the value used for static rating. A rating from 0 to 65535 is used for load balancing across redundant services. This function can only be used when creating a service.
4	2 bytes	CACHE_READ_POLICY	Bit mask indicating type of cache read policy that the server uses to fill each "bucket" in its cache. The client can specify any combination of read-ahead and read-behind. The meaning of the bits (when set) is as follows: • bit 0 = READ_AHEAD. The server fills the cache by reading ahead. • bit 1 = READ_BEHIND. The server fills the cache by reading behind.
5–127			Reserved for Digital use.
128–255			Reserved for Name Space–specific usage.

3.6 Version 3.1 Connect Response Message

The Connect Response message is used to answer a request for access to a virtual disk. The message is supplied as a parameter to the LASTport AsnServerRcv process with the *AsnConnect* event. The message is also used as data in the LASTport Connect Response message.

Figure 3–6 shows the message format for a Connect Response message. Table 3–11 describes message fields.

Figure 3–6 Version 3.1 Connect Response Message Format

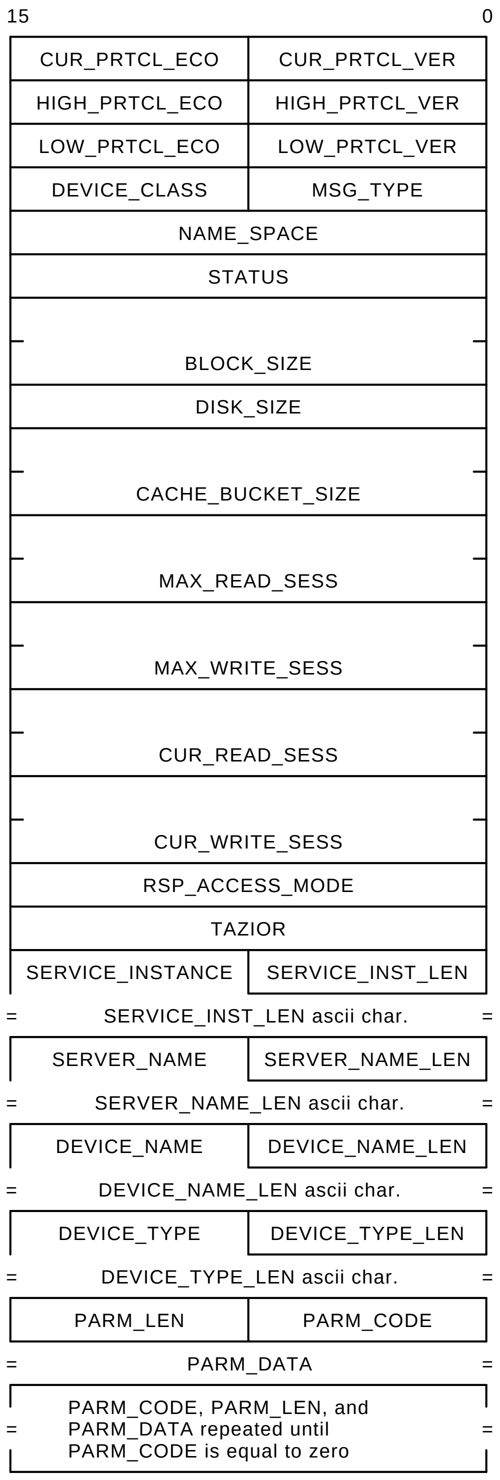

Table 3–11 Version 3.1 Connect Response Message Fields

Name	Length	Description
CUR_PRTCL_VER	1 byte	Protocol version of this message and all following messages (current version is 3.1). In specifiying a version, the server can use the highest common version between the client and the server. This field dictates the protocol version used for the remainder of the messages used for this connection.
CUR_PRTCL_ECO	1 byte	ECO level of CUR_PRTCL_VER for this message and all following messages (current ECO is 0). The server can use the highest common ECO between the client and the server. This field dictates the protocol ECO level used for the remainder of the messages for this connection.
HIGH_PRTCL_VER	1 byte	Highest protocol version supported by node.
HIGH_PRTCL_ECO	1 byte	Highest ECO level supported by node.
LOW_PRTCL_VER	1 byte	Lowest protocol version supported by node.
LOW_PRTCL_ECO	1 byte	Lowest ECO level supported by node.
MSG_TYPE	1 byte	The value 11 (Connect Response message).
DEVICE_CLASS	1 byte	A code indicating the type of device which this service instance is offered on. The following codes are defined: 0 = Fixed Disk 1 = Reserved 2 = Printer (not currently supported) 3 = Reserved 4 = WORM 5 = CDROM 6 = CDROM jukebox 7 = Magneto-optical disk 8 = Magneto-optical jukebox 9 = Removable 3.5-inch floppy 10 = Removable 5.25-inch floppy 11 = Removable disk 12-255 = Reserved
NAME_SPACE	2 bytes unsigned	The Name Space of the service connected. Name Spaces are described in Table 1–1.
STATUS	2 bytes signed	Response status. The Connect Response Message always sends back a value of success (1). If there is an error the Connect Response Message is sent encapsulated in a Disconnect Response Message. In the encapsulated Connect Response Message the following status values are defined: –1 = No such service –2 = Write protected –3 = Access denied –4 = Maximum reference count exceeded –5 = Reserved –6 = Wrong version –7 = Invalid password encoding

(continued on next page)

Table 3–11 (Cont.) Version 3.1 Connect Response Message Fields

Name	Length	Description
BLOCK_SIZE	4 bytes unsigned	An integer specifying the number of bytes per disk block. A value of 0 is illegal.
DISK_SIZE	4 bytes unsigned	An integer specifying the number of blocks of storage for a virtual disk. A value of 0 is illegal.
CACHE_BUCKET_SIZE	4 bytes unsigned	An integer specifying the number of bytes that the server uses to fill each “bucket” in its cache. Clients can use this value to determine read patterns which would cause “cache-striping” across multiple servers when run-time load balancing is used. A value of 0 is illegal.
MAX_READ_SESS	4 bytes unsigned	An integer specifying the maximum number of simultaneous readers that can be opened to this service.
MAX_WRITE_SESS	4 bytes unsigned	An integer specifying the maximum number of simultaneous writers that can be opened to this service.
CUR_READ_SESS	4 bytes unsigned	An integer specifying the current number of read sessions opened to this service.
CUR_WRITE_SESS	4 bytes unsigned	An integer specifying the current number of write sessions opened to this service.
RSP_ACCESS_MODE	2 byte bitmask	Bit mask indicating the access mode for the connection. Servers fill in this field based on the requested access mode from a Connect Response message, or can indicate a default access mode selected by the server if no particular access mode was requested by the client. Table 3–12 defines the meaning of the bits.
TAZIOR	2 bytes	Transmit as zero and ignore on receipt.
SERVICE_INST_LEN	1 byte	Length of next field. A byte containing the length of the SERVICE_INSTANCE field in bytes. A value of 0 is illegal.
SERVICE_INSTANCE	SERVICE_INST_LEN bytes	Service Instance. An array of ASCII characters describing an instance of the requested service name. These characters are constrained as described in Section 1.3.5.
SERVER_NAME_LEN	1 byte	Length of the next field. A byte containing the length of the SERVER_NAME field in bytes. A value of 0 is illegal.
SERVER_NAME	SERVER_NAME_LEN bytes	Server name. An array of ASCII characters describing the name of a particular server being solicited. These characters are constrained as described in Section 1.3.5.
DEVICE_NAME_LEN	1 byte	Length of the next field. A byte containing the length of the DEVICE_NAME field in bytes. A value of 0 is illegal.

(continued on next page)

Table 3–11 (Cont.) Version 3.1 Connect Response Message Fields

Name	Length	Description
DEVICE_NAME	DEVICE_NAME_LEN bytes	Device name. An array of ASCII characters describing the name assigned to a particular port by a server that is being requested. These characters are constrained as described in Section 1.3.5.
DEVICE_TYPE_LEN	1 byte	Length of the next field. A byte containing the length of the DEVICE_TYPE field in bytes. A value of 0 indicates that no device type description is provided.
DEVICE_TYPE	DEVICE_TYPE_LEN bytes	Device type. An array of ASCII characters describing a particular device type being requested. Device type strings are implementation specific, and are constrained as described in Section 1.3.5. Table [TBS] lists globally defined DEVICE_TYPE names.
PARM_CODE	1 byte	A parameter code. Parameter codes are defined in Table 3–13.
PARM_LEN	1 byte	The length of the following field in bytes. Note that the parameter length no longer includes the code and length bytes in the count.
PARM_DATA	PARM_LEN bytes	Parameter data

Table 3–12 Version 3.1 RSP_ACCESS_MODE Bits

Bit	Name	Description
0	READ_ACCESS	When set, indicates that the server is providing read access for this service.
1	WRITE_ACCESS	When set, indicates that the server is providing write access for this service.
2	TIMEOUT	When set, indicates that the service is deleted when a time interval (specified in CONN_TIMEOUT) has expired with no further connections to the service. (If CONN_TIMEOUT is set to zero, the service is deleted when disconnected.) When clear, indicates that the service will never be deleted (that is, the CONN_TIMEOUT is infinite).
3	ZERO_DISCONN	When set, indicates that the virtual disk associated with this service is initialized on disconnect. All the data on the virtual disk is cleared.
4–6		TAZIOR.
7	PASSW_READ	When set, indicates that the password is *not* required for read access.
8	PASSW_WRITE	When set, indicates that the password is *not* required for write access.

(continued on next page)

Table 3–12 (Cont.) Version 3.1 RSP_ACCESS_MODE Bits

Bit	Name	Description
9	RATING	When clear, indicates that the rating for this service is calculated dynamically. When set, indicates that the rating for this service is static. A rating from 0 to 65535 is used for load balancing across redundant services.
10	PERM_FREE_STORAGE	When set, this indicates that the server supports permanent free storage. Permanent free storage means that data created in free disk space is valid across re-boots of the server.
11	READ_AHEAD	When set, this indicates that the server will fill the cache by reading ahead.
12	READ_BEHIND	When set, this indicates that the server will fill the cache by reading behind.
13–15		TAZIOR.

Table 3–13 Version 3.1 Connect Response Message Parameter Codes

Code	Length	Name	Description
0			Denotes the end of a parameter list.
1	2 bytes	CYLINDERS	The number of cylinders for the device connected.
2	1 byte	SECTORS	The number of sectors per track for the device connected.
3	1 byte	TRACKS	The number of tracks per cylinder for the device connected.
4	4 bytes	CONN_TIMEOUT	Indicates the amount of time, in minutes, that this service exists with no connections. If no connections are made to this service in CONN_ TIMEOUT number of minutes, the service is deleted. Servers time out connections to reclaim shared resources from a node that can have hung or crashed. This parameter is meaningful only if the disconnect policy TIMEOUT is selected (in the DISCON_POLICY bitmask). If the disconnect policy TIMEOUT is selected and this parameter is not specified, a default value is used.
5	2 bytes	SERVICE_RATING	A rating of the associated service. Ratings change dynamically based on server load or can be set to a static value.
6	4 bytes	FREE_BLOCKS	An integer specifying the number of free blocks of storage available for creating services. This is the total number of free blocks that the server has allocated for this device.
7	4 bytes	INSTANCE_QUOTA	An integer specifying the maximum number blocks that an individual service instance can use. This is a server specified parameter that applies to all service instances.
8–127			Reserved for Digital use.
128–255			Reserved for Name Space–specific usage.

3.7 Version 3.1 Parameter Request Message

The Parameter Request message is used to obtain or query information about a particular service during an active LASTport/Disk session. The message is supplied as a parameter to the LASTport AsnClientEvent process with the *AsnTransSend* event.

Figure 3–7 shows the message format for a Parameter Request message. Table 3–14 describes message fields.

Figure 3–7 Version 3.1 Parameter Request Message Format

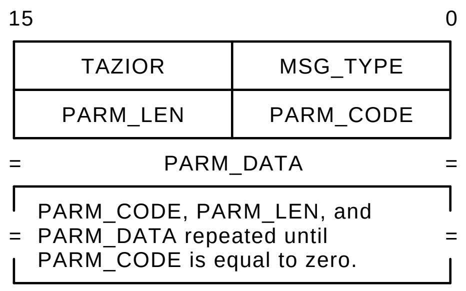
15
0
TAZIOR
MSG_TYPE
PARM_LEN
PARM_CODE
PARM_DATA
PARM_CODE, PARM_LEN, and PARM_DATA repeated until PARM_CODE is equal to zero.

Table 3–14 Version 3.1 Parameter Request Message Fields

Name	Length	Description
MSG_TYPE	1 byte	The value 12 (Parameter Request message)
TAZIOR	1 byte	Transmit as zero and ignore on receipt.
PARM_CODE	1 byte	A parameter code. Parameter codes are defined in Table 3–15.
PARM_LEN	1 byte	The length of the following field in bytes. Note that the parameter length no longer includes the code and length bytes in the count.
PARM_DATA	PARM_LEN bytes	Parameter data.

Table 3–15 Version 3.1 Parameter Request Message Parameter Codes

Code	Length	Name	Description
0			Denotes the end of a parameter list.
1	4 bytes	CONN_TIMEOUT	Indicates the amount of time, in minutes, that this service exists with no connections. If no connections are made to this service in CONN_ TIMEOUT number of minutes, the service is deleted. Servers time out connections to reclaim shared resources from a node that can have hung or crashed. This parameter is meaningful only if the disconnect policy TIMEOUT is selected (in the DISCON_POLICY bitmask). If the disconnect policy TIMEOUT is selected and this parameter is not specified, a default value is used.

(continued on next page)

Table 3–15 (Cont.) Version 3.1 Parameter Request Message Parameter Codes

Code	Length	Name	Description
2	n bytes	NEW_SERVICE_PASSW	An array of ASCII characters used to modify the service password. The SERVICE_PASSW field in this message MUST be valid (or null) for the server password to be modified.
3	2 bytes	STATIC_RATING	Used to set the value used for static rating. A rating from 0 to 65535 is used for load balancing across redundant services. This function can only be used when creating a service.
4	2 bytes	ACCESS_MODE_MASK	Bit mask that allows the specification of various assess features. The bits "enabling set access" are only meaningful if a service is being accessed supports the desired function. Table 3–16 defines the meaning of the bits.
5	2 bytes	DISCON_POLICY	Bit mask specifying the requested disconnect policy that the server should provide. The meaning of the bits (when set) is as follows: • bit 0 = TIMEOUT. When set, indicates that the service is deleted when a time interval (specified in CONN_TIMEOUT) has expired with no further connections to the service. (If CONN_TIMEOUT is set to zero, the service is deleted when disconnected.) When clear, indicates that the service will never be deleted (that is, the CONN_TIMEOUT is infinite). • bit 1 = ZERO_DISCONN. When set, indicates that the virtual disk associated with this service is initialized on disconnect. All the data on the virtual disk is cleared. • bits 2–15 = TAZIOR.
6	4 bytes	SET_READ_REFC	The read reference count is an integer specifying the maximum number of simultaneous readers allowed for a particular service. A value of 1 indicates exclusive read access.
7	4 bytes	SET_WRITE_REFC	The write reference count is an integer specifying the maximum number of simultaneous writers allowed for a particular service. A value of 1 indicates exclusive write access.
8	4 bytes	BLOCK_SIZE	An integer specifying the number of bytes per disk block. If no size is specified, a default block size for the disk is used. Currently only a value of 512 is supported.
9	4 bytes	DISK_SIZE	An integer specifying the number of blocks of storage for a virtual disk.

(continued on next page)

Table 3–15 (Cont.) Version 3.1 Parameter Request Message Parameter Codes

Code	Length	Name	Description
10	2 bytes	CACHE_READ_POLICY	Bit mask indicating type of cache read policy that the server uses to fill each "bucket" in its cache. The client can specify any combination of read-ahead and read-behind. The meaning of the bits (when set) is: bit 0 = READ_AHEAD. The server fills the cache by reading ahead. bit 1 = READ_BEHIND. The server wills fill the cache by reading behind.
11	n bytes	NEW_SERVICE_NAME	An array of ASCII characters used to modify the service name. The client must have write access to the service.
12–127			Reserved for Digital use.
128–255			Reserved for Name Space–specific usage.

Table 3–16 Version 3.1 Parameter Request Message ACCESS_MODE_MASK Bits

Bit	Name	Description
0	REQUEST_READ_ACCESS	When set, indicates that the client is requesting read access for this service.
1	TAZIOR	Transmit as zero and ignore on receipt.
2	REQUEST_WRITE_ACCESS	When set, indicates that the client is requesting write access for this service.
3–6	TAZIOR	Transmit as zero and ignore on receipt.
7	DISABLE_PASSW_READ	When set, indicates that the password is NOT required for read access.
8	DISABLE_PASSW_WRITE	When set, indicates that the password is NOT required for write access.
9	RATING	When clear, indicates that the rating for this service is calculated dynamically. When set, indicates that the rating for this service is static. A rating from 0 to 65535 is used for load balancing across redundant services. If the parameter STATIC_RATING is not provided, a default is used.
10–15		TAZIOR.

3.8 Version 3.1 Parameter Response Message

The Parameter Response message is used to repsond to a request for information about a particular service during an active LASTport/Disk session. The message is supplied as a parameter to the LASTport AsnServerEvent process with the *AsnTransRspMsgSend* event.

Figure 3–8 shows the message format for a Parameter Response message. Table 3–17 describes message fields.

Figure 3–8 Version 3.1 Parameter Response Message Format

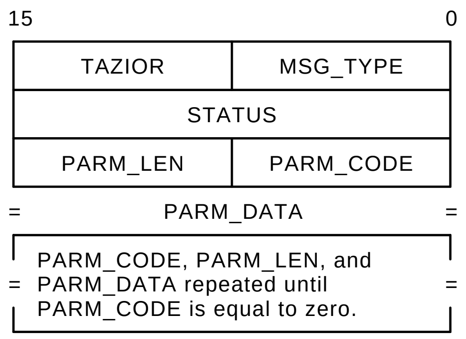
15
0
TAZIOR
MSG_TYPE
STATUS
PARM_LEN
PARM_CODE
PARM_DATA
PARM_CODE, PARM_LEN, and PARM_DATA repeated until PARM_CODE is equal to zero.

Table 3–17 Version 3.1 Parameter Response Message Fields

Name	Length	Description
MSG_TYPE	1 byte	The value 13 (Parameter Response message)
TAZIOR	1 byte	Transmit as zero and ignore on receipt.
STATUS	2 bytes signed	Response status. A positive value indicates success status. Negative values indicate an error. Status code values are defined separately for each service class. Status code values are as follows: 1 = Success 0, –1, –2 = Reserved –3 = Invalid parameter data –4 = Invalid password encoding
PARM_CODE	1 byte	A parameter code. Parameter codes are defined in Table 3–18.
PARM_LEN	1 byte	The length of the following field in bytes. Note that the parameter length no longer includes the code and length bytes in the count.
PARM_DATA	PARM_LEN bytes	Parameter data.

Table 3–18 Version 3.1 Parameter Response Message Parameter Codes

Code	Size (bytes)	Name	Description
0			Denotes the end of a parameter list.
1	4	CONN_TIMEOUT	Indicates the amount of time, in minutes, that this service exists with no connections. If no connections are made to this service in CONN_TIMEOUT number of minutes, the service is deleted. Servers time out connections to reclaim shared resources from a node that can have hung or crashed. This parameter is meaningful only if the disconnect policy TIMEOUT is selected (in the DISCON_POLICY bitmask). If the disconnect policy TIMEOUT is selected and this parameter is not specified, a default value is used.
2	4	BLOCK_SIZE	An integer specifying the number of bytes per disk block.
3	4	DISK_SIZE	An integer specifying the number of blocks of storage for a virtual disk.
4	2	SERVICE_RATING	A rating of the associated service. Ratings change dynamically based on server load or can be set to a static value.
5	4	CACHE_BUCKET_SIZE	An integer specifying the number of bytes that the server uses to fill each “bucket” in its cache. Clients can use this value to determine read patterns that would cause “cache-striping” across multiple servers when run-time load balancing is used. A value of 0 is illegal.
6	4	MAX_READ_SESS	An integer specifying the maximum number of simultaneous readers that can be opened to this service.
7	4	MAX_WRITE_SESS	An integer specifying the maximum number of simultaneous writers that can be opened to this service.
8	4	CUR_READ_SESS	An integer specifying the current number of read sessions opened to this service.
9	4	CUR_WRITE_SESS	An integer specifying the current number of write sessions opened to this service.
10	2	RSP_ACCESS_MODE	Bit mask indicating the access mode for the connection. Servers fill in this field based on the requested access mode from the client or can indicate a default access mode selected by the server if no particular access mode was requested by the client. Table 3–12 defines the meaning of the bits.
11–127			Reserved for Digital use.
128–255			Reserved for Name Space–specific usage.

3.9 Version 3.1 Data Request Message

The Data Request message is used to request application data. The message is supplied as a parameter to the LASTport AsnClientTransSend process with the *AsnTransSend* event.

Figure 3–9 shows the format of the Data Request message. Table 3–19 describes message fields.

Figure 3–9 Version 3.1 Data Request Message Format

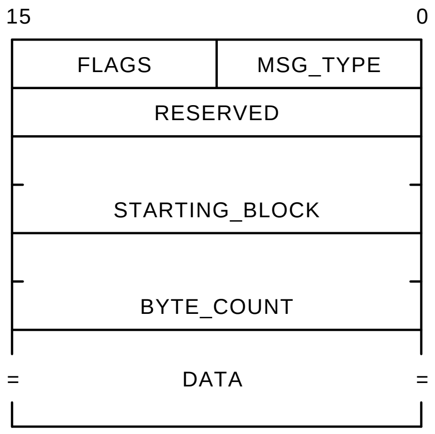

Table 3–19 Version 3.1 Data Request Message Fields

Name	Length	Description
MSG_TYPE	1 byte	Message type fields are as follows: 2 = Read disk 3 = Write disk 6 = Purge cache† 16 = Update request
FLAGS	1 byte bitmask	Bit mask indicating special processing for the Data Request. The meaning of the bits (when set) is as follows: bit 0 = Synchronous write bits 1–7 = TAZIOR
RESERVED	2 bytes unsigned	TAZIOR.
STARTING_BLOCK	4 bytes unsigned	The logical block number on the virtual disk to begin the requested operation.
BYTE_COUNT	4 bytes unsigned	The number of bytes of disk data to be transferred as a result of the Data Request operation.
DATA		Any supplied disk data for the Data Request operation.

†This function purges the cache entries allocated to the current service but does not guarantee that pending writes are completed. Current implementations might purge cache entries before asynchronous writes complete. Future implementations should guarantee that all writes have comleted before purging the cache.

3.10 Version 3.1 Data Response Message

The Data Response message is used to respond to a request for application data. The message is supplied as a parameter to the LASTport AsnServerTransRcv process with the *AsnTransRspMsgSend* event.

Figure 3–10 shows the format of the Data Response message. Table 3–20 describes message fields.

Figure 3–10 Version 3.1 Data Response Message Format

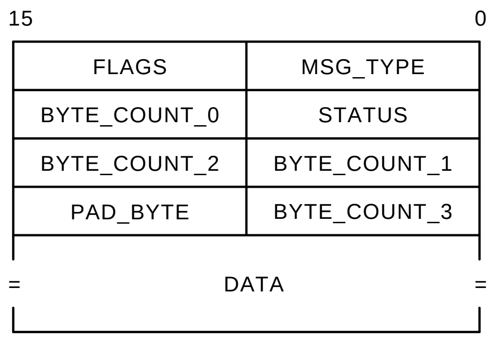

Table 3–20 Version 3.1 Data Response Message Fields

Name	Length	Description
MSG_TYPE	1 byte	Message type fields are as follows: 4 = Read response 5 = Write response 7 = Flush cache response 17 = Update response
FLAGS	1 byte bitmask	Bit mask indicating special processing done as a result of the received Data Request message. The meaning of the bits (when set) is as follows: bit 0 = Synchronous write bits 1–7 = TAZIOR

(continued on next page)

Table 3–20 (Cont.) Version 3.1 Data Response Message Fields

Name	Length	Description
STATUS	1 byte unsigned	Response status A positive value indicates success status. Negative values indicate an error. A value of 0 is illegal. The following status values are defined: 1 = Success –1 = No such service –2 = Write protected –3 = Access denied –4 = Maximum reference count exceeded –5 = Device error –6 = Wrong version –7 = Invalid Block number or range –8 = Reserved –9 = Reserved –10 = Reserved –11 = Reserved –12 = Device off line –13 = Medium off line –12 = Device off line –14 = Parity Error –15 = Drive Fatal Error –16 = Data check Read/Write
BYTE_COUNT	4 bytes unsigned	The number of bytes of disk data actually transferred in response to the received Data Request message.
PAD_BYTE	1 byte	Transmit as zero and ignore on receipt. Note that this pad field has been added from LASTport/Disk Version 2.0 to LASTport/Disk Version 3.0. This extra byte permits Version 2.0 and Version 3.0 implementations to retain the same offset references to the fixed fields in the message, as well as word-aligns and potential data for a Data Response message.
DATA		Any supplied disk data in response to the received Data Request operation.

3.11 Version 3.1 Disconnect Request Message

The Disconnect Request message is used to terminate a virtual disk session. The message is supplied as a parameter to the LASTport AsnClientRcv process with the *AsnDisconnect* event. The message is also used as data in the LASTport Disconnect Request message. The following are valid arguments:

- Disconnect reason (unsigned word)
- Disconnect descriptor (text string)

The disconnect reason is supplied from this list:

- 0 = No reason
- 1 = Session disconnected by third party
- 2 = Insufficient resources
- 3 = Fatal drive error
- 4 = Drive not responding

3.12 Version 3.1 Disconnect Response Message

The Disconnect Response message answers a request to terminate a virtual disk session. The message is supplied as a parameter to the LASTport AsnServerRcv process with the *AsnDisconnect* event. The message is also used as data in the LASTport Connect Response message. The following are valid arguments:

- Disconnect reason (unsigned word). The disconnect reason is supplied from the list of reasons for the Disconnect Request message.
- Disconnect descriptor (text string). If a Disconnect Response is initiated by a Connect Request message, the disconnect descriptor is the LASTport/Disk Connect Response message with the reason in the STATUS field. If the Disconnect Response is initiated by a Disconnect Request message, the disconnect descriptor is null.

A

LASTport/Disk Version 3.0 Messages

This appendix describes the following LASTport/Disk Version 3.0 messages:

- Solicit Request
- Solicit Response
- Connect Request
- Connect Response
- Data Request
- Data Response
- Disconnect Request
- Disconnect Response

Figures A–1 through A–6 illustrate message formats. Accompanying tables describe message fields in detail.

Because the LASTport/Disk wire protocol is meant to operate independently of lower-level transport protocols, the message formats are represented as procedure calls whose fields are bit streams, shown right to left. The right-most bit is the first bit transmitted on the wire (little-endian). Each field is an integer multiple of eight bits. The equals sign (=) is used in Figures A–1 through A–6 to indicate fields of varying or indeterminate length.

A.1 Version 3.0 Solicit Request Message

The Solicit Request message is used to solicit a particular service or set of services from servers offering LASTport/Disk virtual disks on the LAN. The message is supplied as a parameter to the LASTport SolClientEvent process with the *SolReqMsgSend* event. The LASTport Solicitation layer transmits the message in accordance with the LASTport Name Service model.

Figure A–1 shows the message format for a Solicit Request message. Table A–1 describes message fields.

Figure A–1 Version 3.0 Solicit Request Message Format

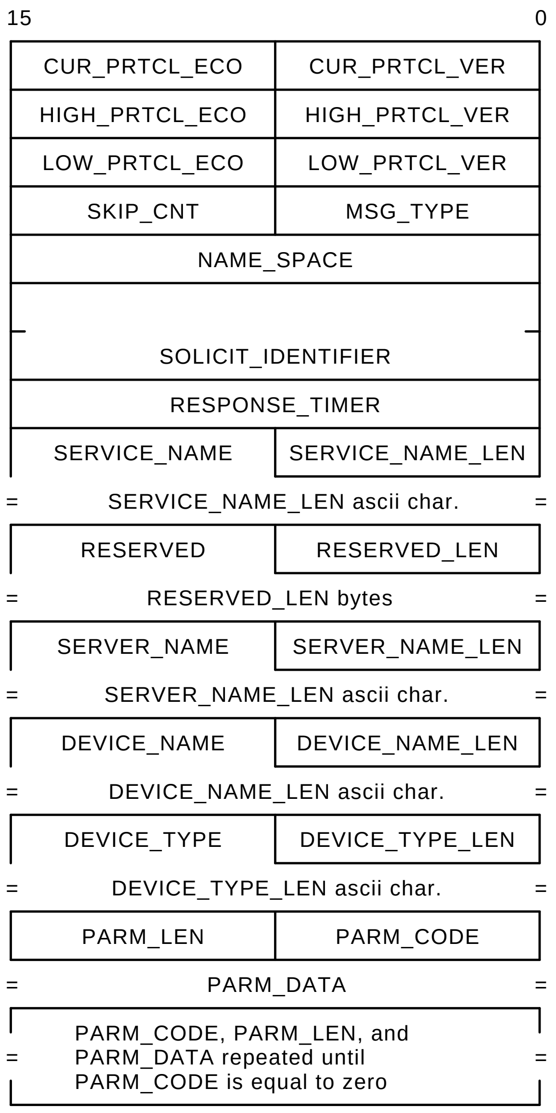

Table A–1 Version 3.0 Solicit Request Message Fields

Name	Length	Description
CUR_PRTCL_VER	1 byte	Protocol version of this message and all following messages (current version is 3).
CUR_PRTCL_ECO	1 byte	ECO level of CUR_PRTCL_VER for this message and all following messages (current ECO is 0).
HIGH_PRTCL_VER	1 byte	Highest protocol version supported by node.
HIGH_PRTCL_ECO	1 byte	Highest ECO level supported by node.
LOW_PRTCL_VER	1 byte	Lowest protocol version supported by node.
LOW_PRTCL_ECO	1 byte	Lowest ECO level supported by node.
MSG_TYPE	1 byte	The value 8 (Solicit Request message)
SKIP_CNT	1 byte	The number of times that the client has seen a particular service. A client Solicit Request can require more than one response from a server. This field indicates to the server how many service instances to “skip over” before returning the next response. Normally this field is set to zero on an initial solicitation and contains the SERVICE_INST_CNT field from a server Solicit Response message when a server indicates that there is more information available.
NAME_SPACE	2 bytes unsigned	The Name Space being solicited. Name Spaces are described in Table 1–1.
SOLICIT_IDENTIFIER	4 bytes unsigned	Identifier produced by the soliciting node that uniquely identifies the Solicit Request message. The soliciting node uses this identifier to correlate corresponding Solicit Response message(s).
RESPONSE_TIMER	2 bytes unsigned	Response timer (seconds) that starts when the Solicit Request message is sent. The soliciting node uses this timer to time out when waiting for Solicit Response messages. Servers should use this value as a maximum for the local response timer.
SERVICE_NAME_LEN	1 byte	Length of the next field. A byte containing the length of the SERVICE_NAME field in bytes. A value of 0 indicates that a particular service is not being solicited.
SERVICE_NAME	SERVICE_NAME_LEN bytes	Service name. An array of ASCII characters describing the name of a particular service being solicited. A server responds to a Solicit Request message only if it offers the service name specified. Service name characters are constrained as described in Section 1.3.5.
RESERVED_LEN	1 byte	Length of the next field. A byte containing the length of the RESERVED field in bytes.
RESERVED	RESERVED_LEN bytes	Reserved.
SERVER_NAME_LEN	1 byte	Length of the next field. A byte containing the length of the SERVER_NAME field in bytes. A value of 0 indicates that no particular server is being solicited.

(continued on next page)

Table A–1 (Cont.) Version 3.0 Solicit Request Message Fields

Name	Length	Description
SERVER_NAME	SERVER_NAME_LEN bytes	Server name. An array of ASCII characters describing the name of a particular server being solicited. A server responds to a Solicit Request message only if its server name matches the name supplied in this field. Server name characters are constrained as described in Section 1.3.5.
DEVICE_NAME_LEN	1 byte	Length of the next field. A byte containing the length of the DEVICE_NAME field in bytes. A value of 0 indicates that a particular device is not being solicited.
DEVICE_NAME	DEVICE_NAME_LEN bytes	Device name. An array of ASCII characters describing a particular device on a server. Servers respond with information for the named device only if it exists. Device names are constrained as described in Section 1.3.5.
DEVICE_TYPE_LEN	1 byte	Length of the next field. A byte containing the length of the DEVICE_TYPE field in bytes. A value of 0 indicates that no special device type is being requested.
DEVICE_TYPE	DEVICE_TYPE_LEN bytes	Device type. An array of ASCII characters describing a particular device type being requested. Servers respond only if they offer the service on the specified device type. Device type strings are implementation specific, and are constrained as described in Section 1.3.5.
PARM_CODE	1 byte	A parameter code.
PARM_LEN	1 byte	The length of the following field in bytes. Note that the PARM_LEN field does not include the code and length bytes in the count.
PARM_DATA	PARM_LEN bytes	Parameter data.

A.2 Version 3.0 Solicit Response Message

The Solicit Response message is used to answer a request for information about a particular service or set of services. The message is supplied as a parameter to the LASTport SolServerEvent process with the *SolRspMsgSend* event.

Figure A–2 shows the message format for a Solicit Response message. Table A–2 describes message fields.

Figure A–2 Version 3.0 Solicit Response Message Format

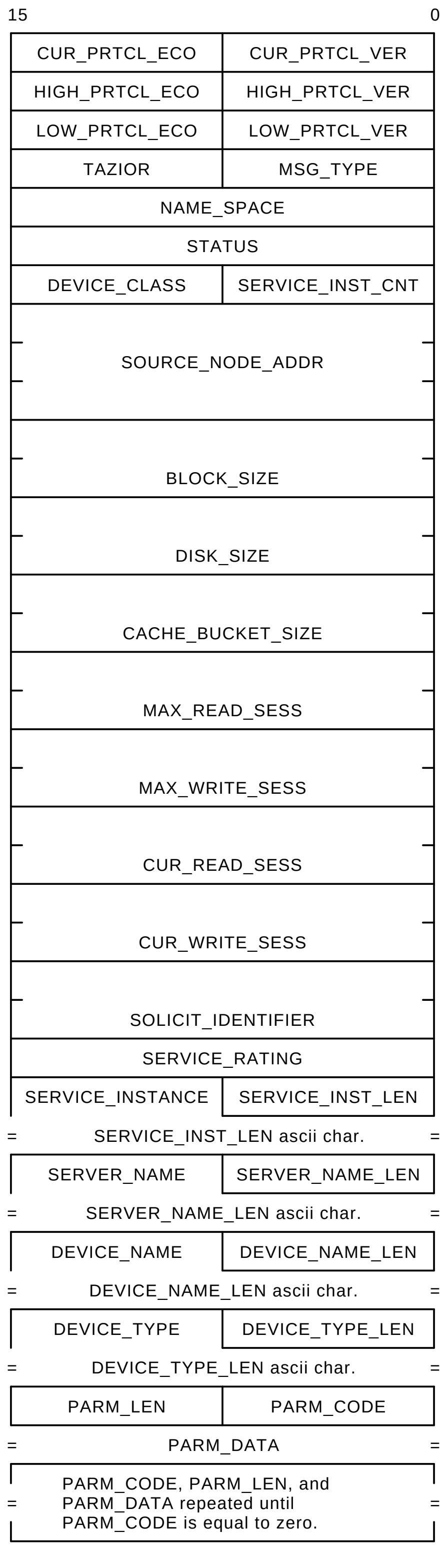

Table A–2 Version 3.0 Solicit Response Message Fields

Name	Length	Description
CUR_PRTCL_VER	1 byte	Protocol version of this message (current version is 3). A responding node is allowed to respond using the highest version number supported by both the soliciting and the responding node.
CUR_PRTCL_ECO	1 byte	ECO level of CUR_PRTCL_VER for this message (current ECO is 0). A responding node is allowed to respond using the highest ECO level supported by both the soliciting and the responding node.
HIGH_PRTCL_VER	1 byte	Highest protocol version supported by node.
HIGH_PRTCL_ECO	1 byte	Highest ECO level supported by node.
LOW_PRTCL_VER	1 byte	Lowest protocol version supported by node.
LOW_PRTCL_ECO	1 byte	Lowest ECO level supported by node.
MSG_TYPE	1 byte	The value 9 (Solicit Response message).
TAZIOR	1 byte	Transmit as zero and ignore on receipt.
NAME_SPACE	2 bytes unsigned	The Name Space of the described service instance descriptor. Name Spaces are described in Table 1–1.
STATUS	2 bytes signed	Response status. A positive value indicates success status. Negative values indicate an error. Status code values are defined separately for each service class. Status code values are as follows: 1 = Success 2 = Success, more information available. 3 = Success, end of list. 0, –1, –2 = Reserved –3 = No such object
SERVICE_INST_CNT	1 byte	The instance number of the service information being returned. A client solicit can require more than one response from a server. This field indicates the number of times a server has responded, including the current response. A client uses this value in the SKIP_CNT field of a Solicit Request message when more information is being requested.
DEVICE_CLASS	1 byte	A code indicating the type of on which device this service instance being offered. The following codes are defined: 0 = Fixed Disk 1 = Tape 2 = Printer 3 = Reserved 4 = WORM 5 = CDROM 6–255 = Reserved

(continued on next page)

Table A–2 (Cont.) Version 3.0 Solicit Response Message Fields

Name	Length	Description
SOURCE_NODE_ADDR	6 byte field	Transmitting node must fill this field with a value equal to the datalink source address of the named server node. Receiving node must reference this field and ignore the datalink source address of message.
BLOCK_SIZE	4 bytes unsigned	An integer specifying the number of bytes per disk block. A value of 0 is illegal.
DISK_SIZE	4 bytes unsigned	An integer specifying the number of blocks of storage for a virtual disk. A value of 0 is illegal.
CACHE_BUCKET_SIZE	4 bytes unsigned	An integer specifying the number of bytes that the server uses to fill each “bucket” in its cache. Clients can use this value to determine read patterns which would cause “cache-striping” across multiple servers when run-time load balancing is used. A value of 0 is illegal.
MAX_READ_SESS	4 bytes unsigned	An integer specifying the maximum number of simultaneous readers that can be opened to this service.
MAX_WRITE_SESS	4 bytes unsigned	An integer specifying the maximum number of simultaneous writers that can be opened to this service.
CUR_READ_SESS	4 bytes unsigned	An integer specifying the current number of read sessions opened to this service.
CUR_WRITE_SESS	4 bytes unsigned	An integer specifying the current number of write sessions opened to this service.
SOLICIT_IDENTIFIER	4 bytes unsigned	Identifier produced by the soliciting node that uniquely identifies the solicitor’s Solicit Request message. This value is equal to the SOLICIT_ IDENTIFIER field of the received Solicit Request message that is being responded to by this Solicit Response message.
SERVICE_RATING	2 bytes unsigned	A rating of the associated service instance. ratings change dynamically based on server load.
SERVICE_INST_LEN	1 byte	Length of next field. A byte containing the length of the SERVICE_INSTANCE field in bytes. A value of 0 is illegal.
SERVICE_INSTANCE	SERVICE_INST_LEN bytes	Service Instance. An array of ASCII characters describing an instance of the requested service name. Clients use this field in Connect Request messages to establish a connection to a server. These characters are constrained as described in Section 1.3.5.
SERVER_NAME_LEN	1 byte	Length of the next field. A byte containing the length of the SERVER_NAME field in bytes. A value of 0 is illegal.
SERVER_NAME	SERVER_NAME_LEN bytes	Server name. An array of ASCII characters describing the name of a particular server being solicited. These characters are constrained as described in Section 1.3.5.

(continued on next page)

Table A–2 (Cont.) Version 3.0 Solicit Response Message Fields

Name	Length	Description
DEVICE_NAME_LEN	1 byte	Length of the next field. A byte containing the length of the DEVICE_NAME field in bytes. A value of 0 indicates that information is not being returned for a particular device.
DEVICE_NAME	DEVICE_NAME_LEN bytes	Device name. An array of ASCII characters describing the name assigned to a particular port by a server. These characters are constrained as described in Section 1.3.5.
DEVICE_TYPE_LEN	1 byte	Length of the next field. A byte containing the length of the DEVICE_TYPE field in bytes. A value of 0 indicates that no device type description is provided.
DEVICE_TYPE	DEVICE_TYPE_LEN bytes	Device type. An array of ASCII characters describing a particular device type being requested. Device type strings are implementation specific, and are constrained as described in Section 1.3.5. Table [TBS] lists globally defined DEVICE_TYPE names.
PARM_CODE	1 byte	A parameter code.
PARM_LEN	1 byte	The length of the following field in bytes. Note that the parameter length no longer includes the code and length bytes in the count.
PARM_DATA	PARM_LEN bytes	Parameter data.

A.3 Version 3.0 Connect Request Message

The Connect Request message is used to initiate access to a virtual disk. The message is supplied as a parameter to the LASTport AsnClientRcv process with the *AsnConnect* event. The message is also used as data in the LASTport Connect Request message.

Figure A–3 shows the message format for a Connect Request message. Table A–3 describes message fields. Table A–4 defines the Connect Request message ACCESS_MODE_MASK bits.

Figure A–3 Version 3.0 Connect Request Message Format

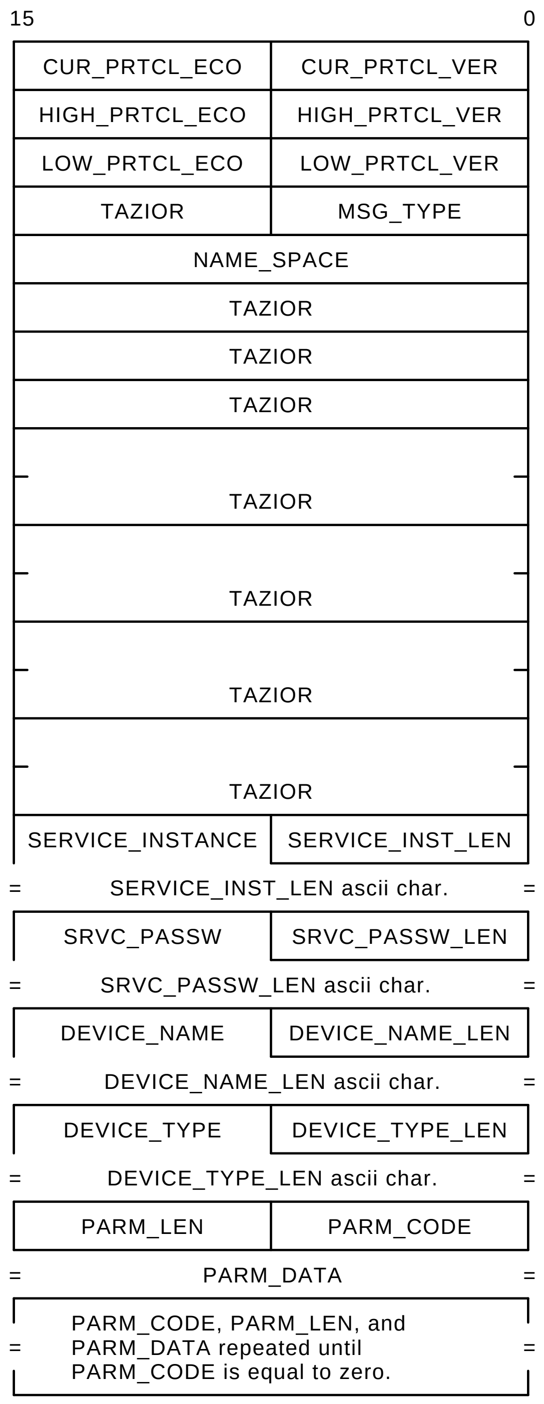

Table A–3 Version 3.0 Connect Request Message Fields

Name	Length	Description
CUR_PRTCL_VER	1 byte	Protocol version of this message (current version is 3).
CUR_PRTCL_ECO	1 byte	ECO level of CUR_PRTCL_VER for this message (current ECO is 0).
HIGH_PRTCL_VER	1 byte	Highest protocol version supported by node.
HIGH_PRTCL_ECO	1 byte	Highest ECO level supported by node.
LOW_PRTCL_VER	1 byte	Lowest protocol version supported by node.
LOW_PRTCL_ECO	1 byte	Lowest ECO level supported by node.
MSG_TYPE	1 byte	The value 10 (Connect Request message).
TAZIOR	1 byte	Transmit as zero and ignore on receipt.
NAME_SPACE	2 bytes unsigned	The Name Space of the service instance being requested. Name Spaces are described in Table 1–1.
TAZIOR	2 byte bitmask	Transmit as zero and ignore on receipt.
ACCESS_MODE_MASK	2 byte bitmask	Bit mask indicating which attributes should be modified via this access. A value of 0, in most cases, indicates that the server should use its default policy or value. The bits “enabling set access” are only meaningful if a service is being created (that is, the Create/Connect Service bit is set in the CONNECT_MODIFIER). Table A–4 defines the bits.
TAZIOR	2 byte bitmask	Transmit as zero and ignore on receipt.
TAZIOR	4 bytes unsigned	Transmit as zero and ignore on receipt.
TAZIOR	4 bytes unsigned	Transmit as zero and ignore on receipt.
TAZIOR	4 bytes unsigned	Transmit as zero and ignore on receipt.
TAZIOR	4 bytes unsigned	Transmit as zero and ignore on receipt.
SERVICE_INST_LEN	1 byte	Length of next field. A byte containing the length of the SERVICE_INSTANCE field in bytes. A value of 0 is illegal.
SERVICE_INSTANCE	SERVICE_INST_LEN bytes	Service Instance. An array of ASCII characters describing the instance of the requested service name. Clients fill in this field from the received field in the Solicit Response message (or other naming service). These characters are constrained as described in Section 1.3.5.
SRVC_PASSW_LEN	1 byte	Length of the next field. A byte containing the length of the SRVC_PASSW field in bytes. A value of 0 indicates that no password is provided.
SRVC_PASSW	SRVC_PASSW_LEN bytes	Service password. An array of ASCII characters used to pass the service password to a server. A server rejects a connection to a password protected service if the client does not supply the correct password.
DEVICE_NAME_LEN	1 byte	Length of the next field. A byte containing the length of the DEVICE_NAME field in bytes. A value of 0 indicates that a particular device is not being requested.

(continued on next page)

Table A–3 (Cont.) Version 3.0 Connect Request Message Fields

Name	Length	Description
DEVICE_NAME	DEVICE_NAME_LEN bytes	Device name. An array of ASCII characters describing a particular device on a server. Servers respond with information only for the named device, if it exists. Device names are constrained as described in Section 1.3.5.
DEVICE_TYPE_LEN	1 byte	Length of the next field. A byte containing the length of the DEVICE_TYPE field in bytes. A value of 0 indicates that no special device type is being requested.
DEVICE_TYPE	DEVICE_TYPE_LEN bytes	Device type. An array of ASCII characters describing a particular device type being requested. Servers complete the connect request only if they offer the service on the specified device type. Device type strings are implementation specific, and are constrained as described in Section 1.3.5. Table [TBS] lists globally defined DEVICE_TYPE names.
PARM_CODE	1 byte	A parameter code.
PARM_LEN	1 byte	The length of the following field in bytes. Note that the parameter length no longer includes the code and length bytes in the count.
PARM_DATA	PARM_LEN bytes	Parameter data.

Table A–4 Version 3.0 Connect Request Message ACCESS_MODE_MASK Bits

Bit	Name	Description
0	REQUEST_READ_ACCESS	When set, indicates that the client is requesting read access for this service. This function can be requested when creating a service or when connecting to an existing service.
1	RESERVED	Transmit as zero and ignore on receipt.
2	REQUEST_WRITE_ACCESS	When set, indicates that the client is requesting write access for this service. This function can be requested when creating a service or when connecting to an existing service.
3–15		TAZIOR

A.4 Version 3.0 Connect Response Message

The Connect Response message is used to answer a request for access to a virtual disk. The message is supplied as a parameter to the LASTport AsnServerRcv process with the *AsnConnect* event. The message is also used as data in the LASTport Connect Response message.

Figure A–4 shows the message format for a Connect Response message. Table A–5 describes message fields.

Figure A–4 Version 3.0 Connect Response Message Format

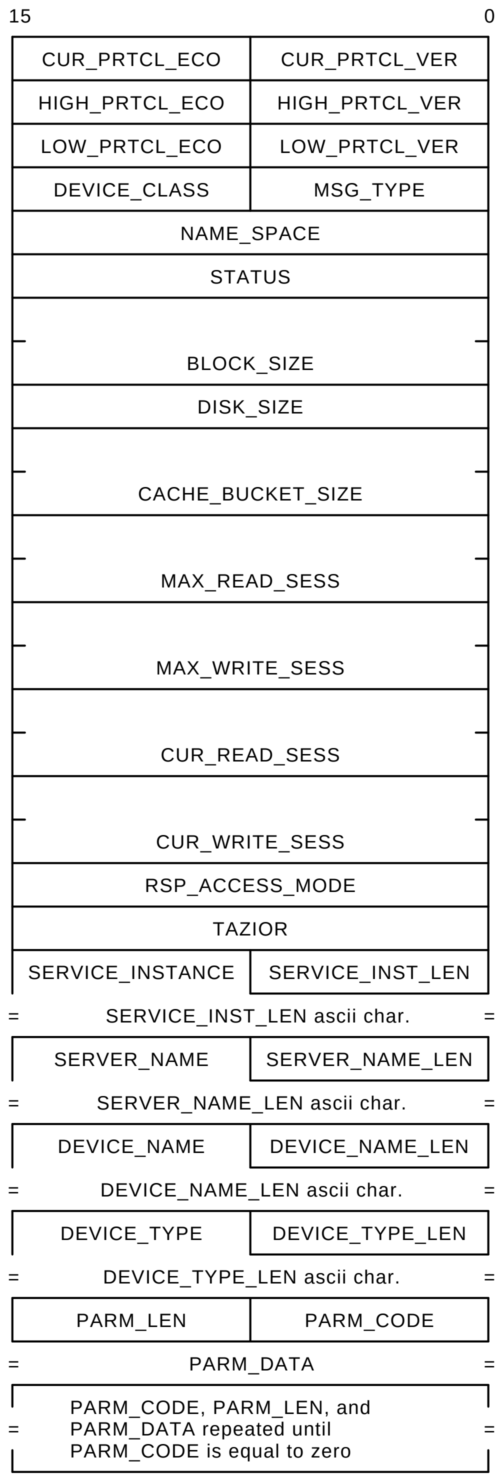

Table A–5 Version 3.0 Connect Response Message Fields

Name	Length	Description
CUR_PRTCL_VER	1 byte	Protocol version of this message (current version is 3). The server can use the highest common version between the client and the server. This field dictates the protocol version used for the remainder of the messages used for this connection.
CUR_PRTCL_ECO	1 byte	ECO level of CUR_PRTCL_VER for this message (current ECO is 0). The server can use the highest common ECO between the client and the server. This field dictates the protocol ECO level used for the remainder of the messages for this connection.
HIGH_PRTCL_VER	1 byte	Highest protocol version supported by node.
HIGH_PRTCL_ECO	1 byte	Highest ECO level supported by node.
LOW_PRTCL_VER	1 byte	Lowest protocol version supported by node.
LOW_PRTCL_ECO	1 byte	Lowest ECO level supported by node.
MSG_TYPE	1 byte	The value 11 (Connect Response message).
DEVICE_CLASS	1 byte	A code indicating the type of device which this service instance is offered on. The following codes are defined: 0 = Fixed Disk 1 = Tape 2 = Printer 3 = Reserved 4 = WORM 5 = CDROM 6 = CDROM jukebox 7 = Magneto-optical disk 8 = Magneto-optical jukebox 9 = Removable 3.5-inch floppy 10 = Removable 5.25-inch floppy 11 = Removable disk 12–255 = Reserved
NAME_SPACE	2 bytes unsigned	The Name Space of the service connected. Name Spaces are described in Table 1–1.
STATUS	2 bytes signed	Response status. The Connect Response Message always returns a value of success (1). If an error occurs, the Connect Response Message is sent encapsulated in a Disconnect Response Message. In the encapsulated Connect Response Message the following status values are defined: –1 = No such service –2 = Write protected –3 = Access denied –4 = Maximum reference count exceeded –5 = Reserved –6 = Wrong version –7 = Invalid password encoding
BLOCK_SIZE	4 bytes unsigned	An integer specifying the number of bytes per disk block. A value of 0 is illegal.

(continued on next page)

Table A–5 (Cont.) Version 3.0 Connect Response Message Fields

Name	Length	Description
DISK_SIZE	4 bytes unsigned	An integer specifying the number of blocks of storage for a virtual disk. A value of 0 is illegal.
CACHE_BUCKET_SIZE	4 bytes unsigned	An integer specifying the number of bytes that the server uses to fill each “bucket” in its cache. Clients can use this value to determine read patterns which would cause “cache-striping” across multiple servers when run-time load balancing is used. A value of 0 is illegal.
MAX_READ_SESS	4 bytes unsigned	An integer specifying the maximum number of simultaneous readers that can be opened to this service.
MAX_WRITE_SESS	4 bytes unsigned	An integer specifying the maximum number of simultaneous writers that can be opened to this service.
CUR_READ_SESS	4 bytes unsigned	An integer specifying the current number of read sessions opened to this service.
CUR_WRITE_SESS	4 bytes unsigned	An integer specifying the current number of write sessions opened to this service.
RSP_ACCESS_MODE	2 byte bitmask	Bit mask indicating the access mode for the connection. Servers fill in this field based on the requested access mode from a Connect Request message, or can indicate a default access mode selected by the server if no particular access mode was requested by the client. Table A–6 defines the meaning of the bits.
TAZIOR	2 bytes	Transmit as zero and ignore on receipt.
SERVICE_INST_LEN	1 byte	Length of next field. A byte containing the length of the SERVICE_INSTANCE field in bytes. A value of 0 is illegal.
SERVICE_INSTANCE	SERVICE_INST_LEN bytes	Service Instance. An array of ASCII characters describing an instance of the requested service name. These characters are constrained as described in Section 1.3.5.
SERVER_NAME_LEN	1 byte	Length of the next field. A byte containing the length of the SERVER_NAME field in bytes. A value of 0 is illegal.
SERVER_NAME	SERVER_NAME_LEN bytes	Server name. An array of ASCII characters describing the name of a particular server being solicited. These characters are constrained as described in Section 1.3.5.
DEVICE_NAME_LEN	1 byte	Length of the next field. A byte containing the length of the DEVICE_NAME field in bytes. A value of 0 is illegal.
DEVICE_NAME	DEVICE_NAME_LEN bytes	Device name. An array of ASCII characters describing the name assigned to a particular port by a server that is being requested. These characters are constrained as described in Section 1.3.5.

(continued on next page)

Table A–5 (Cont.) Version 3.0 Connect Response Message Fields

Name	Length	Description
DEVICE_TYPE_LEN	1 byte	Length of the next field. A byte containing the length of the DEVICE_TYPE field in bytes. A value of 0 indicates that no device type description is provided.
DEVICE_TYPE	DEVICE_TYPE_LEN bytes	Device type. An array of ASCII characters describing a particular device type being requested. Device type strings are implementation specific, and are constrained as described in Section 1.3.5. Table [TBS] lists globally defined DEVICE_TYPE names.
PARM_CODE	1 byte	A parameter code. Parameter codes are defined in Table A–7.
PARM_LEN	1 byte	The length of the following field in bytes. Note that the parameter length no longer includes the code and length bytes in the count.
PARM_DATA	PARM_LEN bytes	Parameter data

Table A–6 Version 3.0 RSP_ACCESS_MODE Bits

Bit	Name	Description
0	READ_ACCESS	When set, indicates that the server is providing read access for this service.
1	WRITE_ACCESS	When set, indicates that the server is providing write access for this service.

Table A–7 Version 3.0 Connect Response Message Parameter Codes

Code	Length	Name	Description
0			Denotes the end of a parameter list.
1	2 bytes	CYLINDERS	The number of cylinders for the device connected.
2	1 byte	SECTORS	The number of sectors per track for the device connected.
3	1 byte	TRACKS	The number of tracks per cylinder for the device connected.
4–127			Reserved for Digital use.
128–255			Reserved for Name Space–specific usage.

A.5 Version 3.0 Data Request Message

The Data Request message is used to request application data. The message is supplied as a parameter to the LASTport AsnClientTransSend process with the *AsnTransSend* event.

Figure A–5 shows the format of the Data Request message. Table A–8 describes message fields.

Figure A–5 Version 3.0 Data Request Message Format

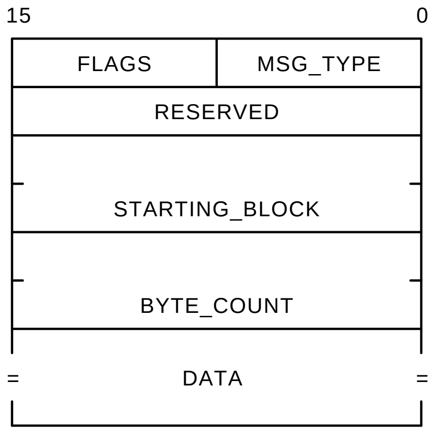

Table A–8 Version 3.0 Data Request Message Fields

Name	Length	Description
MSG_TYPE	1 byte	Message type fields are as follows: 2 = Read disk 3 = Write disk 6 = Purge cache
FLAGS	1 byte bitmask	Bit mask indicating special processing for the data request. The meaning of the bits (when set) is as follows: bit 0 = Synchronous write bits 1–7 = TAZIOR
RESERVED	2 bytes unsigned	TAZIOR.
STARTING_BLOCK	4 bytes unsigned	The logical block number on the virtual disk to begin the requested operation.
BYTE_COUNT	4 bytes unsigned	The number of bytes of disk data to be transferred as a result of the Data Request operation.
DATA		Any supplied disk data for the Data Request operation.

A.6 Version 3.0 Data Response Message

The Data Response message is used to respond to a request for application data. The message is supplied as a parameter to the LASTport AsnServerTransRcv process with the *AsnTransRspMsgSend* event.

Figure A–6 shows the format of the Data Response message. Table A–9 describes message fields.

Figure A–6 Version 3.0 Data Response Message Format

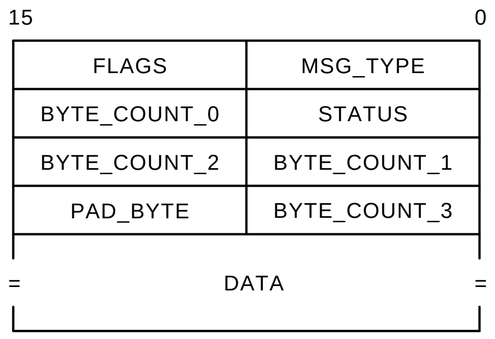

Table A–9 Version 3.0 Data Response Message Fields

Name	Length	Description
MSG_TYPE	1 byte	Message type fields are as follows: 4 = Read response 5 = Write response 7 = Flush cache response
FLAGS	1 byte bitmask	Bit mask indicating special processing done as a result of the received Data Request message. The meaning of the bits (when set) is as follows: bit 0 = Synchronous write bits 1–7 = TAZIOR

(continued on next page)

Table A–9 (Cont.) Version 3.0 Data Response Message Fields

Name	Length	Description
STATUS	1 byte signed	Response status. A positive value indicates success status. Negative values indicate an error. A value of 0 is illegal. The following status values are defined: 1 = Success –1 = No such service –2 = Write protected –3 = Access denied –4 = Maximum reference count exceeded –5 = Device error –6 = Wrong version –7 = Invalid Block number or range –8 = Reserved –9 = Reserved –10 = Reserved –11 = Reserved –12 = Device off line –13 = Medium off line –12 = Device off line –14 = Parity Error –15 = Drive Fatal Error –16 = Data check Read/Write
BYTE_COUNT	4 bytes unsigned	The number of bytes of disk data actually transferred in response to the received Data Request message.
PAD_BYTE	1 byte	TAZIOR. Note that this pad field has been added from LASTport/Disk Version 2.0 to LASTport/Disk Version 3.0. This extra byte permits Version 2.0 and Version 3.0 implementations to retain the same offset references to the fixed fields in the message, as well as word-aligns and potential data for a “read_response”.
DATA		Any supplied disk data in response to the received Data Request operation.

A.7 Version 3.0 Disconnect Request Message

The Disconnect Request message is used to terminate a virtual disk session. The message is supplied as a parameter to the LASTport AsnClientRcv process with the *AsnDisconnect* event. The message is also used as data in the LASTport Disconnect Request message. The following are valid arguments:

- Disconnect reason (unsigned word)
- Disconnect descriptor (text string)

The disconnect reason is supplied from this list:

- 0 = No reason
- 1 = Session disconnected by third party
- 2 = Insufficient resources
- 3 = Fatal drive error
- 4 = Drive not responding

A.8 Version 3.0 Disconnect Response Message

The Disconnect Response message answers a request to terminate a virtual disk session. The message is supplied as a parameter to the LASTport AsnServerRcv process with the *AsnDisconnect* event. The message is also used as data in the LASTport Connect Response message. The following are valid arguments:

- Disconnect reason (unsigned word). The disconnect reason is supplied from the list of reasons for the Disconnect Request message.
- Disconnect descriptor (text string). If a Disconnect Response is initiated by a Connect Request message, the disconnect descriptor is the LASTport/Disk Connect Response message with the reason in the STATUS field. If the Disconnect Response is initiated by a Disconnect Request message, the disconnect descriptor is null.

B
LASTport/Disk Version 2.0 Messages

This appendix summarizes the formats of Service Names and Service Descriptors in LASTport/Disk Version 2.0 messages. The Version 2.0 naming service protocol must be supported concurrently with the Version 3.0 or later protocol. All values in message formats are expressed as hexadecimal values unless otherwise noted.

B.1 LASTport/Disk Version 2.0 Naming Service Protocol

The LASTport/Disk protocol provides messages for dynamically locating services on a LAN. The protocol allows the user to locate a service, obtain its attributes, and connect to it.

The LASTport/Disk Version 2.0 protocol uses the following messages:

- Solicit Request (Service Name, Service Descriptor)
- Solicit Response (Service Name, Service Descriptor)
- Connect Request (Service Name, Service Descriptor)
- Connect Response (Service Name, Service Descriptor)
- Data Request (data) †
- Data Response (data) †
- Disconnect Request (Service Name, Service Descriptor) †
- Disconnect Response (Service Name, Service Descriptor) †

The Data Request, Data Response, Disconnect Request and Disconnect Response messages are identical to the corresponding Version 3.0 messages with the exception of the PAD_BYTE field in the Data Response message. In the Version 2.0 protocol, this field is omitted. The field was added to the Version 3.0 protocol to enhance the performance of data copies by aligning the data portion of the message to an even byte.

B.1.1 Service Names and Service Descriptors

Each LASTport/Disk Version 2.0 message contains a Service Name. Service Names are byte-counted strings and are limited in length to 255 bytes. Service Names are passed as an item list with the following format:

```
Field Description               Field Value
-----------------               -----------

Service Name:
   Begin Item List:             01 (1 byte)
        Service Name:           Length: (1 byte, length of next two
                                   fields+1 - includes this field in count)
                                Code: (1 byte, value 02 = Service Name)
                                Data: ( n bytes - ASCII name)
   End Item List:               00 (1 byte)
```

† See Appendix A.

```
Service Descriptor:)
   Begin Item List:            01 (1 byte)
        Item:                  Length: (1 byte)
                               Code: (1 byte)
                               Data: (n bytes - Item specific format)
   End Item List:              00 (1 byte)
```

Service Descriptors are word-counted fields, and thus are limited only by the length of the LAN datagram size.

The only Item Code used in the Service Name field is Item Code 02, the Service Name code, which specifies the block size in bytes. Service Descriptor Item Codes used in the Version 2.0 protocol are as follows:

00—End of List
01—Status
02—Block Size in bytes
03—Disk Size in blocks
04—Access control enabled
05—Access mode
06—Maximum References Allowed
07—Current Connections

B.1.2 Solicit Request Message

When a client wants to locate a service on the LAN, the client system sends a Solicit Request message that is broadcast to all LAN nodes listening to the LASTport multicast/protocol (09-00-2b-04-00-00/4180) pair. When a server receives a Solicit Request message with a request matching a service offered by an authorized client, the server answers with a Solicit Response message.

Each message has two parts: the Service Name and the Service Descriptor. The Service Descriptor is used in the Solicit Response message but not in the Solicit Request message.

In the following example, the name ESS_DISK is sent in a Solicit Request message. This name appears on the wire as follows:

```
Field Description                 Field Value
-----------------                 -----------

Service Name:
    Begin Item List:              01
        Service Name:             Length: 0A(16) (including this byte)
                                  Code: 02 (Service Name)
                                  Data: "ESS_DISK"
    End Item List (1 byte):       00

Service Descriptor:
    (none)
```

B.1.3 Solicit Response Message

When a server matches a Service Name, it sends a Solicit Response message to inform the client that the service is available at this node.

The Service Name in the Solicit Response message is in the same format as in the Solicit Request message. The Service Descriptor is used in the Solicit Response message to describe the service. As shown in the following example, the attributes of the service are passed as an item list in the same format as the Service Name.

```
Field Description               Field Value
-----------------               -----------

Service Name:
    Begin Item List:            01
        Service Name:           Length: 0A(16) Code: 02 Data: "ESS_DISK"
    End Item List (1 byte):     00

Service Descriptor:
   Begin Item List (1 byte):    01
        Access mode:            Length: 03 Code: 05 Service is write: 01
        Access control enable:  Length: 03 Code: 04 Access Ctrl Enable: 00
        Connections:            Length: 04 Code: 07 Connections: 0000
        Max references allowed: Length: 04 Code: 06 MaxRef: 0004
   End Item List (1 byte):      00
```

B.1.4 Connect Request Message

Once the service has been located, the client can connect to it. The Connect Request message contains two data items. The first is the LASTport/Disk Service Name; the second is connect data, additional parameters needed to connect to this service.

The Service Name is passed in an item list in the same format as in the Solicit Request message.

The Connect Request message Service Descriptor is passed as a block of data with a fixed format. The format of the Connect Request message is as follows:

```
Field Description               Field Value
-----------------               -----------

Service Name:
    Begin Item List:            01
        Service Name:           Length: 0A(16) Code: 02 Data: "ESS_DISK"
    End Item List (1 byte):     00

Service Descriptor:
    Access Mode (1 byte):       00 - Multi-user read
                                01 - Single user write
                                02 - Server sets access mode
    Version (2 bytes):                  02,00 (Version Number 2.0)
    Password (40 decimal bytes):  Length: 27(16) Password: "..."
                                     (39 decimal byte ASCII string)
```

B.1.5 Connect Response Message

The Connect Response message is sent to the client in response to the Connect Request message and to return the attributes of the service.

The Connect Response message sends the Service Name as an item list in the same format as in the Solicit Request message.

The response data sent to the connecting client is formatted as an item list, as shown in the following sample Connect Response message:

```
Field Description               Field Value
-----------------               -----------

Service Name:
    Begin Item List:            01
        Service Name:           Length: 0A(16) Code: 02 Data: "ESS_DISK"
    End Item List (1 byte):     00
```

```
Service Descriptor:
   Begin item list (1 byte):      01
        Status:                   Length: 03  Code: 01  Success: 01
        Block size:               Length: 04  Code: 02  Block size: 200(16)
        Disk size:                Length: 06  Code: 03  Disk size: 1000
   End of item list (1 byte):     00
```

C
Mapping LASTport/Disk Version 3.0 and Later to LASTport Version 3.1

The LASTport Version 3.1 transport is an ECO to the Version 3.0 implementation. The specific ECO is to check the incoming message Version and ECO fields. If the version is 3 and the ECO is 1, then the SERVICE_DESCRIPTOR_LEN (unsigned word) field in the Solicit Request and Connect Request messages is zeroed in the message before the messages are processed.

The message formats defined for Solicit Request, Solicit Response, Connect Request, and Connect Response messages map directly to the SERVICE_DESCRIPTOR field in the LAST Version 3.1 protocol. To ensure transport independence, the LASTport/Disk protocol defines the Service Name as part of the message format and therefore does not use the LASTport Version 3.1 SERVICE_NAME field. The value of the LASTport SERVICE_NAME field must be 0 when LASTport/Disk Version 3.0 message formats are mapped to the LASTport Version 3.1 protocol.

D
Architecturally Defined Constants, Ranges, and Default Values

This appendix lists global constants, ranges, and default values for LASTport and LASTport/Disk architecture specifications.

Table D–1 Service Parameter Defaults of Remotely Created Services

Parameter/Parameter Bits	Length	Default Value	Description
ACCESS_MODE_MASK	2 bytes		
REQUEST_READ_ACCESS		set	Request read access
ENABLE_SET_READ_REFC		clear	Use default value
REQUEST_WRITE_ACCESS		clear	No write access
ENABLE_SET_WRITE_REFC		clear	Use default value
ENABLE_BLOCK_SIZE		clear	Use default value
ENABLE_DISK_SIZE		clear	Use default value
ENABLE_DISCON_POLICY		clear	Use default value
DISABLE_PASSW_READ		clear	Password (if defined) required for read access.
DISABLE_PASSW_WRITE		clear	Password (if defined) required for write access.
RATING		set	Use Static Rating
PERM_FREE_STORAGE		set	Data will be saved across server re-boots
ALLOCATE_DEVICE		clear	Not requesting to allocate a device
DISCON_POLICY	2 bytes		
TIMEOUT		set	Delete service on disconnect. CONN_TIMEOUT=0
ZERO_DISCONN		clear	Do not clear disk on disconnect
SET_READ_REFC	4	1 bytes	One reader
SET_WRITE_REFC	4 bytes	0	No writers
BLOCK_SIZE	4 bytes	512 blocks	
DISK_SIZE	4 bytes	1024 blocks	

(continued on next page)

Table D–1 (Cont.) Service Parameter Defaults of Remotely Created Services

Parameter/Parameter Bits	Length	Default Value	Description
SERVICE_INSTANCE	21 bytes	VXT_addr_num	The server-generated name is made up of a fixed portion (VXT_) and a variable portion. The variable portion is made up of the servers MAC address (addr) and a modulo identifier (num - 2 bytes) that is unique for that service.
SRVC_PASSW	n bytes	null	255 characters maximum
CONN_TIMEOUT	4 bytes	0 minutes	Delete service on disconnect.
NEW_SRVC_PASSW	n bytes	null	255 characters maximum
STATIC_RATING	2 bytes	1	Use a low rating.
PASSW_ENCODE	4 bytes	1	If password is defined use clear text
CACHE_READ_POLICY	2 bytes		
READ_AHEAD		set	The server will fill the cache by reading ahead.
READ_BEHIND		clear	No reading behind.
ALLOCATE_TIMEOUT	4 bytes	30 Minutes	If an allocated device sees no activity for this time period, the service is deleted.
DEVICE_CLASS	1 byte	0	Assume fixed disk.

Glossary

advertisement

A data multicast primitive message that allows LASTport server nodes to advertise node-specific **services** to their potential clients. The LASTport **Circuit layer** users Advertisement, Solicit Request, and Solicit Response messages to maintain connectivity.

application

See **system application**.

association

The active relationship between a client instance and a server instance for a particular **service** provided by a server instance. An association is required to perform transactions between a **client** and a **server**.

Association layer

A component of the LASTport protocol that manages connections and transactions between a **client** and a **server**.

checksum

A sum of digits or bits used to verify whether a number or an operation is valid.

circuit

The relationship between two nodes. Only a single circuit exists between any two nodes, regardless of the number of **associations**.

Circuit layer

A component of the LASTport protocol that defines and maintains the **circuit topology** and distributes messages between source and destination nodes.

circuit topology

The set of paths (logical links) that connect client and server nodes in a **local area network**. The circuit topology is mapped during the solicitation process. The topology can change depending on which paths and network adapters are available.

client

The software requesting a **service** of a **server**.

client request semantics

A set of rules that ensures execution of a **transaction** by a **server** at least once or causes an **association** to fail. The **client** specifies the retry interval and the retry count for the transaction.

client response semantics

A set of rules that ensures execution of a **transaction** by a **client** at least once. No execution at the **server** occurs after the **response** is delivered successfully.

client–server protocol

A protocol in which one **node** has priority over the other. For example, in the LASTport protocol, the **client** has priority over the **server**. The client initiates all requests for a **service**, and the **server** must respond to the client's request.

communication interface

A mechanism by which system components located on the various nodes in a network can interact through the exchange of messages conveying data and control information. Communication interfaces are usually implemented using communication **protocols**.

commutative transaction

A **transaction** that can be completed in any order without introducing errors in the result. The LASTport protocol requires that all concurrently executing transactions be commutative. This requirement allows the protocol to process multiple transactions concurrently, because transactions can complete across the session interface in the order in which responses arrive. No state need be maintained to preserve the order of transactions.

component

A constituent element of a **system**.

congestion control

The mechanisms that prevent catastrophic collapse of the distributed system (also known as "livelock") when instantaneous demand for service exceeds the available network capacity.

connection

The function that creates a new, agent–specific **association** between a **client** and a **server**. A connection is initiated by a client.

dally

A system application mechanism used to prevent flooding of the client's network adapter when multiple servers respond to a Solicit Request message.

datagram

A **packet** assembled into a network frame for transmission to remote nodes.

data link

An extended **local area network (LAN)** segment.

Data Link layer

A network protocol component that transmits datagrams between a **client** and a **server**. The Data Link layer assembles packets into network frames for transmission to remote nodes as datagrams.

data link port

The device that connects a **node** to a **data link**, also called an adapter. A data link port is associated with a single node. A node can have multiple data link ports.

data slot

A division of a LAT **packet** that contains the data from a LAT session.

data transfer service

A **service** that enables a **client** to send requests to a **server** and that enables servers to respond.

directory service

A **service** that locates a **server** for the **system application**. Directory services occur in the LASTport **Solicitation layer**.

disconnection

The act of terminating an **association**.

end-to-end guarantee

A guarantee ensuring that all transaction segments transmitted by a **client** arrive at a **server**, that the complete **transaction** is processed by the server, and that the results are returned to the client.

frame

The primitive unit of data transfer between nodes on the **local area network**.

header

The control information prefixed in a message text, such as source or destination code or message type.

idempotent transaction

A **transaction** that can be repeated without changing the system state. The repeated transaction request has the “same strength” as the original request, assuming that the request is repeated in isolation from other requests.

idempotent procedure

A procedure that can use the LASTport protocol directly, such as Digital Equipment Corporation’s Mass Storage Control Protocol (MSCP), access to read-only data in general, and name translation.

interface

A point of interaction between two components or between a **system** and its environment. The LASTport protocol contains interfaces of the following types:

- User interfaces
- Communications interfaces
- Programming interfaces
- Data interfaces
- System interfaces

LAN

See **local area network**.

LAT protocol

A communications protocol used in a **local area network**. LAT is the abbreviation for local area transport.

layer

A set of software that is ordered partially with respect to dependency; that is higher layers can depend on lower layers, but lower layers cannot depend oon higher layers.

The LASTport protocol includes three functional layers: Circuit, Association, and Solicitation layers.

local area network (LAN)

Loosely, the set of nodes capable of communicating with any other node in the set across some data link segment. All nodes in the LAN must have at least one data link segment in common.

message

A **datagram** that is formatted with one of the architecturally defined LASTport protocol headers and identified in the local area network header as a LASTport protocol packet.

multicast

The ability to “address” a single **message** for receipt by any **data link port** that enables the multicast address. This capability supports the LASTport advertisement and solicit primitives.

Name Space

A collection of logically related names, such as LASTport/Disk **Service Names**, that identify objects accessible on a network.

Name Space Handler

A system application data structure that identifies a **Name Space**.

Network layer

A layer that provides user control of and access to operational parameters and counters in lower layers.

network topology

The physical arrangement and relationship of interconnected nodes in the network.

Name String

An architecturally specified name conveyed in a **Parameter Code**.

node

A computer system on the **local area network**. The LASTport protocol requires only that a node be able to identify itself uniquely in space and time. A 48-bit identifier, such as the hardware address of a particular local area network controller, identifies the node in space, and a 16-bit value identifies a specific instance of the operating system.

packet

The unit of data created by the LASTport **Circuit layer** and transmitted to the **Data Link layer**. A packet is inherently unreliable; that is, the local area network can neither guarantee reliable delivery of packets nor indicate delivery of any given packet.

performance

A measurement of the resources required to perform an operation. The most common measurement is the elapsed time needed to perform an operation. The elapsed time from the end of user input until output data is displayed is called response time. Consumption of other resources such as processor, communication, I/O, and memory resources might also be measured as part of performance.

Parameter Code

A system application code that conveys architecturally specified names called **Name Strings**.

Parameter List

A LASTport/Disk mechanism that allows extensions to message formats and **Name Spaces**.

path maintenance

The process of determining which paths remain viable between a **client** and a **server** after the solicitation process. A **circuit** is maintained as long as messages arrive on a path during a timed interval.

The **client** and **server** follow the same protocol for maintaining the **network topology**.

peer-to-peer transport protocol

A transport protocol such as DECnet NSP, in which data exchanges between nodes operate symmetrically. Usually, only one **transaction** can be conducted at a time, and one transaction must complete before another can start. The protocol used at both nodes is identical. By contrast, in a client–server protocol, one node has priority over the other.

platform

A combination of computer hardware and software that provides an environment for higher-level software.

protocol

A complete specification of a sequence of operations and messages needed to accomplish some specified processing or information exchange.

Examples: DECnet NSP protocol, LASTport protocol, LAT protocol, OSI stack, TCP/IP stack, X.400 protocol.

registration service

A **service** that allows the **system application** to interact with the LASTport protocol. Registration services make the Directory, Association, and Data Transfer services available to the system application. Registration services are not described in the LASTport architecture because they are implemented differently for each **system**.

reliability

The extent to which a **system** or part of a system yields the expected results on repeated trials. “Expected results” means correct output, without unintended side effects, meeting the performance specification. Reliability is measured by mean time between failures (MTBF).

request

The data that a **client** sends to a **server**.

request–response semantics

A set of rules that defines the relationship between a **client** and a **server**. The client sends a **request** to the server, and the server sends a **response** to the client.

response

The data returned by a **server** reacting to the request data supplied by a **client**.

segment

The unit of communication created by the LASTport **Association layer** and passed to the **Circuit layer** for transmission. Each segment has a **transaction identifier**.

segment number

A number that identifies the position of a **segment** in a **transaction**. The segment number is used to reassemble a transaction once the transaction packets reach the destination node.

server

Software that provides a **service** to a **client**.

server name

A **Name String** that uniquely identifies a **server** on the **local area network**.

server request semantics

A set of rules that ensures execution of a **transaction** by a **server** exactly once.

server response semantics

A set of rules that requires a **server** to return the results of a **transaction** to a **client** exactly once.

service

A set of invocable routines that provides a complete, well-defined function. A service comprises a number of operations and can also maintain state.

The software that uses a service is called a **client**; the software providing a service is called a **server**. A given piece of software can be both a client and a server (for different services) at the same time. In a distributed system, client and server may be on different nodes of a network.

Examples: File services, naming services, virtual disk services.

Service Instance

A description of an instance of the requested **Service Name**.

Service Name

The name of a system application virtual disk. Multiple virtual disks or a single disk can be mapped to a single physical medium. Conversely, a single virtual disk can span multiple physical media.

Session layer

A layer that defines the system-dependent aspect of logical link communication. A logical link is a **circuit** on which information flows in two directions. Session contol functions include name to address translation, process addressing, and, in some, systems, process activation and access control.

slot

See **transaction slot**.

slot number

See **transaction slot number**.

solicitation

A function that allows a **client** to transmit data to all **servers** and receive response data from specific servers. A client builds a dynamic service binding capability, in which the decision to connect to a particular server can be deferred until a specific **service** is needed.

Solicitation layer

A component of the LASTport protocol that provides a combined service for directory operations, association management, and task naming.

subsystem

See **system**.

system

A set of **components** organized in a particular fashion and working together to achieve a certain outcome or objective. A system exists within an environment and interacts with objects in that environment.

Systems themselves can be composed to form even larger systems, sometimes referred to as macrosystems; the constituent systems are then known as **subsystems**.

system application

An application program that facilitates the development, management, or use of an information system.

Examples: language compilers, data structure viewers, backup and restore tools, file editors, system builders, command language interpreters, compound document editors, mail viewers.

transaction

The fundamental client–server communication paradigm supported by the LASTport protocol. A transaction comprises of a two-step sequence that takes place within the context of a preexisting association. First, the client establishes the association and supplies request data to the server. The server then returns response data to the client in the context of the transaction that supplied the request data.

transaction identifier

An identifier that includes a **transaction slot number** and that is found in the **header** of every **segment** passed from the **Association layer** to the **Circuit layer**.

transaction slot

An active exchange in the context of its **association**. The number of slots supported by an association defines how many exchanges within that association can be active concurrently.

transaction slot number

A one-byte number that identifies a **slot**. The slot number identifies a **segment** as belonging to a particular **transaction**. All segments for one transaction carry the same slot number for both the **request** and the **response**.

Transport layer

A layer used to route a **datagram** to its destination. A Transport layer also provides **congestion control** and packet lifetime control.

Index