# Digital Network Appliance Platform Binding to:

## IEEE 1275-1994

## Standard for Boot

## (Initialization, Configuration)

## Firmware

Revision: 0.2 DRAFT

Date: January 26, 1998

PRELIMINARY

# Purpose of the Digital Network Appliance Platform Binding

This document specifies the application of *IEEE Std 1275-1994 Standard for Boot (Initialization, Configuration) Firmware, Core Practices and Requirements* to Digital Network Appliance Platform compliant computer systems, including practices for client program interface and data formats. An implementation of Open Firmware for a Digital Network Appliance compliant system *shall* implement the core requirements as defined in [1], the ARM processor-specific extensions described in [2] and the Digital Network Appliance specific extensions described in this binding.

# Task Group Members

The Digital Network Appliance Platform Binding Team Members were the following:

Mitch Bradley, FirmWorks

David Chaiken, Digital Equipment Corporation

Greg Hill (editor), FirmWorks

Jay Kistler, Digital Equipment Corporation

# Trademarks

The following terms, denoted by a registration symbol (®) or trademark symbol(™) on the first occurrence in this publication, are registered trademarks or trademarks of the companies as shown in the list below:

| **Trademark** | **Company** |
|---|---|
| Ethernet™ | Xerox |
| Microsoft® | Microsoft Corporation |
| MS-DOS® | Microsoft Corporation |

# Revision History

| | | |
|---|---|---|
| **Revision 0.2 DRAFT** | **July 3, 1997** | Initial, unapproved release. |
| **Revision 0.2 DRAFT** | **January 26, 1998** | Changed name to “Digital Network Appliance Platform Binding”. |

# Table of Contents

# 1. Overview

This document specifies the application of *IEEE Std 1275-1994 Standard for Boot (Initialization, Configuration) Firmware, Core Practices and Requirements* to Digital Network Appliance compliant computer systems, including practices for client program interface and data formats. An implementation of Open Firmware for a Digital Network Appliance compliant system *shall* implement the core requirements as defined in [1], the ARM processor-specific extensions described in [2] and the Digital Network Appliance specific extensions described in this binding.

The document defines the binding to ARM platforms that use 32-bit addressing. Since the minimum cell size of Open Firmware is 32 bits, only one cell is necessary to represent addresses of processor bus devices; hence the value of `"#address-cells"` for `/` *shall* be 1.

# 2. References and Terms

## 2.1. References

This standard *shall* be used in conjunction with the following publications. When the following standards are superseded by an approved revision, the revision *shall* apply.

[1] *IEEE Std 1275-1994 Standard for Boot (Initialization, Configuration) Firmware, Core Practices and Requirements*.

[2] *ARM processor binding to: IEEE Std 1275-1994, Standard for Boot (Initialization, Configuration) Firmware*.

[3] *MS-DOS® Programmer's Reference*, published by Microsoft®. This document describes the MS-DOS partition, directory and FAT formats used by the `"disk-label"` support package.

[4] a.out file format as defined in the file `include/sys/exec_aout.h` of the NetBSD distribution. http://www.netbsd.org.

[5] Croft, B., and J. Gilmore, *Bootstrap Protocol (BOOTP)*, RFC 951, Stanford University and Sun Microsystems, September 1985, http://ds.internic.net/rfc/rfc951.txt

[6] Deutsch, P., *DEFLATE Compressed Data Format Specification, Version 1.3*, RFC 1951, Aladdin Enterprises, May 1996, htttp://ds.internic.net/rfc/rfc1951.txt

[7] Deutsch, P., *GZIP File Format Specification, Version 4.3*, RFC 1952, Aladdin Enterprises, May 1996, http://ds.internic.net/rfc/rfc1952.txt

[8] Droms, R., *Dynamic Host Configuration Protocol*, RFC 1531, Bucknell University, October 1993, http://ds.internic.net/rfc/rfc1531.txt

[9] Wimer, W., *Clarifications and Extensions for the Bootstrap Protocol*, RFC 1532, Carnegie Mellon University, October 1993, http://ds.internic.net/rfc/rfc1532.txt

[10] *ISO-9660, Information processing -- Volume and file structure of CD-ROM for information interchange*, published by International Organization for Standardization.

[11] *Device Support Extensions to IEEE 1275-1994 Standard for Boot (Initialization, Configuration) Firmware*.

[12] *Open Firmware Recommended Practice: 16-color Text Extension*.

[13] *Open Firmware Recommended Practice: Graphics Extension*.

[14] *Open Firmware Recommended Practice: TFTP Booting Extension, Version 0.8* (Unapproved Draft).

## 2.2. Terms

This standard uses technical terms as they are defined in the documents cited in "References", plus the following terms:

**a.out**: A binary object file format defined in [4] that is used to represent *client programs* in Open Firmware for ARM.

**core, core specification**: refers to *IEEE Std 1275-1994 Standard for Boot (Initialization, Configuration) Firmware, Core Practices and Requirements*

**FDISK:** Refers to the boot-record and partition table format used by MS-DOS, as defined in [3].

**Open Firmware**: The firmware architecture defined by the core specification or, when used as an adjective, a software component compliant with the core specification.

# 3. Packages

This section describes the Digital Network Appliance-specific requirements of Open Firmware packages.

## 3.1. "disk-label" Support Package

The Digital Network Appliance is primarily intended to support diskless systems that store most of their data on network servers. Consequently, many implementations of this architecture lack disk-like mass storage devices (e.g. disks, diskettes, CD-ROMs). Therefore, this binding does not mandate the presence of a "`disk-label`" support package, nor does it specify the exact behavior of a "`disk-label`" support package, should it exist. For variants that do support disk devices, the behavior of the "`disk-label`" support package should be specified in additional documentation pertaining to the platform in its application domain.

## 3.2. Program-image Formats.

### 3.2.1. Load Address

The default load address is 0xF0000000, the value of **`load-base`**. Client programs are assumed to be designed to be loaded at 0xF0000000.

Prior to the first execution of **`load`**, the firmware *shall* allocate and map at least 6 MB of physical memory at this address, unless the hardware configuration of the system makes this impossible. In that case, the firmware *shall* map as much memory as practical.

### 3.2.2. Load Image Formats

The **`load`** User Interface command in conjunction with **`init-program`** *shall* prepare the loaded image for later execution according to the following algorithm:

> If the three bytes beginning at the default load address are the gzip compression signature (i.e. Byte 0 = 0x1F, Byte 1 = 0x8B, Byte 2 = 0x08) as described in [7], skip the header according to [7] and inflate the image according to the algorithm described in [6], placing the inflated image at the default load address, and proceed with the following steps.
>
> If the 4 bytes beginning at the default load address are the a.out signature as described in Section 3.2.3, prepare the image for execution as described in Section 3.2.3 and skip the remaining steps.
>
> Otherwise, if the 2 bytes beginning at the default load address are the Forth source code signature characters "\ " (i.e. 0x5C, 0x20), arrange for the next execution of **`go`** to interpret the load image as Forth source code, and skip the remaining steps.
>
> Otherwise, if the byte at the default load address is the **`start1`** FCode token (i.e. 0xF1), arrange for the next execution of **`go`** to evaluate the load image as FCode (e.g. as with **`"load-base 1 byte-load"`**), and skip the remaining steps.

**Note: At this point, the firmware may recognize and prepare other image formats not specified here.**

`Otherwise (i.e. if the load image format has not been recognized explicitly), prepare the image for execution as described in Section 3.2.4.`

### 3.2.3. `a.out` Format

The offsets given below are from the default load address.

| Offset | Name | Endianess | Contents |
|---|---|---|---|
| 0 | `a_midmag` | Big | Signature: 0x008F010B |
| 4 | `a_text` | Little | The length in bytes of the header plus the text segment in both the file and the execution image. |
| 8 | `a_data` | Little | The length in bytes of the data segment in both the file and the execution image |
| 12 | `a_bss` | Little | The length in bytes of the bss segment in the execution image. The bss segment is not stored in the file, because its initial contents are always zero. |
| 16 | `a_sym` | Little | The length in bytes of the symbols portion of the file. (The symbol table in the execution image consists of two portions, one for the symbols and a second for the strings.[1]) |
| 20 | `a_entry` | Little | The virtual address at which program execution is to begin. |
| 24 | `a_trsize` | Little | The size of the text relocation table. Used only for object files. Contains 0 for executable files. |
| 28 | `a_drsize` | Little | The size of the data relocation table. Used only for object files. Contains 0 for executable files. |

The loaded image consists of the above header immediately followed in order by the text segment, the data segment, the symbols portion of the symbol table and the strings portion of the symbol table. After recognizing this header, **`load`** *shall*:

- Synchronize the instruction and data caches from **`load-base`** to **`load-base`** + `a_text` + `a_data` + 0x20,
- Move the symbol portion and string portion of the symbol table from **`load-base`** + `a_text` + `a_data` + 0x20 to **`load-base`** + `a_text` + `a_data` + `a_bss` + 0x20.
- Zero `a_bss` bytes of memory beginning at **`load-base`** + `a_text` + `a_data` + 0x20,
- Release and unmap the physical memory from **`load-base`** + `a_text` + `a_data` + `a_bss` + `a_sym` + `string_size`[1] + 0x20 (i.e. from the end of the prepared client program memory image) to the end of the load area. (The goal of this step is to have the `"available"` properties in the `/memory` and `/mmu` nodes accurately reflect the memory actually consumed by the client program prepared image.)
- Set the **`pc`** in the *saved-program-state* to `a_entry`.

[1] `string_size` is the 32-bit, little-endian value at **`load-base`** + `a_text` + `a_data` + `a_sym` which describes the length in bytes of the string portion of the symbol table.

- Set the remaining elements of the *saved-program-state* to their initial values as defined in Section 6.2.1 of [2].

**Note: The above header is that used by NetBSD [4].**

### 3.2.4. Raw Binary Format

In order to prepare a raw binary image for execution, **`load`** *shall*:

- Synchronize the instruction and data caches from **`load-base`** to **`load-base`** + the size of the loaded image,
- Release and unmap the physical memory from **`load-base`** + the size of the loaded image (i.e. from the end of the loaded image) to the end of the load area. (The goal of this step is to have the `"available"` properties in the `/memory` and `/mmu` nodes accurately reflect the memory actually consumed by the client program prepared image.)
- Set the **`pc`** in the *saved-program-state* to **`load-base`**.
- Set the remaining elements of the *saved-program-state* to their initial values as defined in Section 6.2.1 of [2].

## 3.3. `"obp-tftp"` Support Package

The `"obp-tftp"` Support Package *shall* comply with the requirements set forth in Sections 5.1 and 5.2 of [14] with the following extensions.

### 3.3.1. Use of BOOTP

BOOTP (in its DHCP variation as described below) *shall* be the default discovery protocol (i.e. if the first argument to the `"obp-tftp"` **`open`** method is not an IP address BOOTP *shall* be used).

### 3.3.2. DHCP Support

In the `"obp-tftp"` **`load`** method, if the BOOTP protocol is to be used to discover the server's IP address, the DHCP variant of the BOOTP protocol *shall* be used, beginning at the DHCP INIT state as specified in [8]. In addition to the DHCP options required by [8], the DHCPDISCOVER and DHCPREQUEST packets shall contain a "client identifier" option whose value is derived from the `"system-id"` property of the device tree root node, and if the device tree root node contains an `"architecture"` property (whose value is a text string encoded as with `encode-string`), the DHCPDISCOVER and DHCPREQUEST packets shall contain a "vendor class identifier" option whose value is the string decoded from the prop-encoded-array of that `"architecture"` property's value.

**Note: For this binding, the `"system-id"` property must be derived from `system-mac-address`, so the "client identifier" option's value is derived from the hardware address.**

If the server responds to the DHCP `DISCOVER` with a `BOOTREPLY` that is not a `DHCPOFFER` (i.e. the options field is empty and does not contain DHCP options), Open Firmware *shall* proceed using the BOOTP protocol. Otherwise (i.e. if the server responds with a DHCPOFFER BOOTREPLY), Open Firmware shall proceed using the DHCP protocol, until the DCHP client (i.e. the firmware) enters the DHCP BOUND state, at which point the firmware has acquired an IP address for its subsequent use.

Upon successful completion of the discovery protocol, via either the full DHCP process or the truncated "BOOTP server responds with non-DHCPOFFER" version, the verbatim contents of the final BOOTP or DHCPACK packet from the server shall be reported by encoding those contents as with `encode-bytes` and using the resulting *prop-encoded-array* to create both a `"bootreply-packet"` property and a `"bootp-response"` property in the `/chosen` node. Client programs should use the `"bootreply-packet"` property; the `"bootp-response"` property is a concession to existing client programs.

The firmware may, but need not, implement an additional mechanism for remembering a previously-assigned IP address and attempting to re-use that address by skipping the DHCPDISCOVER phase and proceeding directly to the DHCPREQUEST phase, as permitted in [8]. The details of such a mechanism are not specified by this binding.

# 4. Properties

This section describes the standard properties of Digital Network Appliance Platform Open Firmware implementation.

## 4.1. Root Node Properties

The following properties of the root node (“*/*”) *shall* be created by an Open Firmware implementation.

**Note: The root node typically corresponds to the VL bus in a Digital Network Appliance system**

**`"#address-cells"`**

Standard property, encoded as with `encode-int`, that specifies the number of cells required to represent physical addresses on the processor bus. The value of `"#address-cells"` for the processor bus *shall* be 1.

**`"clock-frequency"`**

Standard property, encoded as with `encode-int`, that represents the primary system bus speed (in hertz).

**`"dma-ranges"`**

Standard property, whose value is an array of entries of the form: *child-address child-size.*

Each entry describes a contiguous DMA address range, giving the base addresses of the range in the child address space and the length of the range, which is expressed in the child's size format. (The “child” address space is the address space defined by the root node.)

Each *child-address* is a physical address, encoded as with `encode-phys`, within the child address space. The number of cells therein can be determined from the `"#address-cells"` property of the root node.

Each *child-size* is a sequence of integers, each encoded as with `encode-int`. The number of integers is determined from the `"#size-cells"` property of the root node, or by the default value of 1 if the root node contains no `"#size-cells"` property.

**Note: The general form of a `"dma-ranges"` entry is *child-address parent-address child-size*. Within this node, the *parent-address* portion is zero cells long.**

**`"system-id"`**

Standard property, encoded as with `encode-string`, that contains the identification of the computer system. This string *shall* be unique across all systems and all manufacturers. The system-id *shall* be of the form "0nnnnnnmmmmmm" where nnnnnn is a sequence of 6 uppercase hexadecimal digits representing a 24-bit Organizationally Unique Identifier (OUI) assigned by the IEEE Registration Authority Committee, and mmmmmm is a sequence of 6 uppercase hexadecimal digits representing a 24-bit binary number assigned by the manufacturer to assure uniqueness.

The 6-byte MAC address returned by **`system-mac-address`** meets the above requirements. For this binding, that MAC address, encoded in the form specified above, shall be used as the system-id.

## 4.2. `"/chosen"` Node Properties

The following properties of the `"/chosen"` node *shall* be created by an Open Firmware implementation.

**`"clock"`**

Standard property, encoded as with `encode-int`, that represents the ihandle of an instance of the real-time clock node. The purpose of this property is to enable a client program to use the `set-time` method of the `"/isa/rtc"` node easily.

**`"nvram"`**

Standard property, encoded as with `encode-int`, that represents the ihandle of an instance of the NVRAM node.

**Note: The nvram node identified in the `/chosen` node shall support a `size` method as specified in Section 7.2 of [11]. The `size` method will return a value that is the total DNA platform NVRAM size.**

## 4.3. "/isa" Node Properties

The following properties of the "/isa" node *shall* be created by an Open Firmware implementation.

**"dma-ranges"**

Standard property, whose value is an array of entries of the form: *child-address parent-address child-size*

Each entry describes a contiguous DMA address range, giving the base addresses of the range in both the child and parent address spaces and the length of the range, which is expressed in the child's size format. (The "child" address space is the address space defined by the /isa node. The "parent" address space is the address space defined by the root node, the /isa node's parent.)

Each *child-address* is a physical address, encoded as with encode-phys, within the child address space. The number of cells used is 2 reflecting the default value of the "#address-cells" property since the /isa node contains no "#address-cells" property.

Each *parent-address* is a physical address, encoded as with encode-phys, within the parent address space. The number of cells therein is determined from the "#address-cells" property of the root node, the /isa node's parent.

Each *child-size* is a single integer encoded as with encode-int. This reflects the default value of the "#size-cells" property since the /isa node contains no "#size-cells" property.

## 4.4. "/isa/rtc" Node Properties

The following properties of the "/isa/rtc" node *shall* be created by an Open Firmware implementation.

**"status"**

Standard property, encoded as with encode-string, that specifies the state of the latched and unlatched battery bits of the real-time clock. If either of those bits indicates a battery problem, the "status" property value *shall* be set to the string "bad battery". In all other cases, the property value *shall* be set to the string "okay".

**Note: The "latched battery bits" are those bits found at 0xE of the RTC chip. The "unlatched battery bits" are those found at 0xD of the RTC chip.**

**Note: Other values may be defined for this property in the future.**

## 4.5. "/openprom" Node Properties

The following properties of the "/isa/rtc" node *shall* be created by an Open Firmware implementation.

**"built-on"**

Standard property, encoded as with encode-int, that specifies the year, month and day on which the ROM image was built. The unencoded integer consists of 8 decimal digits ordered from left to right as four digits for the year, two digits for the month, and two digits for the day (e.g. 19970610 for firmware built on June 10, 1997).

# 5. Extensions for Digital Network Appliance Platform Systems

This section describes the properties, methods, and device subtrees that are applicable to devices required by the Digital Network Appliance Platform architecture. It is strongly recommended that other platforms follow these definitions for the corresponding devices.

## 5.1. Display Devices

Display device packages (i.e. device_type = "display") for Digital Network Appliance Platform systems should include in their implementation all the properties and methods called for in [12] and [13].

Display device package *shall* also include the following method:

**`text-mode3 ( -- )`**

Changes the display device's operating mode to "EGA/VGA text mode 3".

## 5.2. Device Support

Open Firmware implementations for Digital Network Appliance Platform systems must implement those device types from [11] that are appropriate to the hardware present.

## 5.3. Conventions for Devices on ISA

This section defines the naming and device type conventions for typical devices on ISA buses. The following lists are the values of the `"name"` and `"device_type"` properties of the devices on an ISA bus:

```
name              device_type
8042
     kbd          "keyboard"
     mouse        "mouse"
floppy            "block"
com               "serial"
timer             "timer"
lpt               "parallel"
ide               "block"
nvram             "nvram"
rtc               "rtc"
```

**Note: The "kbd" and "mouse" names are indented to show that they are the child nodes of the 8042 node.**

Some systems use an I/O controller, often called a super I/O chip, which provides control functions of multiple I/O devices. When a system uses a super I/O chip, a device node representing the super I/O chip itself need not exist. Instead, the device nodes of the devices attached to the super I/O chip may be direct children of the bus node representing the bus to which the super I/O chip is attached.

It is strongly recommended that the `"compatible"` property be implemented for ISA bus devices to help operating systems find appropriate device drivers for these devices.

## 5.4. `"/aliases"` Node Properties

An implementation of Open Firmware for the Digital Network Appliance Platform *shall* provide the following aliases under the `"/aliases"` node if an underlying hardware device is present:

```
audio
cdrom
com1     The serial port at I/O address 0x3F8
com2     The serial port at I/O address 0x2F8
disk
floppy
keyboard
net
rom
screen
scsi
tape
```