Ethernet CDreader Project
DRAFT Functional Specification

DIGITAL CONFIDENTIAL

Revision/Update Information: V0.1

Elliott A. Drayton

27–July–1989

Abstract

This document describes the functions of the Ethernet CDreader. The MARIAH project will use an Ethernet CDreader as its remote console device for software installation and upgrades.

Review Team

Ellen Batbouta
Paul Jacobi
Kathy Morse
Ben Thomas
John Ywoskus

Digital Equipment Corporation

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may be used or copied only in accordance with the terms of such license.

No responsibility is assumed for the use or reliability of software on equipment that is not supplied by Digital Equipment Corporation or its affiliated companies.

The postpaid READER'S COMMENTS form on the last page of this document requests the user's critical evaluation to assist in preparing future documentation.

The following are trademarks of Digital Equipment Corporation:

DEC	DIBOL	UNIBUS
DEC/CMS	EduSystem	VAX
DEC/MMS	IAS	VAXcluster
DECnet	MASSBUS	VMS
DECsystem–10	PDP	VT
DECSYSTEM–20	PDT	
DECUS	RSTS	
DECwriter	RSX	digital™

This document was prepared using VAX DOCUMENT, Version 1.1

CONTENTS

TABLES

Preface

This Functional Specification describes the goals, capabilities, and external characteristics of Ethernet CDreader.

Associated Documents

1. Ethernet CDreader An Overview
   Elliott A. Drayton, Star::Drayton
2. MARIAH CDreader Client Specification
   Barbara Leahy, STAR::B_LEAHY
3. Local Area System Transport
   Phil Wells, MOSAIC::WELLS
4. Local Area Disk Protocol Architecture
   Marshall R. Goldberg, WSINT::Goldberg

Change History

Table 1: History

Revision Number	Date	Reason for Change
X0.0.1	19-Jun-1989	First Draft
V0.1	18-Jul-1989	Second Draft

CHAPTER 1

PRODUCT OVERVIEW

1.1 Product Summary

This document specifies the functional pieces necessary to develop and access an Ethernet CDreader. The Ethernet CDreader will provide access to Compact Disc Read Only Memories (CD ROM). These CD ROMs can be utilized as common software distribution media by any processor supporting the access of Ethernet CDreaders.

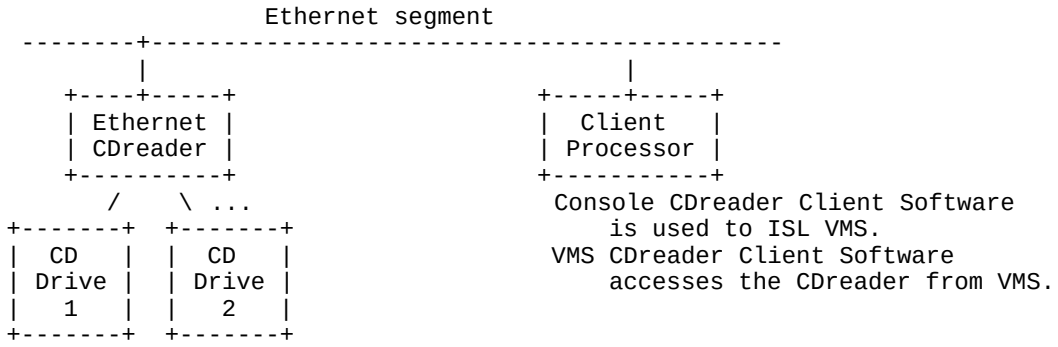
By providing Ethernet CDreader support on our Midrange and High-End systems consoles, imbedded console devices such as the TK70 cartridge tape are no longer required in these systems for software distribution and installation. With this ability to access a CD ROM, our customers can elect to obtain our software on a convenient easy to use Compact Disc (CD). This disc provides a formatted capacity of approximately 622MBytes of low cost "tamper-resistant" random access storage from which Initial System Loads (ISL), Software Upgrades and Layered Product Installations can be performed.

1.2 Environment

The Ethernet CDreader Version 1 system will consist of an Ethernet controller with both thick and thinwire connectors, a CPU, memory, one to six CD drives and the software environment necessary to control it. These items form a stand alone system which operates as a "load and go" system. The software environment in this controller will have no understanding of the file structure on the media placed in the CD drive(s), nor will it have any ability to perform a write operation. Only read operations to a CD ROM are supported. The Ethernet CDreader will respond to read requests by reading the specified blocks of data from a CD and transmitting these blocks back to the requestor via the Ethernet. The Ethernet CDreader will make use of the Local Area Disk (LAD) and Local Area System Transport (LAST) protocols.

By providing the Ethernet CDreader, other processors and operating systems that support LAD, LAST and Ethernet can then access any CD ROM on the reader. By implementing a subset of the full LAD/LAST protocols in the console code of our Midrange and High-End systems, a software distribution CD ROM can then be booted from, or accessed, as if it were a local disk on these systems. This capability allows a newly delivered processor to perform an ISL from a CD ROM on the Ethernet CDreader. A description of the console functions and protocols needed to access the Ethernet CDreader can be found in the MARIAH CDreader Client Specification.

Once an ISL is complete and the resultant operating system is running, it can continue to perform an upgrade. Or layered products can be installed by accessing the software distribution CD ROM on the Ethernet CDreader. The VMS CDreader Client software will provide the necessary software to support VMS access to the Ethernet CDreader.



1.3 Performance Goals

The performance of the Ethernet CDreader can vary due to many conditions. The performance of the Ethernet CDreader will be limited primarily by the access time of a CD ROM drive. These access times can range from 350 msec to 1000 msec. The next limiting factor can be the utilization of the Ethernet on a busy backbone.

It is a goal of this project to load software faster with the Ethernet CDreader than with a TK50 cartridge tape. We intend to use STABACKUP to compare the loading times from the different devices.

The Ethernet CDreader should perform best and is easily characterized when connected to a host processor via a private Ethernet segment. In this configuration the loading of STABACKUP should take an average of TBD (1.5) minutes and will take no more than TBD (3) minutes.

1.3.1 CPU Impact

The CPU in the Ethernet CDreader will be a PVAX class processor which will provide more than enough processing power for controlling access to the CD ROM drives and controlling Ethernet communications.

A processor (eg. MARIAH) which is requesting an ISL is executing code supplied by its console ROMs. This code executes at IPL 31 with all memory being accessed in physical mode. This project will specify the protocols and console command additions necessary to allow consoles to communicate with the Ethernet CDreader. These console enhancements are designed to be operating system(OS) independent and are intented to be used in future processors.

Once a processor (eg. RAVEN) has completed its ISL and VMS is installed, bundled LAST and LAD drivers can then be used to configure a communications path to access the Ethernet CDreader.

1.3.2 Memory Impact

The Ethernet CDreader will have 4 MBytes of RAM Memory. The Ethernet CDreader controlling software will consume no more than 2MB of RAM memory. It is a goal to utilize less than 2MB of memory to allow the remaining memory to be used for the queuing of transmit and receive buffers. To enhance the responsiveness of the Ethernet CDreader a large cache (~ 1MB) will also be used in an attempt to reduce disk accesses.

On any processor (eg. MARIAH) which implements the console protocols and command additions necessary to access an Ethernet CDreader some quantity of additional memory (ROM or RAM) will be required on that system.

On a VMS system which has the LAD/LAST support loaded, memory usage will depend on cache sizes and the number of Ethernet CDreader drives that are configured on the system.

1.3.3 I/O Impact

Systems accessing an Ethernet CDreader will display increased I/O utilization. The increased utilization will be proportional to the number of users accessing the Ethernet CDreader. The VMS CDreader Client software will provide the necessary software to support VMS access of the Ethernet CDreader.

1.3.4 Initialization/Startup

The Ethernet CDreader is designed to be a "load and go" system. During installation of the Ethernet CDreader the results of the power on diagnostics, executed by the Ethernet CD controller, need to be examined. Assuming no failures, the installer can then decide to modify the default boot device string and/or specify a node name for the CD Ethernet controller. Any modifications to these two strings are then saved in the non-volatile RAM (NVR) of the Ethernet CDreader.

Each CD drive on the Ethernet CDreader system is considered a service once the CD ROM media is inserted into a drive. Each service has a name. The service name is always the concatenation of server name and CD drive unit number. The installer will be urged to always physically tag the server and its drives with the clearly legible labels provided with the system. For example, an Ethernet CDreader named "TONY_" with the VMS software distribution CD ROM loaded in CD drive 200 would report it has TONY_200 as an available service.

After completing the installation of the Ethernet CDreader the installer can then cable the reader to a host processor, like a RAVEN. Through the use of the processor console commands the installer can enter the SHOW SERVICES/SOLICIT command. This command will solicit the Ethernet and display all available services. As in the example above, the service TONY_200 would be displayed by the RAVEN console. The installer could request the RAVEN to "BOOT TONY_200".

If a processor supports the SHOW SERVERS/DEVICE/AVAILABLE console command then the installer can see what devices are availiable(empty); and subsequently place the VMS software distribution CD ROM in an available drive. A service with the name, i.e. TONY_100, is then formed and the user can request the RAVEN to "BOOT TONY_100". The VMS software distribution CD ROM contains STABACKUP which will be used to perform the ISL of the RAVEN.

It is suggested that Ethernet CDreader node names be unique to prevent user confusion. When two Ethernet CDreader systems have identical node names causing a replication of service names the resultant behavior can be problematic. There is no means to prevent users from doing this. However, when a user does do this, we attempt to notify the user of duplicate names via an index of the duplicate service names on the console display from the SHOW SERVICES command. This index is not guaranteed to refer to the same service

across two or more executions of the SHOW SERVICE command. Examples of this display can be found in the MARIAH CDreader Console Client Specification.

1.4 Compatibility Goals

The Ethernet CDreader utilizes communications protocols that are based on the LAD/LAST architectures. These architectures need to define their compatibility goals and ECO methodology. A policy of the architectures, needs to specify the minimal set of messages which must be compatible and supported in all future versions of LAD/LAST devices which will support Console Clients.

Since a version of these protocols will be placed in non-upgradeable ROM storage, all future versions of the protocol must support the minimal set of messages in order to maintain compatibility with older implementations of the protocol.

Today a few products have shipped or will have shipped before the LAD/LAST architectures are accepted Digital standards under ECO control. It is a goal of this project to work with these product groups and the responsible architects in an effort to define the Digital standard.

1.4.1 VAXCluster Mixed Version Operation

This product will have no affect on mixed-version VAXClusters operation.

1.4.2 Standards and Architecture Conformance

The Digital standards and architectures that will be adhered to by the Ethernet CDreader V1.0 are:

- Local Area System Transport Architecture
- Local Area Disk Protocol Architecture
- Philips yellow book CD standard
- Ethernet standard

1.4.3 Internationalization

The Ethernet CDreader will utilize any internationalization support provided by the ROM code of the CPU module (EG. TEAMATE II) used as the CD Ethernet controller.

The VMS CDreader Client software is necessary to support the accessing of the Ethernet CDreader. This software will conform to the typical VMS internationalization standards.

1.5 Support and Maintenance Goals

The Ethernet CDreader control software will be provided as loadable software on CD ROM. The Ethernet CDreader will boot in its operating environment from this loadable media. When software updates are necessary a replacement load media can be shipped.

The Mariah CDreader client software imbedded in the console ROM of our systems (eg. RAVEN) is more difficult to upgrade. The problem with this code is that there is a limited amount of patch space available for updates in the CPU ROMs of the future CPUs (EG.

MARIAH). It is a goal of this project to specify console code which is relatively simple and can been exhaustively tested to reduce the likelihood of serious problems.

Any CDreader client software which executes in an operating system environment can be updated as the operating system is updated.

1.5.1 Reliability/Availability/Serviceability

It is a goal of the project to define the Ethernet CDreader hardware as two - Field Replicable Unit (FRU). The Ethernet-CD controller box is one FRU while the tabletop CD drives are the other FRU.

Each Ethernet CDreader will have access to the hardware diagnostics provided with the CPU module (EG. TEAMATE II) used as the CD Ethernet controller.

1.6 Security Goals

In version one of the Ethernet CDreader the only security policy which may be needed is the ability to specify, from an operating system environment that one and only one connection can be made to a specified CD drive. The Ethernet CDreader would reject any attempt of a second connect request. The ability to setup this state from a console client is not mandatory.

Future versions of an Ethernet CDreader could inforce a tighter security policy if necessary.

1.7 Testing Goals

The overall testing plan for the product is TBS.

At specific times during code development baselevels will be made available to PCSG and RSM for testing.

A series of tests must be devised to exercise the console client code.

All software must be tested while performing under varying Ethernet loads, as well as testing with some form of fault insertion.

CHAPTER 2

FUNCTIONAL DEFINITION

2.1 Overview

This chapter describes the functions of three environments.

The first section specifies the functions of a stand-alone Ethernet CDreader.

The second section specifies the functions a console client can perform to interact with an Ethernet CDreader.

The third section specifies the functions of the VMS operating system used to interact with an Ethernet CDreader.

2.2 Ethernet CDreader Operational Description

This section describes the simple operations performed by a stand-alone Ethernet CDreader.

Once an Ethernet CDreader is installed the device awaits messages of the LAD/LAST protocol type and attempts to perform the the requestors function.

The Ethernet CDreader can:

— announce to all interested clients that a server is available

— announce to all interested clients that a service has been loaded

— perform block reads of any CD loaded in one of its CD drives

— notify all interested clients that a service has been unloaded

— return a list of all available drives

— return a list of all drives on the CDreader

In the event of a failure in the server, the device will perform an auto-reboot unless the installer setup the device for no autoboot.

2.3 Ethernet CDreader Console Client Operation

Any processor which supports Ethernet CDreader Client operations from its console, must perform the following operations from its console prompt:

— BOOT service_name - A service_name is the name of a CD ROM drive which contains a CD ROM.

— SHOW SERVICE - Lists the known service_names.

— SHOW SERVICE/SOLICIT - Finds and lists the available service_names.

The following operations are optional and may not appear in all implementations:

— SHOW SERVER - Finds and lists the available servers.

— SHOW SERVER/DEVICE server_name - Finds and lists the available devices on each servers.

— SHOW SERVER/DEVICE/AVAILABLE - Lists only available(empty) devices on each server.

— ALLOCATE server_name/DEVICE=device id - Reserves a device

— DEALLOCATE server_name/DEVICE=device id - Unreserves a device

Example output and descriptions of these commands can be found in the MARIAH CDreader Console Client Specification. The following section describes the booting sequence for a VMS system on a Calypso/Mariah CPU.

2.3.1 BOOTING OPERATIONS AND SYNTAX

An overview of VMS booting is described in this section.

Booting consists of the user typing in the boot command, the console program parsing the command, and calling the boot drivers to read in the boot block program. The boot block program uses the boot drivers to read in VMB. After VMB is read in control is transferred to it and VMB reads in the next module. For VMS this module is sysboot. Sysboot continues reading in the operating system, turns on virtual mode addressing, and then transfers control to the next module that operates with virtual addressing mode on. The boot drivers prior to VMB reside in the console; VMB then switches to the boot drivers read in with it.

The actions taken to boot are:

The user enters a boot command. The syntax for the boot command is:

```
>>> b service_name
```

The optional qualifiers for this command are:

- Software boot flags, /R5:#
- Port information: /XMI:#, /BI:#

The console program parses the input and determines that the user is booting from a server. It next determines if the client's server database has any entries. If the database is empty it informs the user of this fact and issues a prompt:

```
"NO SERVICES, PERFORM SOLICIT SERVICES? (Y)"
```

The goal of the prompt is to inform the user that the database has no server entries in it, and to let the user perform a show services/solicit command to find the service.

When the database is empty the console program issues calls to the boot drivers to fill the database. Two boot drivers are used, a class boot driver and a port boot driver. The class boot driver posts read requests to the port boot driver, and then requests the port boot driver to write a solicit message over the wire. Responses from the solicit message are received and loaded into memory.

If and when the database is not empty, the console searches the database for an entry matching the service name. The entries in the database are referred to as "Available Target Identifiers", ATI's. If a matching entry is not found, the console sends a message back to the user indicating "no such service". The user then has the option to perform another show service command to find another server to boot from. When a matching ATI entry is found in the database, the console moves the ATI to a portion of memory that it has reserved for it. This portion of memory is located before VMB and is not used or overwritten by VMB. The console then loads the necessary information from the ATI into the calling interface locations. The register R3, usually denoting the device unit number, is used to point to the Available Target Identifier. The console program picks up the destination Ethernet address, port information (XMI,BI), boot device type, and the served-disk name.

The console program now wants to read in the boot block, LBN 0, sized at 512 bytes. To do this it calls the console class driver with a request to set up a circuit to the served-disk. The console class driver creates a start message and hands it off to the Ethernet port driver to send. The Ethernet port driver sends a message, waits for a response, and returns to the console class driver with either the response, or a status indicating that no response has occurred in the time allowed for a server to respond. On all failure cases, bad status is returned to the console program which is responsible for outputting failure messages to the console device. Upon successful return status from the Ethernet port drivers sending of the start message, the console class driver gives a connection request message to the Ethernet port driver to send. The Ethernet port driver sends the message and waits for response or a time expiration. Upon success, the console class driver returns to the console program for the next sequence in the booting operation.

The console program now begins the reading phase. It calls the console class driver with a logical block read request, staring at block 0, with a size of 512 bytes. As a result the console class driver creates a LAST/LAD read message. It then calls the Ethernet port driver to sends this message to the served-disk and waits for a response. The target server handles the request and sends data back to the client. The client Ethernet driver receives the blocks of data it has requested and returns to the console class driver. The console class driver strips off the overhead and loads/reloads only the 512 bytes requested into the location requested.

After the boot block has been read in the console program sets up the calling interface to the boot block program. It sets up any registers necessary for the boot block program to call back into the console class boot driver. Control is then transferred to the boot block program.

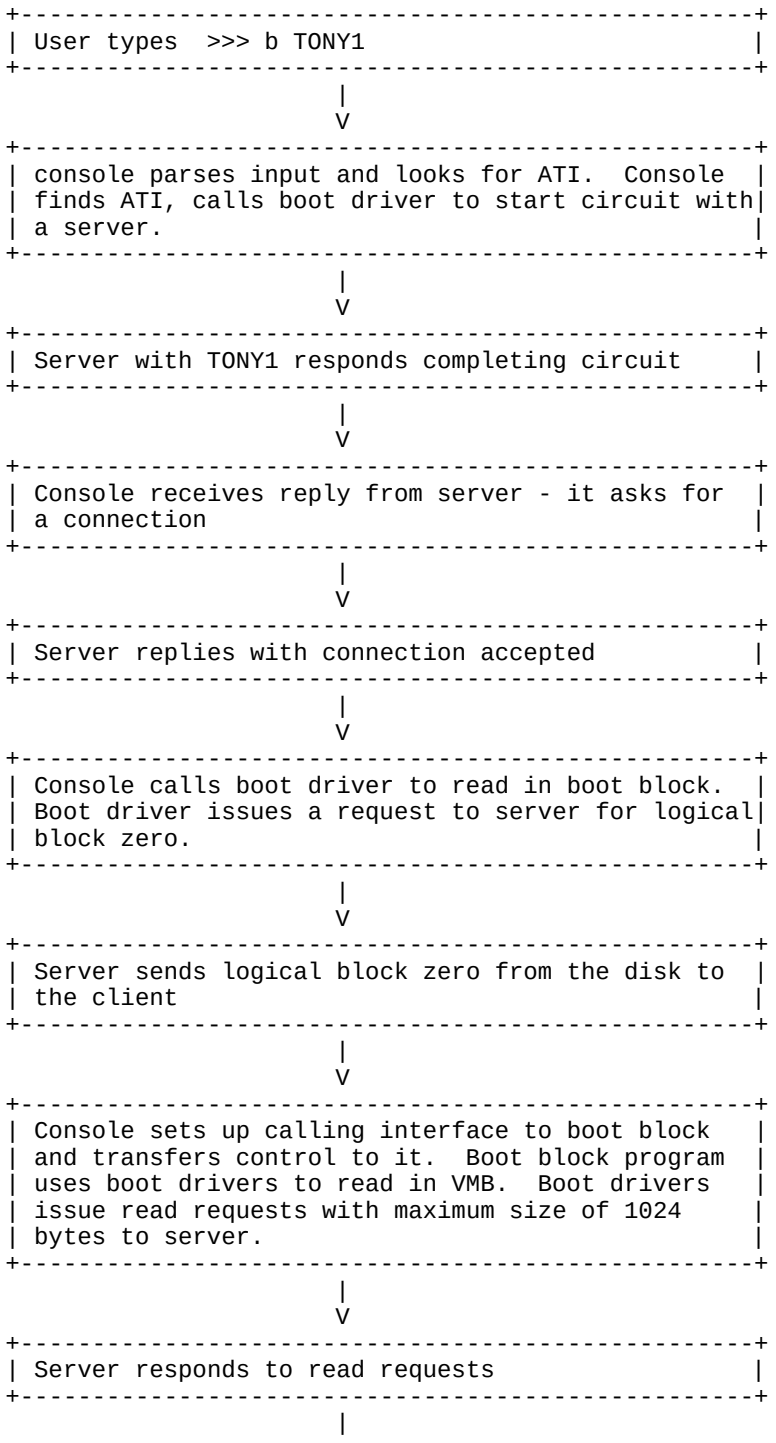
The Boot block program continues to use the console Ethernet boot driver, and the class boot driver in the console until VMB has been read in. VMB is read in by the boot block program issuing read requests to the class boot driver. The console class driver issues read packets, with the restriction that the maximum number of blocks to be read in one read request to the server through the Ethernet boot driver is 1024 bytes. This restriction is necessary to avoid oversizes Ethernet packets. Since boot drivers only have one read request outstanding, there is no need for concern of receiving packets out of order.

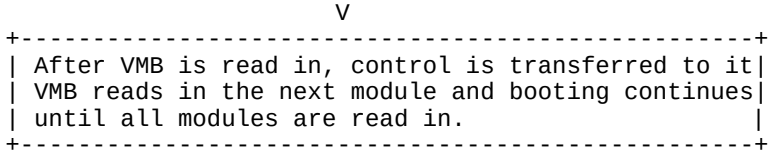
Once VMB has been read in the console sets up the calling interface to VMB and control is transferred to VMB. VMB contains Ethernet boot drivers and its version of the LAST/LAD protocol boot driver. This VMB class boot driver supports read requests to the served-disk. It handles maximum transfer sizes of 1024. So when it receives a request from VMB for a transfer of a larger size, the VMB class boot driver must break the request up into multiple

requests, and issues those requests one at a time, waiting until the data from the previous request has been received.

The boot drivers in VMB are finished with their task once all initial images are read in, IPL is lowered to an operating system level, and an operating system exists that has support for operating system device drivers.

Below is a flow of code for the operations just described.





2.3.2 VMB and other Elements

VMB determines what device to use for booting from the boot device type symbol, and then relocates the boot driver associated with this device. All of these actions are taken under the assumption that VMB is interacting with one driver. The device types supported for the Ethernet CDreader are:

```
BTD$K_SERVER_DEBNA
BTD$K_SERVER_DEBNI
BTD$K_SERVER_XNA
```

The port/class drivers are two different drivers that act as one driver. There is only one class driver for Ethernet CDreader but there are possibly multiple types of port drivers. VMB issues calls to the class driver only, never to the port driver. VMB finds the class driver through the boot device type symbol; valid boot device type symbols consists of the string "SERVER_" concatenated with the appropriate Ethernet controller designator. The boot device type symbol is used by the class boot driver to determine which port driver it must use. Because of this port/class boot driver architecture it is necessary for VMB to relocate both the port and class boot driver; however, the VMB code is implicitly performing this action. VMB performs the relocation of both the port and class boot drivers by calling the "action" routine of the class boot driver. The action routine does the following:

- determines which port driver to use,
- finds the location of the port driver,
- relocates the port driver behind the class driver,
- increases the class driver size to include the port driver,
- zeros out the address of its own action routine in the device table,
- finds the boot driver table for the port driver and stores the offsets to those entry points. (The class driver uses the offsets into the port driver's routines when it calls into the port driver.)

VMB will now relocate both the class driver and port driver as if it were one driver.

The class driver is responsible for creating request messages in the form of Ethernet packets and delivering those packets to the Ethernet boot driver. The class driver is also responsible for receiving packets, unpacking them, and relocating the blocks of data received to the correct location in memory. The Ethernet boot driver is responsible for sending and receiving packets over the wire to and from the destination server. The class driver's initilization routine calls the initialization sequence for Ethernet driver. The class driver must set up all appropriate registers and fields in the calling interface to the Ethernet driver. The Ethernet driver must return to the class driver with a status value.

After initialization, calls are made to the VMB class boot driver through the BOO$QIO calling interface. For each request to read a block the actions below take place. The VMB class boot driver:

- creates a packet to send to the server
- requests the Ethernet boot driver send that packet
- waits for status
- acts upon status
- If status was a success it issues a read request to the Ethernet boot driver
- If status was a failure it exits

The VMB class boot driver continues to issue reads until all packets for a single request have been read. When all requests have been read they are processed and returned to the caller.

2.4 VMS CDreader Client Operational Description

Once the Ethernet CDreader is installed, a booted VMS system which supports access to the Ethernet CDreader can then load the VMS CDreader Client software. This software will provide:

— a VMB transport class boot driver, a VMB Ethernet port boot driver and modified VMS booting code to allow booting to occur from the Ethernet CDreader

— the system manager a means to load, connect and start the LAD/LAST drivers used to communicate with the Ethernet CDreader

— the system manager a means to stop all LAD/LAST communications

— the system manager the means to create a large media standalone BACKUP kit which contains the code necessary to configure and access Ethernet CDreaders

— any user a means to display the version number of the software running on the Ethernet CDreader and the version numbers of the relevant parts in the VMS CDreader Client software

— any user a means to list the Ethernet CDreaders out on the network
SHOW SERVERS

— any user a means to list all the devices on any Ethernet CDreader
SHOW SERVER/DEVICES

— any user a means to find and list all services
SHOW SERVICES

— any user a means to allocate or deallocate a CD drive as a private or shared volume, if the device is not already allocated or in use

— any user a means to bind a remote device to a local device name

— any user the capability to MOUNT and DISMOUNT the media located in the remote CD drive

— any user the capability to perform DIRECTORY, COPY, BACKUP and execute VMSINSTAL.COM using a CD drive on the Ethernet CDreader as the source device