Open Firmware

Recommended Practice:

Interposition

Version 0.2 (draft)

July 31, 1995 10:57 pm

Published by the Open Firmware Working Group

This document is a voluntary-use recommended practice of the Open Firmware Working Group. The Open Firmware Working Group is an ad hoc committee composed of individuals interested in Open Firmware as defined by IEEE 1275-1994, related standards, and their application to various computer systems.

The Open Firmware Working Group is involved both in IEEE sanctioned standards activities, whose final results are published by IEEE, and in informal recommendations such as this, which are published on the Internet at:

`http://playground.sun.com/pub/1275`.

Membership in the Open Firmware Working Group is open to all interested parties. The working group meets at regular intervals at various locations. For more information send email to:

`p1275-wg@prombo.eng.sun.com`.

Revision History

0.1 - Created from proposal 272.

0.2 - Corrected the "http:" address above.
Fixed typos in sections 4.1 ("a5" should be "b5") and 5.2 (misspelled word).

1. Introduction

Interposition is a system ROM implementation technique that provides a modular way to layer file access capabilities on top of unmodified Open Firmware device drivers.

1.1. Purpose

Historically, Open Firmware implementations have provided "raw" access to partitioned disk devices. Recently, however, it has become desireable to extend the semantics of disk devices to include access to named files. The interposition technique is a way of adding file access capabilities, supporting many different file system structures, to an Open Firmware system, without changing existing Open Firmware disk device drivers.

1.2. Scope

Interposition is intended as an internal implementation technique for Open Firmware system ROMs. Its effects are intended to be invisible to the external interfaces defined by IEEE Std 1275-1994, except for the addition of a new client interface service to allows client programs that are cognizant of interposition to inquire about the presence of interposition in particular instance chains. Existing client programs are expected to be unaffected.

2. References and Definitions

2.1. References

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

2.2. Definitions

`core document`: The standard cited in reference [1].

3. FCode Functions and User Interface Commands

A system that implements interposition and the Open Firmware Device Interface shall implement the following FCode function.

A system that implements interposition and the FCode Debugging Command Group of the Open Firmware User Interface shall implement the following function as a User Interface command.

`interpose` (adr len phandle --) F 0x12b

Schedule a package for interposition

Schedule the package identified by 'phandle' for interposition, with the string 'adr len' as its arguments.

If a package is currently scheduled for interposition when "interpose" is executed, the result is undefined (in other words, a system implementation need not support multiple simultaneous interposition attempts).

Usage Restriction: This function must be executed only during the creation of an instance chain, i.e. during the execution of a package's "open" method during pathname resolution in "open-dev" context, as in clauses (f2), (k1iii) and (m2) of section 4.3.1 of the core document.

4. Modification to the Pathname Resolution Algorithm

A system that implements interposition shall implement the following modifications to the core document's pathname resolution algorithm (clause 4.3.1 of the core document).

4.1. Interposition Mechanism

This modification creates the mechanism by which interposition occurs as a result of the execution of **interpose**.

Create the following additional procedure for the pathname resolution process:

4.3.7 Handle interposers procedure

a) If no package is currently scheduled for interposition

1) Exit this procedure

b) Otherwise (i.e. a package is currently scheduled for interposition)

1) Unschedule that package, so that it is no longer scheduled for interposition

2) Save the specification of the active package for later restoration

3) Set the active package to the package that was scheduled for interposition.

4) Set ARGUMENTS to the string that was scheduled for that interposition.

5) Create a new linked instance using the **4.3.2 Create new linked instance procedure**2.

6) Execute the node's open method

7) Restore the active package to the value saved in step 2)

8) Go back to step a)

After 4.3.1 f2), add:

3) Handle possible interpositions using the **4.3.7 Handle interposers** procedure.

After 4.3.1 k1iii), add:

iv) Handle possible interpositions using the **4.3.7 Handle interposers** procedure.

After 4.3.1 m2), add:

2') Handle possible interpositions using the **4.3.7 Handle interposers** procedure.

Modify 4.3.1 m5) to read:

5) Exit from this procedure, returning either the 'ihandle' of the instance created in step m1) or, if any interpositions were handled in step 2'), the 'ihandle' of the instance created in the last execution of step b5) of the **4.3.7 Handle interposers** procedure.

4.2. User-specified Interposition

This modification creates an additional way of causing "manual" interposition by including a "%<package_name" component in a path name.

After 4.3.1 i), add:

i') If NODE_NAME begins with the character '%'

1) Remove the leading '%' from NODE_NAME

2) Search for a matching package among the direct children of the "/packages" node, using the **4.3.6 Node Name Match criteria**. If a match is found:

i) Save the specification of the active package for later restoration

ii) Set the active package to the package that was scheduled for interposition.

iii) Create a new linked instance using the **4.3.2 Create new linked instance procedure.**

iv) Execute the node's open method

v) Restore the active package to the value saved in step a)

vi) Go back to step g)

3) Otherwise (i.e. if no match is found), go to step l)

5. Client Services

5.1. Existing client services

A system that implements interposition and the Open Firmware Client Interface shall conform to the following requirement:

The names of packages interposed either via the use of **interpose** or via the "%" pathname syntax shall not appear in the pathname returned by the `instance-to-path` client service.

5.2. New client services

A system that implements interposition and the Open Firmware Client Interface shall implement the following additional client service:

`instance-to-interposed-path`

`IN: ihandle, [address] buf, buflen`

`OUT: length`

This service returns the fully-qualified pathname, including the names and arguments of interposed packages, corresponding to the identifier *ihandle*, storing at most *buflen* bytes as a null-terminated string in the memory buffer starting at the address *buf*. The names of interposed packages shall begin with the character '%'. If the length of the null-terminated pathname is greater than *buflen*, the trailing characters and the null termiator are not stored. *Length* is the length of the fully-qualified pathname excluding any null terminator, or -1 if *ihandle* is invalid.