*Universal Serial Bus
Device Class Specification for
OCP Open Boot and Management Framework
Interface Consolidation Protocol*

Version 0.7
Oct 8, 2025

Editor:
Mariusz Oriol — NVIDIA Corporation
Contributors:
Janusz Jurski — Intel Corporation
Bharat Pillilli — Microsoft Corporation

Table of Content

1. License

THE UPDATED DEFAULT CONTRIBUTOR LICENSE AGREEMENT (CLA) IS OWFa 0.9. PLEASE VERIFY THE CORRECT CLA/FSA IS USED AND EXECUTED FOR THIS CONTRIBUTION.

1.1 Open Web Foundation (OWF) CLA

Contributions to this Specification are made under the terms and conditions set forth in Modified Open Web Foundation Agreement 0.9 (OWFa 0.9). (As of October 16, 2024) (“Contribution License”) by:

- Advanced Micro Devices
- ASPEED Technology, Inc.
- Intel Corporation
- Microsoft Corporation
- Nuvoton Technology Israel Ltd.
- NVIDIA

Usage of this Specification is governed by the terms and conditions set forth in Modified OWFa 0.9 Final Specification Agreement (FSA) (As of October 16, 2024) (“Specification License”).

You can review the applicable Specification License(s) referenced above by the contributors to this Specification on the OCP website at https://www.opencompute.org/contributions/templates-agreements.

For actual executed copies of either agreement, please contact OCP directly.

Notes:

> The above license does not apply to the Appendix or Appendices. The information in the Appendix or Appendices is for reference only and non-normative in nature.

NOTWITHSTANDING THE FOREGOING LICENSES, THIS SPECIFICATION IS PROVIDED BY OCP "AS IS" AND OCP EXPRESSLY DISCLAIMS ANY WARRANTIES (EXPRESS, IMPLIED, OR OTHERWISE), INCLUDING IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, FITNESS FOR A PARTICULAR PURPOSE, OR TITLE, RELATED TO THE SPECIFICATION. NOTICE IS HEREBY GIVEN, THAT OTHER RIGHTS NOT GRANTED AS SET FORTH ABOVE, INCLUDING WITHOUT LIMITATION, RIGHTS OF THIRD PARTIES WHO DID NOT EXECUTE THE ABOVE LICENSES, MAY BE IMPLICATED BY THE

IMPLEMENTATION OF OR COMPLIANCE WITH THIS SPECIFICATION. OCP IS NOT RESPONSIBLE FOR IDENTIFYING RIGHTS FOR WHICH A LICENSE MAY BE REQUIRED IN ORDER TO IMPLEMENT THIS SPECIFICATION. THE ENTIRE RISK AS TO OCP Open Boot and Management Framework – Interface Consolidation Protocol IMPLEMENTING OR OTHERWISE USING THE SPECIFICATION IS ASSUMED BY YOU. IN NO EVENT WILL OCP BE LIABLE TO YOU FOR ANY MONETARY DAMAGES WITH RESPECT TO ANY CLAIMS RELATED TO, OR ARISING OUT OF YOUR USE OF THIS SPECIFICATION, INCLUDING BUT NOT LIMITED TO ANY LIABILITY FOR LOST PROFITS OR ANY CONSEQUENTIAL, INCIDENTAL, INDIRECT, SPECIAL OR PUNITIVE DAMAGES OF ANY CHARACTER FROM ANY CAUSES OF ACTION OF ANY KIND WITH RESPECT TO THIS SPECIFICATION, WHETHER BASED ON BREACH OF CONTRACT, TORT (INCLUDING NEGLIGENCE), OR OTHERWISE, AND EVEN IF OCP HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

2. Introduction

This document describes proposed requirements and specifications for Universal Serial Bus (USB) devices that support the Open Boot and Management Framework (OBMF) Interface Consolidation Protocol (OBMF ICP) capability as defined by Opencompute.org. This specification outlines the binding spec for native USB encapsulation Universal Serial Bus Device Class Specification for OBMF ICP protocol.

2.1 Related Documents

For details to the OBMF protocol please refer to published version of OBMF specification at https://www.opencompute.org/documents/

2.2 Terms and Abbreviations

The meanings of some words have been stretched to suit the purposes of this document. These definitions are intended to clarify the discussions that follow.

OBMF	Open Boot and Management Framework as defined by opencompute.org
OBMF ICP	OCP OBMF Interface Consolidation Protocol

3. Overview

Open Boot and Management Framework Interface Consolidation Protocol is defined by OCP to consolidate boot and management interfaces to platform components. This document defines binding specification for native support of OBMF ICP over USB. OBMF over USB will use Interrupt IN, Bulk IN and Bulk OUT endpoints for exposing OBMF channels to the Host. OBMF is using class-interface descriptor and associated functional descriptor embedded within the device's normal run-time descriptors to expose class-specific interface.

4. Interface definition

OBMF Interface use standard USB device enumeration process including device descriptor, configuration descriptor, and interface descriptor exchange compatible with USB 1.1 and later host controllers.
OBMF over USB uses Application Specific bInterfaceClass = 0xFE with a unique bInterfaceSubClass = <0xXX> value assigned by USB-IF to uniquely identify OCP OBMF interfaces across all USB versions and distinguish them from other device functions. OBMF over USB uses USB transfer integrity checking through standard USB CRC mechanisms and packet validation as defined in the respective USB specification versions (1.1, or later).

4.1 USB Interface Descriptor

Offset	**Field**	**Size**	**Value**	**Description**
0	bLength	1	0x09	Size of this descriptor in bytes.
1	bDescriptorType	1	0x04	INTERFACE descriptor type.
2	bInterfaceNumber	1	Number	Number of this interface.
3	bAlternateSetting	1	0x00	Alternate setting. Must be zero.
4	bNumEndpoints	1	0x03 or 0x02	Bulk IN, Bulk OUT are mandatory Interrupt IN support is optional based on device OBMF capability
5	bInterfaceClass	1	0xFE	Application Specific Class Code
6	bInterfaceSubClass	1	to be assigned	OCP OBMF SubClass **<assigned by USB-IF>**
7	bInterfaceProtocol	1	0x01	OCP OBMF version 1.
8	iInterface	1	Index	Index of string descriptor for this interface pointing to "OCP OBMF "

4.2 USB Interface Functional Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	0x0C	Size of this descriptor in bytes.
1	bDescriptorType	1	0x24	Interface-specific/Class-specific descriptor CS_INTERFACE **<confirm with USB-IF>**
2	bDescriptorSubtype	1	0x01	Interface-specific subtype for OCP OBMF OCP_OBMF_FUNCTIONAL
3	bReserved	1	0x00	Reserved
4	wMaxWrTransferSize	2	Value	Maximum write transfer size in bytes
6	wMaxRdTransferSize	2	Value	Maximum read transfer size in bytes
8	wMaxRdInterruptSize	2	Value	Maximum read interrupt EP size in bytes. SHALL fit into single Interrupt IN transfer size. Return 0 bytes if interrupt IN endpoint is not supported by the device
10	bcdOCPOBMFVersion	2	BCD	Numeric expression identifying the version of the OCP OBMF Specification release.

4.3 USB Descriptors for OCP OBMF Interface

A USB device implementing OCP OBMF SHALL expose a minimum set of descriptors for runtime operations, including:

- A Device Descriptor with bDeviceClass=0, bDeviceSubClass=0, and bDeviceProtocol=0 indicating a USB Composite Device where individual interfaces define their respective classes;
- At least one Configuration Descriptor defining the device's operational configuration;
- Exactly one Interface Descriptor dedicated to the OCP OBMF Interface with bInterfaceClass=0xFE (Application Specific), bDeviceSubClass=TBD
- Exactly one OCP_OBMF_FUNCTIONAL descriptor specifying OBMF capabilities and transfer limitations.

4.4 OCP OBMF over USB binding

The OCP OBMF interface is dedicated solely to OBMF ICP communication. There is no additional encapsulation for Bulk IN, Bulk OUT, and Interrupt IN endpoints. The Interrupt IN endpoint is designated only to pass asynchronous messages, if supported by the device. For Interrupt IN there is no response defined. Example of Interrupt IN is GPIO state change notification or BIOS POST code.

OBMF control traffic is using channel 0 over Bulk IN and Bulk OUT endpoints. The size of the endpoints is defined by the device.

OBMF Bulk IN, Bulk OUT allows for using multiple channels concurrently. The ordering is defined within channel. In typical channel implementation the is only one outstanding message that requires response allowed.

OBMF messages over multiple packets on Bulk IN/OUT endpoints are supported using standard USB packet spanning. OBMF messages over Interrupt IN must fit into a single interrupt transaction, as these endpoints cannot span across multiple packets.

This version of OBMF ICP over USB allows for only single OBMF message for specific channel to be sent.

4.4.1 Length of the OBMF message that spans across multiple packets:

1. Bulk IN transfer ends when the device sends a short packet (<wMaxPacketSize), or the exact requested length is reached. If the payload is an exact multiple of wMaxPacketSize, the device MUST send a ZLP to terminate the current OBMF payload.
2. 2. Bulk OUT transfer ends when the host sends a short packet (<wMaxPacketSize), or the exact requested length is reached. If the payload is an exact multiple of wMaxPacketSize, the host MUST send a ZLP to terminate the current OBMF payload.
3. Interrupt IN transfer ends when each service interval the host asks for up to wMaxPacketSize bytes; the device returns 0 to wMaxPacketSize and the transfer completes.

4.4.2 Message Length Determination

For transfers spanning multiple packets, the total message length is determined by:

1. Detection of short packets (less than wMaxPacketSize)
2. Reception of Zero-Length Packets (ZLP) when payload size is exact multiple of wMaxPacketSize

4.4.3 Error Handling:

If a transfer is interrupted or incomplete as defined by the USB specification:

1. The receiving endpoint should discard partial OBMF messages
2. Subsequent transfers start with new message boundaries
3. No message fragmentation recovery is performed at the USB binding level

4.4.4 OCP OBMF over USB integrity protection

OCP OBMF over USB is using USB integrity checks. There are no additional integrity needed, unless enforced by the OBMF specification

4.4.5 Compatibility

This OBMF binding specification is compatible with USB 1.1 and higher.