Greybus Specification

Release 1.0

Google Project Ara

Dec 16, 2016

Contents

Warning: This document contains a preliminary specification for various aspects of a Greybus system's communication. It is important to note that the information contained within is in a draft stage, and has not yet been fully implemented. The specifications defined herein are **unstable**, and may change incompatibly in future versions of this document.

This document abides by Section 13.1 of the IEEE Standards Style Manual, which describes the use of the words "shall", "should", "may", and "can" in a document as follows.

- The word *shall* is used to indicate mandatory requirements strictly to be followed in order to conform to the Specification and from which no deviation is permitted (*shall* equals *is required to*).
- The use of the word *must* is deprecated and shall not be used when stating mandatory requirements; must is used only to describe unavoidable situations
- The use of the word *will* is deprecated and shall not be used when stating mandatory requirements; will is only used in statements of fact
- The word *should* is used to indicate that among several possibilities one is recommended as particularly suitable, without mentioning or excluding others; or that a certain course of action is preferred but not necessarily required; or that (in the negative form) a certain course of action is deprecated but not prohibited (*should* equals *is recommended that*).
- The word *may* is used to indicate a course of action permissible within the limits of the Specification (*may* equals *is permitted to*).
- The word *can* is used for statements of possibility and capability, whether material, physical, or casual (*can* equals *is able to*).

Unless explicitly designated informative, all sections are normative.

Chapter 1

Glossary

AP

AP Module

Application Processor Module A specially designated *Module* within a *Greybus System*.

An AP Module administers a *Greybus System* by exchanging *SVC Protocol Operation*s with the *SVC*, and *Control Protocol* Operations with *Module*s connected to *Interface*s on the *Frame*.

Bridged PHY Protocol One of the designated set of *Protocol*s which allow *Module*s to expose functionality to the Greybus System which is provided by chipsets using alternative physical interfaces than UniPro, or which do not comply with an existing *Device Class Protocol*.

Connection

Greybus Connection A Greybus Connection, or simply Connection, is a bidirectional communication path between exactly two *Interface*s.

There is a UniPro CPort at each end of a Connection; each such CPort is part of a Module or is associated with the SVC. *Module*s may exchange data on a Connection through transmission and reception of UniPro Messages.

The *AP* may establish Connections to Interfaces during *The Interface Lifecycle*. When a Connection is established, Greybus *Operation*s may be exchanged between the two users of the CPorts at either end of the Connection. The semantics for these Operations are defined by *Protocol*s in the Greybus Specification.

The *AP* may also subsequently close Connections. When a Connection is closed, Greybus Operations can no longer be exchanged between the CPort Users.

The AP also exchanges data on a Connection with the SVC.

Connection Protocol See *Protocol*.

Control Connection A *Connection* which is used to exchange *Operation*s in the *Control Protocol*.

Control CPort A UniPro CPort provided by an Interface which, under certain conditions, responds to Greybus *Operation*s in the *Control Protocol*.

CSI-2 Camera Serial Interface 2. See *CSI 2 Specifications*.

CSI-3 Camera Serial Interface 3. See *CSI 3 Specifications*.

Device Class Protocol One of the designated set of *Protocol* which allow *Module*s to expose functionality commonly found on mobile handsets to the *Greybus System*, in a manner that abstracts various hardware-specific aspects by which that functionality is implemented.

Frame A physical entity within a *Greybus System*, containing a UniPro switch, exactly one *SVC*, and a collection of *Interface*s. Each Interface may be occupied by a *Module*. A Module may occupy multiple Interfaces. Every Module exchanges UniPro Messages with other elements of a *Greybus System* by physical connection to one or more Interfaces.

FrameTime A global monotonic clock shared by all processors in the system. FrameTime is based off of a common reference clock and is synchronized using Greybus Operations and a series of *TimeSync Pulse*s. FrameTime provides a global 64 bit timestamp at a clock rate specified by the AP.

Greybus System An implementation of the Project Ara platform which complies with the Greybus Specification.

Interface An entity with a Greybus *Module* which can interact with a *Frame* via its physical connection to an *Interface Block* if the Module is attached to the Frame.

Interface Backend Firmware The Interface Backend Firmware may be required for a Module for the functioning of an entity other than the *Interface*.

Interface Block The physical connectors exposed by the *Frame* for connection to *Module*s as defined by the Project Ara *MDK*.

Interface Firmware The Interface Firmware may be required for a Module for the functioning of an *Interface*, which is responsible for exchanging Greybus Operations.

Interface Lifecycle A *state machine* which defines the changes occurring on each *Interface Block*'s *Interface State* from the time a *Module* is attached to the Interface Block until it is removed.

Interface State An abstract representation of the state of each *Interface Block* in a *Greybus System*.

MDK

Module Developers' Kit Project Ara Module Developer's Kit. This comprises various documents which collectively define the Ara platform.

Message Header The Message Header is a common data structure which occurs at offset zero of each UniPro Message containing an individual Greybus *Operation*'s *Request* or *Response*. Within the Message, the Message Header is followed by an optional payload, as defined by the *Operation*'s *Protocol*.

Module A physical entity within a *Greybus System*, which is inserted into exactly one *Slot* in a *Frame*. *Module*s exchange information with one another and with the *SVC* via UniPro Messages as defined by *[MIPI01]* and in accordance with the Greybus Specification.

Operation An abstraction defined as part of a *Protocol*. An Operation comprises an *Operation Type*, an Operation *Request* (or simply "Request"), and an Operation *Response* (or simply "Response").

Requests and Responses are UniPro Messages as defined in *[MIPI01]*; the UniPro L4 payload and semantics of each Request and Response are defined by the Greybus Specification.

Operation Type Each *Protocol* defines a set of Operation Types. Each Operation Type has a name, a Request Value, and a Response Value.

An Operation Type has a name, along with a one-byte nonzero value, from which the Operation Type's Request Value and Response Value are derived.

Each Operation Type has an associated unsigned value, which lies in the range 1 to 127 (the value 0 is invalid). Each Operation Type has a Request Value, which equals the Operation Type's value, and a Response Value, which equals the Operation Type's value logically ORed with 0x80.

For example, an Operation Type with value 0x03 has Request Value 0x03, and Response Value 0x83.

Primary Interface When a *Module* is attached to one or more *Interface Block*s in a *Slot*, exactly one such Interface Block is the *Primary Interface* to the Module.

This Interface Block shall have an Interface ID which is the lowest in value of all of the Interface Blocks attached to the Module.

An attached Module can only be ejected from a *Greybus System* via its Primary Interface. The means of ejection are implementation-defined.

Protocol A Greybus Protocol defines the layout and semantics of the *Operation*s which may be exchanged on a *Connection*.

Protocols are grouped according to their function:

- *Special Protocol*s
- *Device Class Protocol*s
- *Bridged PHY Protocol*s

Request A UniPro Message sent by a *Module* which initiates an *Operation*.

The UniPro L4 payload and semantics of each Request are specified by the *Protocol* definition of the Request's associated Operation.

Requestor Within the context of an *Operation*, the *Module* which sends or sent the *Operation*'s *Request*.

Respondent Within the context of an *Operation*, the *Module* which sends or sent the Operation's *Response*.

Response A UniPro Message which is an *Operation*.

The UniPro L4 payload and semantics of each Response are specified by the *Protocol* definition of the Response's associated Operation.

Secondary Interface When a *Module* is attached to one or more *Interface Block*s *Slot*,only one such Interface Block is the *Primary Interface* to the Module. All other such Interface Blocks are Secondary Interfaces to the Module.

These Interface Blocks, if any, have Interface IDs which are consecutive integers following the Interface ID of the Primary Interface to the Module.

*Module*s may communicate via Greybus via Secondary Interfaces, but the Module as a whole is generally identified by the Interface ID of its Primary Interface. Additionally, the Module can only be physically ejected from the *Greybus System* via its Primary Interface, through implementation-defined means.

Slot The *Interface*s in a *Frame* are physically partitioned into groups of one or more Interfaces. Each such group is called a Slot.

While each Interface in a Slot may be physically connected to at most one *Module* at any given time, a Slot with multiple Interfaces may be connected to multiple *Module*s. Additionally, a Module may be connected to multiple Interfaces, depending upon its size.

Special Protocol One of the designated set of Greybus *Protocol*s which permits discovery and enumeration of *Module*s by the *SVC*, and for other special-purpose tasks, such as network and power bus management.

SVC

Supervisory Controller An entity within the *Frame* that configures and controls the UniPro network, and controls other elements of each *Interface*.

Switch An entity within the *Frame* that allows UniPro implementations on *Module*s to communicate with one another via UniPro CPorts.

The Switch is managed directly by the *SVC*. Through the use of the *SVC Protocol*, the *AP* may request the SVC to configure the Switch in order to manage its internal state, as well as to establish *Greybus Connection*s between *Interface*s.

TimeSync Pulse An assertion and deassertion of the WAKE pin associated with an Interface Block for the purposes of communicating the FrameTime to an Interface Block. The duration of the assertion is implementation-defined but must be shorter than both the *WAKE Pulse* and the *WAKE Pulse Cold Boot Threshold* respectively.

WAKE Pulse An assertion and deassertion of the *WAKE* sub-state of an *Interface State*.

WAKE Pulse Cold Boot Threshold An implementation-defined duration in time. If a *WAKE Pulse* occurs on an *Interface State* and exceeds this duration, then any Module which is attached to the corresponding Interface Block which is capable of Greybus communications shall initialize or re-initialize itself.

Additional details are described in *WAKE*.

Chapter 2

Contributors

The following individuals made significant contributions to the development of this specification:

- Mark Greer – Animal Creek Technologies
- Alexandre Bailon – BayLibre
- Benoit Cousson – BayLibre
- Bartosz Golaszewski – BayLibre
- Axel Haslam – BayLibre
- Fabien Parent – BayLibre
- Patrick Titiano – BayLibre
- Jun Li – BSquare
- Phong Tran – BSquare
- Winnie Wang – BSquare
- Chris Myerchin – BSquare
- Jason Hung – BSquare
- Jeffrey Carlyle – Google
- Paul Eremenko – Google
- David Fishman – Google
- Greg Kielian – Google
- David Lin – Google
- Elwin Ong – Google
- Sandeep Patil – Google
- Robert C Johnson – GTG Productions
- Johan Hovold – Hovold Consulting
- Laurent Pinchart – Ideas On Board
- Marti Bolivar – LeafLabs
- Perry Hung – LeafLabs

- Michael Mogenson – LeafLabs
- Eli Sennesh – LeafLabs
- Joel Porquet – LeafLabs
- Vishal Bhoj – Linaro
- Bin Chen – Linaro
- Alex Elder – Linaro
- Thara Gopinath – Linaro
- George Grey – Linaro
- Rob Herring – Linaro
- Vaibhav Hiremath – Linaro
- Viresh Kumar – Linaro
- Khasim Mohammed – Linaro
- Jacopo Mondi – Linaro
- Sachin Pandhare – Linaro
- Satish Patel – Linaro
- Matt Porter – Linaro
- Rui Silva – Linaro
- John Stultz – Linaro
- Glen Valante – Linaro
- Pankaj Bharadiya – Linaro
- Vaibhav Agarwal – Linaro
- Greg Kroah-Hartman – Linux Solutions
- Blagovest Kolenichev – MM Solutions
- Georgi Dobrev – MM Solutions
- Jean Pihet – NewOldBits
- Bryan O'Donoghue – Nexus Software
- Ara Knaian – NK Labs, LLC
- Seth Newburg – NK Labs, LLC
- Karim Yaghmour – Opersys
- Olin Sibert – Oxford Systems
- Tony Carosa – Protocol Insight

Chapter 3

Legal Stuff

Don't Panic.

— Douglas Adams

3.1 Copyright

3.2 License

3.3 Additional IP Rights Grant (Patents)

"This specification" means the software components of the Greybus specification distributed by Google.

Google hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, transfer and otherwise run, modify, implement and propagate the contents of this specification, where such license applies only to those patent claims, both currently owned or controlled by Google and acquired in the future, licensable by Google that are necessarily infringed by this specification. This grant does not include claims that would be infringed only as a consequence of further modification of this specification or a related implementation.

If you or your agent or exclusive licensee institute or order or agree to the institution of patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that this specification or any code incorporated within this specification constitutes direct or contributory patent infringement, or inducement of patent infringement, then any patent rights granted to you under this License for this specification shall terminate as of the date such litigation is filed.

Chapter 4

Introduction (Informative)

> Good artists copy, great artists steal.
>
> — Pablo Picasso

The Greybus Specification describes a suite of communications protocols required to support the Project Ara modular cell phone platform.

The Project Ara Module Developer's Kit (MDK) is the official Project Ara platform definition; it comprises various documents which collectively define the Ara platform, including its industrial, mechanical, electrical, and software design and requirements. Refer to the main MDK document for an introduction to the platform and its components. Familiarity with this document is assumed throughout the Greybus Specification; its definitions are incorporated here by reference.

The Greybus Specification is included within the MDK; its purpose is to define software interfaces whose data and control flow cross Module boundaries. This is required to ensure software compatibility and interoperability between Modules and the *Frame*.

Project Ara utilizes the UniPro protocol for inter-Module communication. The UniPro specification is defined by the MIPI Alliance. UniPro's design follows a layered architecture, and specifies how communication shall occur up to the Application layer in the OSI model. Project Ara's architecture requires an application layer specification which can handle dynamic device insertion and removal from the system at any time and at variable locations. It also requires that existing Modules interoperate with Modules introduced at later dates. This document aims to define a suite of application layer protocols which meet these needs.

In addition to UniPro, Project Ara also specifies a small number of other interfaces between Modules and the Frame. These include a power bus, signals which enable hotplug and power management functions, and interface pins for Modules which emit and receive radio signals. The Greybus Specification also defines the behavior of the system's software with respect to these interfaces.

A Project Ara "Module" is a device that slides into a physical slot on a Project Ara Frame. The Frame has one or more "Interface Blocks." Each Interface Block is a single physical port through which UniPro packets are transferred. Modules connect one or more Interface Blocks on the Frame. Greybus represents each Interface Block with an "Interface" abstraction. A Greybus Interface can support one or more "Bundles". A Bundle represents a logical "device" in Greybus that does one logical "thing" as far as the host operating system works. Bundles communicate with each other on the network via one or more UniPro CPorts. A CPort is a bidirectional pipe through which UniPro traffic is exchanged. Bundles send "messages" via CPorts; messages are datagrams with ancillary metadata. All CPort traffic is peer-to-peer; multicast communication is not supported.

Project Ara presently requires that exactly one application processor (AP) is present on the system for storing user data and executing applications. The Module that contains the AP is the *AP Module*; the

Greybus specification defines a *Control Protocol* to allow the AP Module to accomplish its tasks.

In order to ensure interoperability between the wide array of application processors and hardware peripherals commonly available on mobile handsets, the Greybus Specification defines a suite of *Device Class Connection Protocols*, which allow for communication between the various Modules on the system, regardless of the particulars of the chipsets involved.

The main functional chipsets on Modules may communicate via a native UniPro interface or via "bridges," special-purpose ASICs which intermediate between these chipsets and the UniPro network. In order to provide a transition path for chipsets without native UniPro interfaces, the Greybus Specification defines a variety of *Bridged PHY Connection Protocols*, which allow Module developers to expose these existing protocols to the network. In addition to providing an "on-ramp" to the platform, this also allows the implementation of Modules which require communication that does not comply with a device class Protocol.

Chapter 5

Greybus Hardware Model

5.1 Overview

An implementation of the Project Ara platform which complies with the Greybus Specification is a *Greybus System*.

A Greybus System has the following physical components:

- A *Frame*, which contains at least one *Slot*. The Slots on a Frame are separated by the Frame's spine and ribs.
- A collection of Slots on the Frame, each of which contains at least one *Interface Block*. Each Interface Block contains several pins, which allow for power distribution, Module hotplug detection, time synchronization and communication between the Frame and attached Modules.
- Zero or more attached *Module*s, which mate with the Frame via one or more Interface Blocks. All Interface Blocks used by an individual Module are in the same Slot on the Frame. An Interface Block on the Frame can be used by at most one Module at a time.

The Frame contains the *SVC*, which, in collaboration with the *AP Module*, manages physical signals present on the Interface Blocks. The Frame also contains a *Switch*, which can be directly configured by the SVC.

Each Interface Block contains connections for M-PHY *[MIPI02]* LINK establishment between attached Modules and the Switch. The SVC can configure the Switch and permit communication between the Switch and attached Modules, and thereby indirectly between Modules themselves, via these LINKs. The AP is also able–as a Module–to communicate with other Modules, as well as the SVC, using these LINKs.

The following sections define abstract representations of state present in a Greybus System for use representing these components within the Greybus Specification.

- The first section, *Interface States*, defines the *Interface State* data structure. This structure represents the state of components related to an Interface Block on the Frame.

  The dynamics of a Greybus System effect changes to Interface States as defined in the remainder of this document. These changes map to Interface Block components in implementation-defined ways.
- The subsequent section, *Initial Interface States*, defines the initial values of each Interface State.
- *Interfaces* then defines the Interface, which models the entities within attached Modules that communicate with the Frame via Greybus.
- Following that, *Interface Lifecycle States* provides a state diagram which describes Interface lifetimes within a Greybus System. The states in this diagram are *Interface Lifecycle States*.

- Finally, *Special AP Module Requirements* defines special requirements related to the AP Module and the SVC.

Each Interface Block in a Greybus System is given a unique identifier, its Interface ID. Interface IDs increase consecutively, moving counter-clockwise around the Frame. The Interface State and the Interface Block it is associated with share the same Interface ID.

Subsequent definitions within the Greybus Specification define how certain Greybus *Operation*s affect Interface States and Interface Lifecycle States in a Greybus System.

5.2 Interface States

An *Interface State* is a tuple containing "sub-state" values. Each Interface State is defined by the specific values of its sub-states. Each Interface Block in a Greybus System has an associated Interface State, which represents its state within the Frame. The initial value of each Interface State is given in *Initial Interface States*. An Interface Block's Interface State is well-defined at the time a Greybus Operation's request message is transmitted or response message is received. A Greybus Operation can lead to a change to one or more sub-state values, and consequently change the Interface State associated with an Interface Block.

The names of the sub-states of each Interface State are as follows, along with an overview of their meaning within a Greybus System.

- DETECT: whether the SVC has sensed that a Module is attached to the Interface Block.
- V_SYS: whether system power is supplied from the Frame to the Interface Block.
- V_CHG: whether the Interface Block can supply power to the Frame.
- WAKE: whether the Frame is "activating" the Interface Block for communication via Greybus.
- UNIPRO: a representation of the state of the Switch components connected to the Interface Block.
- REFCLK: whether the Frame is providing a reference clock signal to the Interface Block.
- RELEASE: whether the Frame is attempting to physically eject a Module attached to the Interface Block.
- INTF_TYPE: denotes capabilities the SVC has determined related to the Interface communicating with the Interface Block.
- ORDER: If the SVC has determined the Interface Block is attached to a Module, this indicates whether the SVC has determined the Interface Block is the "*Primary Interface*" or a "*Secondary Interface*" to the Module.
- MAILBOX: the value of a special-purpose and Greybus implementation-specific UniPro DME attribute within the Switch used by Modules as a non-CPort based means of communication with the Frame.

An Interface State is written as a tuple as follows:

Value	Description
DETECT_UNKNOWN	Whether a Module is attached to the Interface Block is unknown
DETECT_INACTIVE	No Module is currently attached to the Interface Block
DETECT_ACTIVE	A Module is attached to the Interface Block

Table 5.2: DETECT sub-state values

Sub-State	Value
DETECT	<detect>
V_SYS	<v_sys>
V_CHG	<v_chg>
WAKE	<wake>
UNIPRO	<unipro>
REFCLK	<refclk>
RELEASE	<release>
INTF_TYPE	<type>
ORDER	<ord>
MAILBOX	<mbox>

Table 5.1: Interface State Tuple

Where in each case <detect>, <v_sys>, etc. are the values of the corresponding sub-states.

For brevity, the phrase "an Interface State's DETECT" is used to denote the value of the DETECT sub-state of that Interface State, and similarly for the other sub-states.

5.2.1 DETECT

The values of the DETECT sub-state are given in Table 5.2.

The DETECT sub-state of an Interface State represents the state of signals used to determine whether the Interface Block currently has a Module attached to it. This determination shall be performed by the SVC. The means by which the SVC does so are implementation-defined.

Under normal operation, a Module shall be physically removed from a Greybus System as a consequence of Operations exchanged between the AP and SVC only. However, it is possible that a Module can be physically removed from the system without intervention from the AP and SVC. This condition is a *forcible removal* of the Module; alternatively, the Module is said to have been *forcibly removed*.

If a Module attached to an Interface Block is forcibly removed, there may be an implementation-defined delay during which the DETECT sub-state of the corresponding Interface State remains DETECT_ACTIVE. Furthermore, the DETECT sub-state may become DETECT_UNKNOWN following a forcible removal. However, the SVC shall, potentially following such a delay and period during which DETECT is DETECT_UNKNOWN, determine that the DETECT sub-state is DETECT_INACTIVE.

5.2.2 V_SYS

The values of the V_SYS sub-state are given in Table 5.3.

The value of the V_SYS sub-state is set by the SVC.

The V_SYS sub-state of an Interface State represents the state of system power as supplied by the Frame to the corresponding Interface Block via the Interface Block's connection to the system power bus.

Value	Description
V_SYS_ON	The Frame supplies system power to the Interface Block
V_SYS_OFF	The Frame does not supply system power to the Interface Block

Table 5.3: V_SYS sub-state values

Value	Description
V_CHG_ON	The Interface Block may supply power to the Frame
V_CHG_OFF	The Interface Block cannot supply power to the Frame

Table 5.4: V_CHG sub-state values

Modules may draw power from Interface Blocks, depending on the V_SYS sub-state of the corresponding Interface State. A Module can only draw power from an Interface Block whose Interface State's V_SYS sub-state is V_SYS_ON.

Note that the V_SYS sub-state only indicates whether the Frame is supplying system power to the corresponding Interface Block; it does *not* imply that a Module is attached to the Interface Block.

The SVC shall set the V_SYS sub-state of any Interface States associated with a *forcibly removed* Module to V_SYS_OFF after an implementation-defined delay.

5.2.3 V_CHG

The values of the V_CHG sub-state are given in Table 5.4.

The value of the V_CHG sub-state is set by the SVC.

The V_CHG sub-state of an Interface State represents whether power may be supplied to the Frame via that Interface Block, via the Interface Block's charger power bus.

The Frame may draw power from an Interface Block, depending on the V_CHG sub-state of the corresponding Interface State. The Frame can only draw power from an Interface Block whose Interface State's V_CHG sub-state is V_CHG_ON.

Note that the V_CHG sub-state only indicates whether the Frame may draw power from the corresponding Interface Block; it does *not* imply that a Module is attached to the Interface Block.

The SVC shall set the V_CHG sub-state of any Interface States associated with a *forcibly removed* Module to V_CHG_OFF after an implementation-defined delay.

5.2.4 WAKE

The values of the WAKE sub-state are given in Table 5.5.

Value	Description
WAKE_UNSET	Wake signal is neither asserted nor deasserted
WAKE_ASSERTED	Wake signal is asserted to an Interface Block
WAKE_DEASSERTED	Wake signal is deasserted to an Interface Block

Table 5.5: WAKE sub-state values

The WAKE sub-state of an Interface State represents the state of a signal used to initialize an attached Module. The value of the WAKE sub-state is set by the SVC.

During the initialization of a Greybus System, all Interface States have WAKE equal to WAKE_UNSET. The SVC shall only set WAKE to a value other than WAKE_UNSET for an Interface State whose DETECT sub-state is DETECT_ACTIVE and V_SYS is V_SYS_ON.

Note that the WAKE sub-state only indicates whether the wake signal is asserted, deasserted, or neither to corresponding Interface Block; it does *not* imply that a Module is attached to the Interface Block.

The SVC shall set the WAKE sub-state of any Interface States associated with a *forcibly removed* Module to WAKE_UNSET after an implementation-defined delay.

WAKE Pulse

Subject to the above restrictions, the SVC may assert and deassert the WAKE sub-state by following this sequence, assuming WAKE is WAKE_UNSET.

1. Set WAKE to WAKE_ASSERTED
2. Delay for some duration
3. Set WAKE to WAKE_DEASSERTED
4. Set WAKE to WAKE_UNSET

This is called a *WAKE Pulse*. When the duration of the WAKE Pulse equals or exceeds an implementation-defined threshold, the *WAKE Pulse Cold Boot Threshold*, this is a signal to any attached Interface to initiate (or reinitiate) UniPro, and subsequently Greybus, communication, as described in later sections.

TimeSync Pulse

In addition to the restrictions described in the *Wake section*; once an Interface is in the *ENUMERATED Lifecycle State* and upon successful completion of the *Greybus SVC TimeSync Wake Pins Acquire Operation* the interpretation of the WAKE signal is re-defined as a TimeSync signal until successful completion of the *Greybus SVC TimeSync Wake Pins Release Operation*.

During the period between successful completion of a *TimeSync Wake Pins Acquire Operation* and completion of a *Greybus SVC TimeSync Wake Pins Release Operation* the SVC may toggle WAKE_ASSERTED and WAKE_DEASSERTED to an Interface Block to indicate a Greybus SVC *TimeSync Pulse* event. The SVC is required to ensure the duration of the WAKE_ASSERTED signal is sufficiently short that it cannot be misinterpreted as any type of *WAKE Pulse*.

Assuming WAKE is WAKE_UNSET:

1. Set WAKE to WAKE_ASSERTED
2. Delay for some duration less than the duration of a *WAKE Pulse*
3. Set WAKE to WAKE_DEASSERTED
4. Set WAKE to WAKE_UNSET

This is called a *TimeSync Pulse*. The duration of the *TimeSync Pulse* is implementation-defined but must be less than the implementation-defined *WAKE Pulse Cold Boot Threshold*.

Value	Description
UNIPRO_OFF	UniPro port is powered off
UNIPRO_DOWN	UniPro port is powered on, and the link is down
UNIPRO_LSS	UniPro link startup sequence is ongoing between Module and Frame
UNIPRO_UP	UniPro link is established
UNIPRO_HIBERNATE	UniPro link is in low-power hibernate state
UNIPRO_RELINK	UniPro peer is attempting to re-initiate linkup

Table 5.6: UNIPRO sub-state values

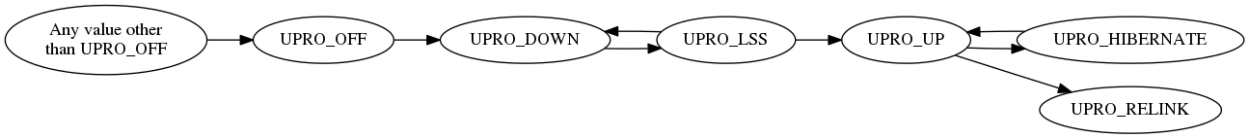
5.2.5 UNIPRO

The values of the UNIPRO sub-state are given in Table 5.6.

The UNIPRO sub-state of each Interface State represents entities within the Switch. These entities can communicate with Interfaces within Modules, and can be configured by the SVC.

Since all Greybus Protocols exchange data via UniPro Messages, each Interface Block contains the necessary signals to connect a UniPro implementation within a Module attached to that Interface Block to the Switch, which can route these Messages to other Modules, and perform some other UniPro protocol communication with attached Modules.

Transitions between successive values of the UNIPRO sub-state are shown in the following figure. All other transitions are illegal.



Greybus communication between Modules (including the AP Module) is only possible through Interface Blocks whose Interface State's UNIPRO sub-state is UNIPRO_UP: this is required to allow CPorts managed by Module Interfaces to exchange Greybus Operations via UniPro Messages. It is also necessary for *routes* within the Switch to be established to allow UniPro Messages sent by Interfaces to be relayed through the Switch to the Interfaces which are their intended recipients.

Other UNIPRO sub-state values are used primarily during communication between the SVC and AP during Module initialization, teardown, power management, and error handling, and are subject to the following constraints:

- Before a Module is first attached to an Interface Block, and during the initialization of a Greybus System, UNIPRO is either UNIPRO_OFF or UNIPRO_DOWN.
- If a Module is not attached to an Interface Block, UNIPRO cannot become UNIPRO_UP, UNIPRO_HIBERNATE, or UNIPRO_RELINK.
- The SVC can set UNIPRO to either UNIPRO_OFF (and subsequently to UNIPRO_DOWN) at any time, regardless of whether a Module is attached to the Interface Block.
- Both the SVC and any attached Module's Interface shall be notified, by implementation-specific means, if UNIPRO becomes any of the values UNIPRO_LSS, UNIPRO_UP, UNIPRO_HIBERNATE, or UNIPRO_RELINK.
- If UNIPRO is UNIPRO_DOWN, either the SVC or an attached Module's Interface may set UNIPRO to UNIPRO_LSS.
- If the SVC sets UNIPRO to UNIPRO_LSS, the attached Module's Interface may subsequently set UNIPRO to UNIPRO_UP, within a duration defined by the UniPro standard.

Value	Description
REFCLK_ON	The Frame is supplying a reference clock signal to the Interface Block
REFCLK_OFF	The Frame is not supplying a reference clock signal to the Interface Block

Table 5.7: REFCLK sub-state values

- If an attached Module's Interface sets UNIPRO to UNIPRO_LSS, the SVC may subsequently set UNIPRO to UNIPRO_UP, within the same duration.
- If UNIPRO remains UNIPRO_LSS for a duration defined by the UniPro standard, it autonomously (i.e., without the SVC or Module making the change) is set to UNIPRO_DOWN.

  When this occurs, if the SVC set UNIPRO to UNIPRO_LSS, the SVC shall be notified by implementation-specific means; similarly, if the Interface sets UNIPRO to UNIPRO_LSS, the Interface shall be notified by implementation-specific means.
- The SVC can set UNIPRO to UNIPRO_HIBERNATE.
- If UNIPRO is UNIPRO_HIBERNATE, the SVC can attempt to set UNIPRO to UNIPRO_UP.

  The SVC shall be notified whether the attempt succeeds or fails. If a Module is attached to the Interface Block, the Interface on the Module may be notified if the attempt succeeds or fails. In both cases, the notification is through implementation-specific means.
- An attached Module can, but should not, set UNIPRO to UNIPRO_HIBERNATE or UNIPRO_RELINK.
- The SVC can, but should not, set UNIPRO to UNIPRO_RELINK.

Note that the UNIPRO sub-state is a Frame-centric view of the state of entities within the Switch. Following a *forcible removal* of a Module which had established a LINK to the Frame via the corresponding Interface Block, the UNIPRO sub-state may retain its previous value or change values. This may depend upon its current value and any ongoing activity on the LINK.

5.2.6 REFCLK

The values of the REFCLK sub-state are given in Table 5.7.

The value of the REFCLK sub-state is set by the SVC.

The Frame may transmit a reference clock signal of an implementation-defined frequency to any attached Modules through the Interface Blocks the Modules are attached to. The REFCLK sub-state indicates whether this transmission is currently ongoing.

Note that the REFCLK sub-state only indicates whether the Frame is supplying a reference clock signal to the corresponding Interface Block; it does *not* imply that a Module is attached to the Interface Block.

The SVC shall set the REFCLK sub-state of any Interface States associated with a *forcibly removed* Module to REFCLK_OFF after an implementation-defined delay.

5.2.7 RELEASE

The values of the RELEASE sub-state are given in Table 5.8.

The value of the RELEASE sub-state is set by the SVC.

The Frame may physically eject any attached Modules through implementation-defined means. Any attached Module has exactly one Primary Interface, and may contain Secondary Interfaces, as described in *ORDER*.

Value	Description
RELEASE_ASSERTED	Frame is asserting ejection signal to the Interface Block
RELEASE_DEASSERTED	Frame is not asserting ejection signal to the Interface Block

Table 5.8: RELEASE sub-state values

INTF_TYPE	Value	Description
IFT_UNKNOWN	0	Module not attached, type is undetermined, or error occurred
IFT_DUMMY	1	Module attached does not support UniPro communication
IFT_UNIPRO	2	Module attached supports UniPro, but not Greybus Protocols
IFT_GREYBUS	3	Module attached supports Greybus Protocols

Table 5.9: INTF_TYPE sub-state values

The SVC may set the RELEASE sub-state of an Interface Block which is the Primary Interface to an attached Module to RELEASE_ASSERTED for an implementation-defined duration, then set RELEASE to RELEASE_DEASSERTED, in order to attempt to eject the attached Module from the Frame. This is called a "RELEASE pulse".

The consequences of setting an Interface State's RELEASE sub-state for a Secondary Interface to a Module, or when the Interface State's DETECT state is not DETECT_ACTIVE, are not defined by the Greybus Specification.

Note that the RELEASE sub-state only indicates whether the Frame is supplying ejection signaling to the corresponding Interface Block; it does *not* imply that a Module is attached to the Interface Block.

The SVC shall set the RELEASE sub-state of any Interface States associated with a *forcibly removed* Module to RELEASE_DEASSERTED after an implementation-defined delay.

5.2.8 INTF_TYPE

The values of the INTF_TYPE sub-state are given in Table 5.9.

The value of the INTF_TYPE sub-state is set by the SVC. Because the INTF_TYPE sub-state is communicated to the AP via Greybus Operations, its symbolic names are also given numeric values as shown in the table.

From the Module perspective, the physical connections made to Interface Blocks may not always support Greybus communications. Additionally, Greybus Systems are intended to concurrently support non-Greybus UniPro-based application protocols, such as UFS *[JEDEC-UFS]*.

The INTF_TYPE sub-state encodes this distinction for each Interface State.

When it is unknown whether a Module is attached to an Interface Block (DETECT sub-state is DETECT_UNKNOWN), or it is known that no Module is attached to an Interface Block (DETECT is DETECT_INACTIVE), the INTF_TYPE sub-state is IFT_UNKNOWN.

Subsequent sections describe how the AP and SVC coordinate during the Module detection and boot process to allow the SVC to set the INTF_TYPE sub-state, and how the AP is informed of its value.

5.2.9 ORDER

The values of the ORDER sub-state are given in Table 5.10.

The value of the ORDER sub-state is set by the SVC.

Value	Description
ORDER_UNKNOWN	No Module is attached, or Primary vs. Secondary status unknown
ORDER_PRIMARY	Interface is the Primary Interface to an attached Module
ORDER_SECONDARY	Interface is a Secondary Interface to an attached Module

Table 5.10: ORDER sub-state values

MAILBOX sub-state	Value	Description
MAILBOX_NULL	(none)	UNIPRO is UNIPRO_OFF; DME attribute access is not possible
MAILBOX_NONE (Reserved)	0x0	Initial DME attribute value; reserved for internal use
MAILBOX_AP	0x1	AP Interface is ready for *SVC Protocol* Connection
MAILBOX_GREYBUS	0x2	Module is ready for *Control Protocol* Connection
(Reserved)	0x3..0xFFFFFFFF	Reserved for future use

Table 5.11: MAILBOX sub-state values

A *Module* may attach to one or more Interface Blocks on a Slot in the Frame. Exactly one of these Interface Blocks is the "Primary Interface" to the Module; signaling on this Interface Block may be used to physically eject the Module from the Frame. All other Interface Blocks attached to the Module, if any, are "Secondary Interfaces": they may communicate via Greybus to the AP and the SVC, but the Frame cannot eject the Module through these Interface Blocks.

Whether an Interface Block is the Primary or a Secondary Interface to a Module is mirrored in the Interface State abstraction using the ORDER sub-state. The correspondence between the physical and abstract states is given in Table 5.10.

After a Module is attached to a Greybus System, the SVC determines which of the Interface Blocks it is attached to is the Primary Interface, and which are Secondary Interfaces, through implementation-defined means.

Note that the ORDER sub-state only indicates the most recent value set by the SVC, if any. It does *not* imply that a Module is attached to the Interface Block.

The SVC shall set the ORDER sub-state of any Interface States associated with a *forcibly removed* Module to ORDER_UNKNOWN after an implementation-defined delay.

5.2.10 MAILBOX

The MAILBOX sub-state is either the value MAILBOX_NULL or a 32-bit unsigned integer.

The MAILBOX sub-state represents the value of an implementation-defined DME attribute, named the "mailbox", which is present on each port in the UniPro switch inside the Frame.

The mailbox attribute ID is 0xA000, and its selector index is ignored.

When an Interface State's UNIPRO sub-state is UNIPRO_OFF, its MAILBOX sub-state is MAILBOX_NULL. Otherwise, it is a positive integer.

When an Interface State's UNIPRO sub-state is UNIPRO_UP, a Module may write to this DME attribute using a UniPro peer write. In a Greybus System, the SVC shall detect such a write and subsequently read the value of the mailbox attribute.

The values that a Module may write to the mailbox attribute are given in Table 5.11.

5.3 Initial Interface States

During the initialization of a Greybus System, the initial value of each Interface State is:

Sub-State	Value
DETECT	DETECT_UNKNOWN
V_SYS	V_SYS_OFF
V_CHG	V_CHG_OFF
WAKE	WAKE_UNSET
UNIPRO	UNIPRO_OFF
REFCLK	REFCLK_OFF
RELEASE	RELEASE_DEASSERTED
INTF_TYPE	IFT_UNKNOWN
ORDER	ORDER_UNKNOWN
MAILBOX	MAILBOX_NULL

Table 5.12: Initial Interface States

As a consequence of the reset sequence of a Greybus System, the SVC determines a value of DETECT for each Interface State in the system. This is explained in more detail in later sections, and forms the basis of the state machine described in *Interface Lifecycle States*.

5.4 Interfaces

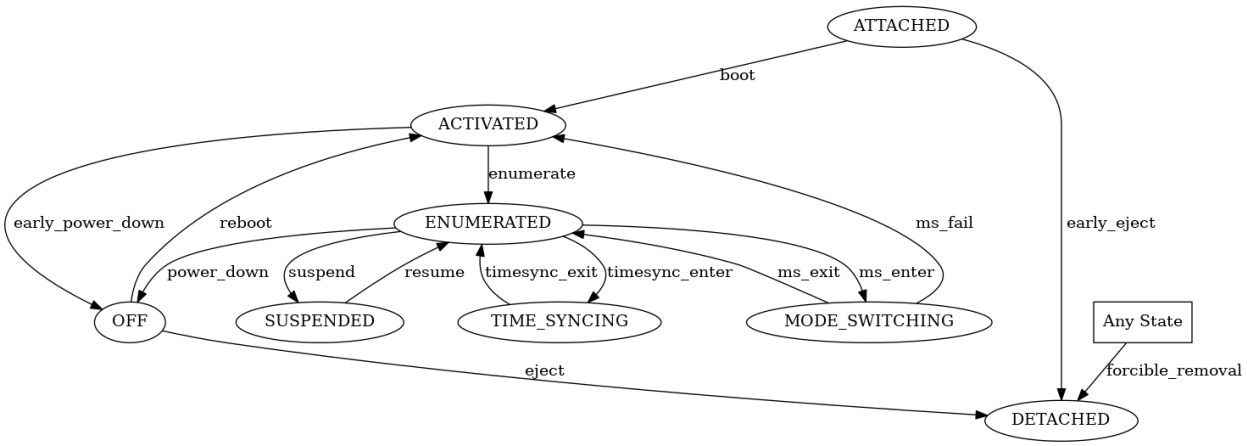
As stated above, a Module attached to the Frame may contain one or more entities called *Interfaces*, each of which is able to detect and respond to signals at a unique Interface Block to which the Module is attached. That is, each Interface communicates with the Frame via exactly one Interface Block, and no two Interfaces communicate with the Frame via the same Interface Block.

A Module shall contain exactly one Interface for each of the Interface Blocks to which it is attached. For brevity, it is written that an Interface "is connected to the Frame" via this Interface Block.

Interfaces within Modules shall communicate with the Frame as specified in this document, but Interfaces may vary in their capabilities. For example, an Interface may not be able to communicate via UniPro. Certain Interface communication capabilities can be discovered by the AP and SVC, which can record the information discovered in the *INTF_TYPE* sub-state of the Interface State associated with that Interface.

5.5 Interface Lifecycle States

This section briefly introduces the *Interface Lifecycle* state machine, shown in the following figure. A detailed description of this state machine is provided in *The Interface Lifecycle*.



Each of the states is a *Lifecycle State*. Lifecycle States denote the current status of an Interface, and transitions between Lifecycle States manage the dynamic behavior of the Interface as it interacts with the Frame. For example, in the ATTACHED Lifecycle State, the SVC has determined a Module is attached to an Interface Block, and thus an Interface can communicate with the Frame via that Interface Block. No other action has been taken by the Greybus System to communicate with the Interface, and it is unknown whether the Interface supports UniPro commmunication.

The DETACHED Lifecycle State is a special case. In this state, the SVC has determined an Interface Block has no Module attached. In this case, no Interface is connected to the Frame.

This section defines a group of Interface States which are the legal Interface States within the Frame when an Interface is in each Interface Lifecycle State.

For example, when an Interface is in the ACTIVATED Lifecycle State, the Interface State within the Frame has an INTF_TYPE other than IFT_UNKNOWN. Multiple permitted values for the sub-states of the Interface States within each Interface Lifecycle State are shown between angle brackets (<>).

The square node labeled “Any State” denotes that the transition is allowed from any Interface status whatsoever, and models the consequences of a *forcible removal*.

The Interface Lifecycle States are introduced, and their associated Interface States are defined, in the following sections.

Subsequent chapters define Greybus *Protocol*s, of which the *Control Protocol* and *SVC Protocol* are especially significant in terms of their impact on an Interface’s Lifecycle State. Following those chapters, a detailed description of the actions taken by the AP, SVC, and each Interface is given describing how transitions between Lifecycle States are managed.

5.5.1 ATTACHED

In the ATTACHED Lifecycle State, the SVC has:

- determined that a Module is attached to the Interface Block, setting DETECT to DETECT_ACTIVE
- determined whether this is the *Primary Interface* or a *Secondary Interface* to the Module, setting ORDER.

No actions have been taken to boot the Module, communicate with it via UniPro, etc. That is, in the ATTACHED Lifecycle State, the Interface State is otherwise identical to its *initial state*.

In the ATTACHED Lifecycle State, the following Interface States are allowed as described in later sections:

INTF_TYPE	UNIPRO	MAILBOX
IFT_DUMMY	UNIPRO_DOWN	MAILBOX_NONE
IFT_UNIPRO	UNIPRO_UP	MAILBOX_NONE
IFT_GREYBUS	UNIPRO_UP	MAILBOX_GREYBUS

Table 5.14: INTF_TYPE relationship to UNIPRO and MAILBOX in ACTIVATED

Sub-State	Value
DETECT	DETECT_ACTIVE
V_SYS	V_SYS_OFF
V_CHG	V_CHG_OFF
WAKE	WAKE_UNSET
UNIPRO	UPRO_OFF
REFCLK	REFCLK_OFF
RELEASE	RELEASE_DEASSERTED
INTF_TYPE	IFT_UNKNOWN
ORDER	ORDER_PRIMARY or ORDER_SECONDARY
MAILBOX	MAILBOX_NULL

Table 5.13: ATTACHED Lifecyle State

5.5.2 ACTIVATED

In the ACTIVATED Lifecycle State, system power and clock have been applied to the Interface Block, and an attempt to establish a UniPro link between Frame and Module has been made.

As a consequence, it is known whether the Module supports UniPro, so UNIPRO is either UNIPRO_DOWN or UNIPRO_UP. If UNIPRO is UNIPRO_UP, then the Module may signal readiness for communication via Greybus *Protocol*s by setting MAILBOX. Thus, MAILBOX either remains its initial value, MAILBOX_NONE, or is set by the Module to MAILBOX_GREYBUS.

The SVC also sets INTF_TYPE when the Interface is ACTIVATED, based on a combination of the UNIPRO and MAILBOX sub-states. The correspondence between UNIPRO, MAILBOX, and INTF_TYPE is given in Table 5.14.

In the ACTIVATED Lifecycle State, the following Interface States are allowed as described in later sections:

Sub-State	Value
DETECT	DETECT_ACTIVE
V_SYS	V_SYS_ON
V_CHG	V_CHG_OFF
WAKE	WAKE_UNSET
UNIPRO	UPRO_DOWN or UPRO_UP
REFCLK	REFCLK_ON
RELEASE	RELEASE_DEASSERTED
INTF_TYPE	IFT_DUMMY, IFT_UNIPRO, or IFT_GREYBUS
ORDER	ORDER_PRIMARY or ORDER_SECONDARY
MAILBOX	MAILBOX_NONE or MAILBOX_GREYBUS

Table 5.15: ACTIVATED Lifecyle State

5.5.3 ENUMERATED

The ENUMERATED Lifecycle State can only be reached when an Interface signals readiness for Greybus *Protocol* communication during the transition to ACTIVATED. Thus, INTF_TYPE is IFT_GREYBUS, and MAILBOX is MAILBOX_GREYBUS.

When an Interface is ENUMERATED, a Greybus *Control Protocol* Connection has been established to that Interface, and its *Manifest* has been read by the AP and successfully parsed.

For brevity, the phrases "an Interface is being enumerated" and "the AP is enumerating an Interface" shall mean that one of the following conditions holds:

- The Interface was *ACTIVATED*, its INTF_TYPE is IFT_GREYBUS, and the procedure in *Enumerate (ACTIVATED → ENUMERATED)* is subsequently being followed in the "enumerate" transition from ACTIVATED to ENUMERATED in the Interface Lifecycle state machine,
- The Interface was *MODE_SWITCHING*, and the procedure in *Mode Switch Exit (MODE_SWITCHING → ENUMERATED)* is subsequently being followed in the "ms_exit" transition from MODE_SWITCHING to ENUMERATED, or
- The Interface was *SUSPENDED*, and the procedure in *Resume (SUSPENDED → ENUMERATED)* is subsequently being followed in the "resume" transition from SUSPENDED to ENUMERATED.

The procedure is referred to as *enumeration* in any of the above cases. *Re-enumeration* may be used instead when an Interface is being enumerated a second or subsequent time.

While an Interface is ENUMERATED, the AP may determine through application- or Protocol-specific means that the Frame's reference clock is not required for the Interface to function correctly. Thus, REFCLK may be set to REFCLK_OFF.

Similarly, when the Interface is ENUMERATED, the AP may determine through application- or Protocol-specific means that the Interface can supply power to the Frame via the Interface Block. Thus, V_CHG may be set to V_CHG_ON.

In the ENUMERATED Lifecycle State, the following Interface States are allowed as described in later sections:

Sub-State	Value
DETECT	DETECT_ACTIVE
V_SYS	V_SYS_ON
V_CHG	V_CHG_OFF or V_CHG_ON
WAKE	WAKE_UNSET
UNIPRO	UPRO_UP
REFCLK	REFCLK_ON or REFCLK_OFF
RELEASE	RELEASE_DEASSERTED
INTF_TYPE	IFT_GREYBUS
ORDER	ORDER_PRIMARY or ORDER_SECONDARY
MAILBOX	MAILBOX_GREYBUS

Table 5.16: ENUMERATED Lifecyle State

5.5.4 MODE_SWITCHING

The MODE_SWITCHING Lifecycle State is a special case which is used to allow for re-enumeration of an Interface without physically removing it from, and attaching it to, a Greybus System.

As part of entering the MODE_SWITCHING Lifecycle State, all Greybus *Connection*s involving the Interface are closed. The Interface may then perform internal re-initialization, and subsequently signal to the Frame when this is complete by setting MAILBOX. The Frame can then attempt to re-enumerate the Interface, including retrieving its (possibly different) *Manifest* again.

Before an Interface enters the MODE_SWITCHING Lifecycle State, REFCLK shall be set to REFCLK_ON if it is REFCLK_OFF, and V_CHG shall be set to V_CHG_OFF if it is V_CHG_ON.

An Interface may enter and exit the MODE_SWITCHING Lifecycle State an arbitrary number of times.

In the MODE_SWITCHING Lifecycle State, the following Interface States are allowed as described in later sections:

Sub-State	Value
DETECT	DETECT_ACTIVE
V_SYS	V_SYS_ON
V_CHG	V_CHG_OFF
WAKE	WAKE_UNSET
UNIPRO	UPRO_UP
REFCLK	REFCLK_ON
RELEASE	RELEASE_DEASSERTED
INTF_TYPE	IFT_GREYBUS
ORDER	ORDER_PRIMARY or ORDER_SECONDARY
MAILBOX	MAILBOX_GREYBUS

Table 5.17: MODE_SWITCHING Lifecyle State

5.5.5 TIME_SYNCING

The TIME_SYNCING Lifecycle State represents the Interface state as the frame-time is being synchronized to an Interface from the SVC. For the duration of the TIME_SYNCING state it is not valid to generate a *WAKE Pulse* to an Interface.

A Greybus Operation *TimeSync Wake Pins Acquire Operation* is responsible for transitioning an Interface into the TIME_SYNCING state.

Once an Interface has entered the TIME_SYNCING state it will wait for the SVC to generate a known number of *TimeSync Pulse*s. The Interface will have been informed of how many *TimeSync Pulse*s to expect emanating from the SVC and shall mark the local time of the incoming *TimeSync Pulse* on the rising-edge of the *TimeSync Pulse*.

The *Greybus SVC TimeSync Wake Pins Release Operation* is responsible for transitioning an Interface out of the TIME_SYNCING state.

An Interface may enter and exit the TIME_SYNCING Lifecycle State an arbitrary number of times.

In the TIME_SYNCING Lifecycle State, the following Interface States are allowed as described in later sections:

Sub-State	Value
DETECT	DETECT_ACTIVE
V_SYS	V_SYS_ON
V_CHG	V_CHG_OFF or V_CHG_ON
WAKE	WAKE_UNSET or WAKE_ASSERTED or WAKE_DEASSERTED
UNIPRO	UPRO_UP
REFCLK	REFCLK_ON
RELEASE	RELEASE_DEASSERTED
INTF_TYPE	IFT_GREYBUS
ORDER	ORDER_PRIMARY or ORDER_SECONDARY
MAILBOX	MAILBOX_GREYBUS

Table 5.18: TIME_SYNCING Lifecyle State

5.5.6 SUSPENDED

The SUSPENDED Lifecycle State is a low-power state during which some internal state within the Interface is maintained, and system power is still applied. No Greybus Protocol communication with the Interface is possible when the Interface is in the SUSPENDED state.

An Interface shall not enter this state from the ENUMERATED state unless all bundles associated with it have entered the *BUNDLE_SUSPENDED* or *BUNDLE_OFF* state.

An Interface shall not alter its *Manifest* while it is entering, in, or exiting the SUSPENDED state.

In the SUSPENDED Lifecycle State, the following Interface States are allowed as described in later sections:

Sub-State	Value
DETECT	DETECT_ACTIVE
V_SYS	V_SYS_ON
V_CHG	V_CHG_OFF or V_CHG_ON
WAKE	WAKE_UNSET
UNIPRO	UPRO_HIBERNATE
REFCLK	REFCLK_OFF
RELEASE	RELEASE_DEASSERTED
INTF_TYPE	IFT_GREYBUS
ORDER	ORDER_PRIMARY or ORDER_SECONDARY
MAILBOX	MAILBOX_GREYBUS

Table 5.19: SUSPENDED Lifecyle State

5.5.7 OFF

The OFF Lifecycle State denotes an Interface which has power and communication signals disabled, but whose INTF_TYPE and ORDER are still known, having been determined during previous Lifecycle States in the Interface Lifecycle.

An Interface shall not enter this state from the ENUMERATED state unless all bundles associated with it have entered the *BUNDLE_OFF* state.

In the OFF Lifecycle State, the following Interface States are allowed as described in later sections:

Sub-State	Value
DETECT	DETECT_ACTIVE
V_SYS	V_SYS_OFF
V_CHG	V_CHG_OFF
WAKE	WAKE_UNSET
UNIPRO	UPRO_OFF
REFCLK	REFCLK_OFF
RELEASE	RELEASE_DEASSERTED
INTF_TYPE	IFT_DUMMY, IFT_UNIPRO, or IFT_GREYBUS
ORDER	ORDER_PRIMARY or ORDER_SECONDARY
MAILBOX	MAILBOX_NULL

Table 5.20: OFF Lifecyle State

5.5.8 DETACHED

The DETACHED Lifecycle State is a special case. In this Lifecycle State, no Module is attached to the Interface Block.

The SVC and AP have otherwise coordinated to disable power and other signaling to the Interface Block, as in the OFF Lifecycle State.

The unique Interface State possible in the DETACHED Lifecycle State is:

Sub-State	Value
DETECT	DETECT_INACTIVE
V_SYS	V_SYS_OFF
V_CHG	V_CHG_OFF
WAKE	WAKE_UNSET
UNIPRO	UPRO_OFF
REFCLK	REFCLK_OFF
RELEASE	RELEASE_DEASSERTED
INTF_TYPE	IFT_UNKNOWN
ORDER	ORDER_UNKNOWN
MAILBOX	MAILBOX_NULL

Table 5.21: DETACHED Lifecyle State

5.6 Bundle Power States

A Bundle represents a device in Greybus and as such is the smallest power-manageable entity. A Bundle is always in one of the following power states: BUNDLE_ACTIVE, BUNDLE_SUSPENDED or BUNDLE_OFF.

The Bundle power states impact the Interface Lifecycle transitions between the *ENUMERATED*, *SUSPENDED* and *OFF* states. For example, an Interface shall not enter the *SUSPENDED* state unless all Bundles associated with it are already in BUNDLE_SUSPENDED or BUNDLE_OFF state.

A Bundle State change request can only be issued by the AP when the Interface is in the *ENUMERATED* state.

After an Interface completes the transition to the *ACTIVATED* state, a Bundle is in the BUNDLE_OFF state and shall be activated only when requested by the AP.

All Bundles are required to support four power-state transitions: BUNDLE_ACTIVE -> BUNDLE_OFF, BUNDLE_OFF -> BUNDLE_ACTIVE, BUNDLE_ACTIVE -> BUNDLE_SUSPENDED and BUNDLE_SUSPENDED -> BUNDLE_ACTIVE.

A detailed specification of the Bundle and Interface power-management flow can be found in sections describing the related Greybus Operations.

5.6.1 BUNDLE_ACTIVE

The underlying hardware is fully operational, powered and Greybus Connections for all CPorts associated with this Bundle can be established if required by the AP. The Bundle shall enter this state only when its corresponding Interface is in the *ENUMERATED* state.

5.6.2 BUNDLE_SUSPENDED

The underlying hardware is in a low-power state and Greybus Connections for all CPorts associated with this Bundle are closed, but the internal context may be preserved (in an implementation-specific way) allowing the Bundle to quickly transition back to the BUNDLE_ACTIVE state. Any Greybus Connection that the AP might want to use shall be re-established when transitioning back to the BUNDLE_ACTIVE state.

5.6.3 BUNDLE_OFF

The underlying hardware is disabled and the context is lost. Greybus Connections for all CPorts associated with this Bundle are closed.

This is the default state of a Bundle after the *Boot (ATTACHED → ACTIVATED)* stage.

5.7 Special AP Module Requirements

As stated above, a Greybus System contains an AP Module and an SVC. This section defines special requirements related to these components.

- The AP Module shall be connected to the Frame via Interface Blocks whose Interface IDs are known to the SVC. The AP Module shall contain Interfaces as other Modules do, but these Interfaces shall not provide *Control CPort*s.

  For convenience, the Interface States with these Interface IDs are the *AP Interface States*, the corresponding Interface Blocks are *AP Interface Blocks*, and the corresponding Interfaces are *AP Interfaces*.

  Each AP Interface shall provide a CPort whose user can be configured to communicate with the SVC over a *Greybus Connection* implementing the *SVC Protocol*.

- The Interface Blocks by which the AP Module connects to the Frame may differ from those by which other Modules attach to the Frame, but AP Interface Blocks nonetheless have an associated Interface State as specified above.

- The following sub-states for all AP Interface States may, according to the implementation, be set by the AP, not the SVC:
  - REFCLK

- The following sub-states for all AP Interface States are defined as these constant values:
  - DETECT is DETECT_ACTIVE

  - V_SYS is V_SYS_ON
  - V_CHG is V_CHG_OFF
  - RELEASE is RELEASE_DEASSERTED
  - INTF_TYPE is IFT_GREYBUS
  - ORDER is ORDER_UNKNOWN
- The AP Module shall be able to restore the SVC to its reset state, and to release it from reset.

Chapter 6

Interface Information

> Imitation is the sincerest form of flattery.
>
> — Charles Caleb Colton

A Greybus Interface shall provide self-descriptive information in order to establish communications with other Interfaces on the UniPro network. This information is provided via two mechanisms:

- The Manifest, which describes components present within the Interface that are accessible via UniPro. The Manifest is a data structure, which includes a set of Descriptors, that presents a functional description of the Interface. Together, these Descriptors define the Interface's capabilities and means of communication via UniPro from the perspective of the application layer and above.
- Greybus Interface Attributes, which are UniPro DME attributes which also provide identifying information about the Interface.

6.1 Manifest

The Manifest is a contiguous block of data that includes a Manifest Header and a set of Descriptors. When read, a Manifest is transferred in its entirety. This allows the Interface to be described to the AP Module all at once, alleviating the need for multiple communication messages during the enumeration phase of the Interface.

6.1.1 Manifest Data Requirements

All data found in Manifest structures defined below shall adhere to the following general requirements:

- All numeric values shall be unsigned unless explicitly stated otherwise.
- All descriptor field values shall have little endian format.
- Numeric values prefixed with 0x are hexadecimal; they are decimal otherwise.
- All offset and size values are expressed in units of bytes unless explicitly stated otherwise.
- All string descriptors shall consist of UTF-8 encoded characters.
- All headers and descriptor data within a Manifest shall be implicitly followed by pad bytes as necessary to bring the structure's total size to a multiple of 4 bytes.
- Accordingly, the low-order two bits of all header *size* field values shall be 00.

Offset	Field	Size	Value	Description
0	size	2	Number	Size of the entire manifest
2	version_major	1	0	Greybus major version
3	version_minor	1	1	Greybus minor version

Table 6.1: Manifest Header

Offset	Field	Size	Value	Description
0	size	2	Number	Size of this descriptor
2	type	1	Number	*Descriptor type*
3	(pad)	1	0	Reserved (pad to 4 bytes)

Table 6.2: Descriptor Header

- Any reserved or unused space (including implicit padding) in a header or descriptor shall be ignored when read, and zero-filled when written.
- All major structures (like the Manifest header) and interface Protocols (like that between the AP Module and SVC) shall be versioned, to allow future extensions (or fixes) to be added and recognized.

6.1.2 Manifest Header

The Manifest Header is present at the beginning of the Manifest and defines the size of the manifest and the version of the Greybus Protocol with which the Manifest complies.

The values of version_major and version_minor shall refer to the highest version of this document (currently 0.1) with which the format complies.

version_minor increments with modifications to the Greybus definition, in such a way that any Protocol handler that supports the version_major can correctly interpret a Manifest in the modified format. A changed version_major indicates major differences in the Manifest format. It is not expected that a parser can properly interpret a Manifest whose version_major is greater than the version_major supported by the parser.

All Manifest parsers shall be able to interpret manifests formatted using older (lower numbered) Greybus versions, such that they still work properly (i.e. backwards compatibility is required).

The layout for the Manifest Header can be seen in Table 6.1.

6.1.3 Descriptors

Following the Manifest Header is one or more Descriptors. Each Descriptor is composed of a Descriptor Header followed by Descriptor Data. The format of the Descriptor Header can be seen in Table 6.2.

Descriptor type

The format of the Descriptor Data depends on the type of the descriptor, which is specified in the header. The known descriptor types and their values are described in Table 6.3.

Descriptor Type	Value
Invalid	0x00
Interface	0x01
String	0x02
Bundle	0x03
CPort	0x04
(All other values reserved)	0x05..0xff

Table 6.3: Descriptor Type

Offset	Field	Size	Value	Description
0	size	2	0x0008	Size of this descriptor
2	type	1	0x01	Type of the descriptor (Interface)
3	(pad)	1	0	Reserved (pad to 4 byte boundary)
4	vendor_string_id	1	ID	String ID for the vendor name
5	product_string_id	1	ID	String ID for the product name
6	features	1	Bit Mask	*Greybus Interface Descriptor Feature Bits*
7	(pad)	1	0	Reserved (pad to 4 byte boundary)

Table 6.4: Interface Descriptor

6.1.4 Interface Descriptor

Interface descriptor describes an access point for a Module to the UniPro network. Each interface represents a single physical port through which UniPro packets are transferred. Every Module shall have at least one interface. Each interface has a unique ID within the *Frame*.

This descriptor describes Interface-specific values as set by the vendor who created the Interface. Every Manifest shall have exactly one Interface descriptor as described in Table 6.4.

vendor_string_id is a reference to a specific string descriptor ID that provides a description of the vendor who created the Module. If there is no string present for this value in the Manifest, this value shall be 0x00. See the *String Descriptor* section below for more details.

product_string_id is a reference to a specific string descriptor ID that provides a description of the product. If there is no string present for this value in the Manifest, this value shall be 0x00. See the *String Descriptor* section below for more details.

Greybus Interface Descriptor Feature Bits

Table 6.5 defines the bits which specify the set of features supported by an Interface.

Symbol	Descirption	Value
GB_INTERFACE_TIME_SYNC	The Interface supports Greybus TimeSync Operations	0x01
	(All other values are reserved)	0x02..0x80

Table 6.5: Interface Descriptor Feature Bits

Offset	Field	Size	Value	Description
0	size	2	Number	Size of this descriptor
2	type	1	0x02	Type of the descriptor (String)
3	(pad)	1	0	Reserved (pad to 4 byte boundary)
4	length	1	Number	Length of the string in bytes
5	id	1	ID	String ID for this descriptor
6	string	*length*	UTF-8	Characters for the string
6+*length*	(pad)	0-3	0	Reserved (pad to 4 byte boundary)

Table 6.6: String Descriptor

Offset	Field	Size	Value	Description
0	size	2	0x0008	Size of this descriptor
2	type	1	0x03	Type of the descriptor (Bundle)
3	(pad)	1	0	Reserved (pad to 4 byte boundary)
4	id	1	ID	Interface-unique ID for this Bundle
5	class	1	Number	See Table 6.8
6	(pad)	2	0	Reserved (pad to 8 bytes)

Table 6.7: Bundle Descriptor

6.1.5 String Descriptor

A string descriptor provides a human-readable string for a specific value, such as a vendor or product string. Strings consist of UTF-8 characters and are not required to be zero terminated. A string descriptor shall be referenced only once within the Manifest, e.g. only one product (or vendor) string field may refer to string ID 2. The format of the string descriptor can be found in Table 6.6.

The *id* field shall not be 0x00, as that is an invalid String ID value.

The *length* field excludes any trailing padding bytes in the descriptor.

6.1.6 Bundle Descriptor

A Bundle represents a device in Greybus. Bundles communicate with each other on the network via one or more UniPro CPorts.

The *id* field uniquely identifies a Bundle within the Interface. The first Bundle shall have ID 0, the second (if present) shall have value 1, and so on. The purpose of these Ids is to allow CPort descriptors to define which Bundle they are associated with. The *id* field for a Bundle Descriptor shall not have value 0xff, as that is an invalid Bundle ID value. The Bundle descriptor is defined in Table 6.7.

The *class* field defines the class of the bundle. This shall be used by the AP to find what to expect from the bundle and how to configure/use it. Class types are defined in Table 6.8.

6.1.7 CPort Descriptor

A CPort Descriptor describes a CPort implemented within the Interface. Each CPort is associated with one of the Interface's Bundles, and has an ID unique among CPorts in that Interface. A CPort Descriptor declares the Greybus Protocol implemented by that CPort's User. This information may be used by the AP Module to interact with the CPort User.

Class type	Value
Control	0x00
Unused	0x01
Reserved	0x02
Reserved	0x03
Reserved	0x04
HID	0x05
Reserved	0x06
Reserved	0x07
Power Supply	0x08
Reserved	0x09
Bridged PHY	0x0a
Reserved	0x0b
Display	0x0c
Camera	0x0d
Sensor	0x0e
Lights	0x0f
Vibrator	0x10
Loopback	0x11
Audio	0x12
Reserved	0x13
Unused	0x14
Bootrom	0x15
Firmware Management	0x16
Log	0x17
(All other values reserved)	0x18..0xfd
Raw	0xfe
Vendor Specific	0xff

Table 6.8: Bundle Class Types

Offset	Field	Size	Value	Description
0	size	2	0x0008	Size of this descriptor
2	type	1	0x04	Type of the descriptor (CPort)
3	(pad)	1	0	Reserved (pad to 4 byte boundary)
4	id	2	ID	ID (destination address) of the CPort
6	bundle	1	ID	Bundle ID this CPort is associated with
7	protocol	1	Number	See Table 6.10

Table 6.9: CPort Descriptor

Greybus Interfaces shall contain a special *Control CPort*, which as CPort ID zero; the CPort User of this CPort shall implement the *Control Protocol*. An Interface Manifest shall not contain a CPort Descriptor with id field equal to zero.

The CPort Descriptor is defined in Table 6.9. The details of these Protocols are defined in the sections *Device Class Connection Protocols* and *Bridged PHY Connection Protocols* below.

Todo

The details of how the CPort identifier is determined will be specified in a later version of this document.

The *id* field is the CPort identifier used by other Modules to direct traffic to this CPort. The IDs for CPorts using the same Interface shall be unique. Certain low-numbered CPort identifiers (such as the control CPort) are reserved. Implementors shall assign CPorts low-numbered ID values, generally no higher than 31. (Higher-numbered CPort ids impact on the total usable number of UniPro devices and typically should not be used.)

6.2 Greybus Interface Attributes

A Greybus Interface capable of UniPro communication may support retrieval via DME Peer Get requests of the following values. If any of the Greybus Interface Attributes listed below is supported by an implementation, all shall be supported.

If the Greybus Interface Attributes are supported, their attribute IDs are implementation-defined.

- Ara Vendor ID: a 32 bit identifier, which identifies the vendor of the Project Ara Module containing the Interface.
- Ara Product ID: a 32 bit identifier which in combination with the Ara Vendor ID uniquely identifies the Greybus Module containing the Interface as a particular product released by that vendor.
- Ara Serial Number: a 64 bit identifier which is unique among all Modules, regardless of Ara Vendor ID or Ara Product ID. The Ara Serial Number may require multiple DME attributes for storage.
- Ara Initialization Status: a 32 bit identifier, which defines the initialization status of the Interface. When supported, this may be retrieved during interface initialization, as described in later chapters.

  If supported, the values of the Ara Initialization Status attribute are implementation-defined, with one exception: the values 0x00000006 and 0x00000009 are reserved for Interfaces implementing the *Bootrom Protocol*. Unless an Interface implements that Protocol, the Interface shall not set its Ara Initialization Status attribute to either of those values.

Protocol	Value
Control	0x00
Unused	0x01
GPIO	0x02
I2C	0x03
UART	0x04
HID	0x05
USB	0x06
SDIO	0x07
Power Supply	0x08
PWM	0x09
Unused	0x0a
SPI	0x0b
Display	0x0c
Camera Management	0x0d
Sensor	0x0e
Lights	0x0f
Vibrator	0x10
Loopback	0x11
Audio Management	0x12
Audio Data	0x13
SVC	0x14
Bootrom	0x15
Camera Data	0x16
Firmware Download	0x17
Firmware Management	0x18
Component Authentication	0x19
Log	0x1a
(All other values reserved)	0x1b..0xfd
Raw	0xfe
Vendor Specific	0xff

Table 6.10: CPort Protocol Numbers

Chapter 7

Greybus Operations

Greybus communication is built on the use of UniPro messages to send information between Modules. And although UniPro offers reliable transfer of data frames between interfaces, it is often necessary for the sender to know whether the effects of sending a message were what was expected. For example, a request sent to a UniPro switch controller requesting a reconfiguration of the routing table could fail, and proceeding as if a failure had not occurred in this case leads to undefined (and possibly dangerous) behavior. Similarly, the AP Module likely needs to retrieve information from other Modules; this requires that a message requesting information be paired with a returned message containing the information requested.

For this reason, Greybus performs communication between Modules using Greybus Operations. A Greybus Operation defines an activity (such as a data transfer) initiated in one Module that is implemented (or executed) by another. The particular activity performed is defined by the operation's type. An operation is generally implemented by a pair of messages–one containing a request, and the other containing a response, but *unidirectional* operations (i.e. requests without matching responses) are also supported. Both messages contain a simple header that includes the type of the operation and size of the message. In addition, each operation has a unique ID, and both messages in an operation contain this value so a response can be associated with its matching request (unidirectional operations use a reserved ID). Finally, all responses contain a byte in message header to communicate status of the operation–either success or a reason for a failure.

Whether a particular operation has a response message or not (i.e. is unidirectional) is protocol dependent. It usually makes sense for operations which may be initiated by the AP Module to have responses as any errors can be logged and often also reported up the stack (e.g. to userspace).

Operations are performed over Greybus Connections. A connection is a communication path between two Modules. Each end of a connection is a UniPro CPort, associated with a particular interface in a Greybus Module. A connection can be established once the AP Module learns of the existence of a CPort in another Module. The AP Module shall allocate a CPort for its end of the connection, and once the UniPro network switch is configured properly the connection can be used for data transfer (and in particular, for operations).

Each CPort in a Greybus Module has associated with it a Protocol. The Protocol dictates the way the CPort interprets incoming operation messages. Stated another way, the meaning of the operation type found in a request message depends on which Protocol the connection uses. Operation type 5 might mean "receive data" in one Protocol, while operation 5 might mean "go to sleep" in another. When the AP Module establishes a connection with a CPort in another Module, that connection uses the CPort's advertised Protocol.

Greybus Protocols may support *Protocol Versions*.

The Greybus Operations mechanism forms a base layer on which other Protocols are built. Protocols define the format of request messages, their expected response data, and the effect of the request on state in one or both Modules. Users of a Protocol can rely on the Greybus core getting the operation request message to its intended target, and transferring the operation status and any other data back. In the explanations that

Offset	Field	Size	Value	Description
0	size	2	Number	Size of this operation message
2	id	2	Number	Requestor-supplied unique request identifier
4	type	1	Number	Type of Greybus operation (Protocol-specific)
5	status	1	Number	Operation result (response message only)
6	(pad)	2	0	Reserved (pad to 8 bytes)

Table 7.1: Operation Message Header

follow, we refer to the interface through which a request operation is sent as the source, and the interface from which the response is sent as the destination.

7.1 Message Data Requirements

All data found in message structures defined below shall adhere to the following general requirements:

- All numeric values shall be unsigned unless explicitly stated otherwise.
- All numeric field values shall have little endian format.
- Numeric values prefixed with 0x are hexadecimal; they are decimal otherwise.
- All offset and size values are expressed in units of bytes unless explicitly stated otherwise.
- All string values shall consist of UTF-8 encoded characters.
- String values shall be paired with a numeric value indicating the number of characters in the string.
- String values shall not include terminating NUL characters.
- Any reserved space in a message structure shall be ignored when read, and zero-filled when written.
- All Protocols shall be versioned, to allow future extensions (or fixes) to be added and recognized.

Fields within a message payload have no specific alignment requirements. Message headers are padded to fill 8 bytes, so the alignment of a message's payload is comparable to that of its header. If alignment is required, it is achieved using explicitly defined reserved fields.

7.2 Operation Messages

Operation request messages and operation response messages have the same basic format. Each begins with a short header, and is followed by payload data. A response message records an additional status value in the header, and both requests and responses may have a zero-byte payload.

7.2.1 Operation Message Header

Table 7.1 summarizes the format of an operation message header.

The *size* includes the operation message header as well as any payload that follows it. As mentioned earlier, the meaning of a type value depends on the Protocol in use on the connection carrying the message. Only 127 operations are available for a given Protocol, 0x01..0x7f. Operation 0x00 is reserved as an invalid value for all Protocols. The high bit (0x80) of an operation type is used as a flag that distinguishes a request operation from its response. For requests, this bit is 0, for responses, it is 1. For example the request and response messages for operation 0x0a contain 0x0a and 0x8a (respectively) in their type fields. The ID allows

Status	Value	Meaning
GB_OP_SUCCESS	0x00	Operation completed successfully
GB_OP_INTERRUPTED	0x01	Operation processing was interrupted
GB_OP_TIMEOUT	0x02	Operation processing timed out
GB_OP_NO_MEMORY	0x03	Memory exhaustion prevented operation completion
GB_OP_PROTOCOL_BAD	0x04	Protocol is not supported by this Greybus implementation
GB_OP_OVERFLOW	0x05	Request message was too large
GB_OP_INVALID	0x06	Invalid argument supplied
GB_OP_RETRY	0x07	Request should be retried
GB_OP_NONEXISTENT	0x08	The device does not exist
GB_OP_INVALID_STATE	0x09	Request is incompatible with receiving Bundle state
Reserved	0x0a to 0xfd	Reserved for future use
GB_OP_UNKNOWN_ERROR	0xfe	Unknown error occured
GB_OP_INTERNAL	0xff	Invalid initial value.

Table 7.2: Operation Status Values

many operations to be "in flight" on a connection at once. The special ID 0 is reserved for unidirectional operations.

A connection Protocol is defined by describing the format of the operations supported by the Protocol. Each operation specifies the payload portions of the request and response messages used for the Protocol, along with all actions or state changes that take place as a result of the operation.

7.3 Greybus Operation Status

Table 7.2 defines the Greybus Operation status values.

The Greybus Operation status shall be determined by checking the status field of the Greybus Operation Message Header of the Response as described in Table 7.1.

A *Connection Protocol* can define its own status values in its Response payload if required. These status values shall be interpreted only by its respective protocol handler.

Note that *GB_OP_INTERNAL* shall not be used in a response message. It is reserved for internal use by the Greybus application stack only.

Chapter 8

Connection Protocols

The following sections define the request and response message formats for all operations for specific connection Protocols. Requests are most often (but not always) initiated by the AP Module. Each request has a unique identifier, supplied by the requestor, and each response includes the identifier of the request with which it is associated. This allows operations to complete asynchronously, so multiple operations can be "in flight" between the AP Module and a UniPro-attached adapter at once.

Each response includes a status byte in its message header, which communicates whether any error occurred in delivering or processing a requested operation. If the operation completed successfully, the status value is *GB_OP_SUCCESS*. Otherwise, the reason it was not successful is conveyed by one of the positive values defined in Table 7.2. All Protocols defined herein are subject to the *Message Data Requirements* listed above.

8.1 Protocol Versions

Note: Greybus no longer supports a common Version Operation for Individual Protocols, except for the *SVC Protocol*, the *Control Protocol*, and the *Bootrom Protocol*.

The Protocol version comprises of two one-byte values, major and minor. A Protocol definition can evolve to add new capabilities, and as it does so, its version changes. If existing (or old) Protocol handling code which complies with this specification can function properly with the new feature in place, only the minor version of the Protocol shall change. Any time a Protocol changes in a way that requires the handling code be updated to function properly, the Protocol's major version shall change.

Two Modules may implement different versions of a Protocol, and as a result they shall negotiate a common version of the Protocol to use. This is done by each side exchanging information about the version of the Protocol it supports at the time a connection between Module interfaces is set up. The version of a particular Protocol advertised by a Module is the same as the version of the document that defines the Protocol (so for Protocols defined herein, the version is 0.1). In the future, if the Protocol specifications are removed from this document, the versions will become independent of the overall Greybus Specification document.

To agree on a Protocol, an operation request supplies the (greatest) major and minor version of the Protocol supported by the source of a request. The request destination compares that version with the (greatest) version of the Protocol it supports. The version that is the largest common version number of the Protocol sent by both sides shall be the version that is to be used in communication between the devices. This chosen version is returned back as a response of the request.

Offset	Field	Size	Value	Description
0	version_major	1	Number	Offered Protocol major version
1	version_minor	1	Number	Offered Protocol minor version

Table 8.1: Common Greybus Protocol Version Request

8.2 Common Greybus Protocol Version Operation

Some Connection Protocols specify an Operation which allows the Protocol handling software on both ends of a connection to negotiate the version of the Protocol to use. This Operation shall be named the Protocol Version Operation. All Connection Protocol Operations with this name shall have the same semantics, as defined in this section.

Conceptually, this operation is:

`int version(u8 offer_major, u8 offer_minor, u8 *major, u8 *minor);`
: Negotiates the major and minor version of the Protocol used for communication over the connection. The requestor offers the version of the Protocol it supports. The respondent replies with the version that will be used - either the one offered if supported, or its own (lower) version otherwise.

The request value of each Protocol Version Operation shall be 0x01, and the response value for this Operation shall be 0x81.

For example, the corresponding Operation within the Greybus *Control Protocol* is named the Greybus Control Protocol Version Operation. Its request and response values are respectively 0x01 and 0x81.

For this operation, the request specifies the greatest version of the Protocol supported by the requestor. The response contains the version that shall be used for further communication – either the one offered if supported, or a lower version otherwise.

The following sections define the contents and semantics of this Operation's Request and Response messages.

8.2.1 Common Greybus Protocol Version Request

Table 8.1 defines the request payload for each Protocol's Protocol Version Operation. The request supplies the greatest major and minor version of the Connection Protocol supported by the sender.

The Common Greybus Protocol Version Request shall be sent only by the AP for all Protocols except the SVC Protocol. In the case of the SVC protocol, the request shall be sent only by the SVC.

The values of the version_major and version_minor fields shall be specified on a per-protocol basis; the subsequent sections of this document which define individual Connection Protocols specify the values of these fields for this Operation according to the particular Protocol defined in each section.

8.2.2 Common Greybus Protocol Version Response

Table 8.2 defines the response payload for each Protocol's Protocol Version Operation. The response supplies the version of the protocol that shall be used for any subsequent communication via the Connection.

The values of the version_major and version_minor fields shall be specified on a per-protocol basis; the subsequent sections of this document which define individual Connection Protocols specify the values of these fields for this Operation according to the particular Protocol defined in each section.

Offset	Field	Size	Value	Description
0	version_major	1	Number	Offered Protocol major version
1	version_minor	1	Number	Offered Protocol minor version

Table 8.2: Common Greybus Protocol Version Request

Offset	Field	Size	Value	Description
0	phase	1	1	Current phase in the closure sequence

Table 8.3: Common Greybus Protocol CPort Shutdown Request

8.3 Common Greybus Protocol CPort Shutdown Operation

With some exceptions, every *Connection Protocol* implements a "CPort Shutdown" *Operation*. This Operation is used as part of a sequence that closes a *Greybus Connection*, as described in *Connection Management*.

8.3.1 Common Greybus Protocol CPort Shutdown Request

Table 8.3.2 defines the Request payload for a Protocol's CPort Shutdown Operation. The value of the phase field in the Request payload shall equal one.

This Request shall only be sent by the AP. The AP should not send this Request except as described in *Connection Management*. The results of sending this Request under other circumstances are undefined.

8.3.2 Common Greybus Protocol CPort Shutdown Response

A Protocol's CPort Shutdown Response contains no payload.

The status byte in the Response's *Message Header* shall equal GB_OP_SUCCESS.

8.4 Connection Transmission Restrictions

Greybus Connections use UniPro CPorts to exchange application-specific payload data and, when the UniPro End-to-End Flow Control (E2EFC) feature is enabled, Flow Control Tokens.

Within a Greybus System, this exchange of data and Flow Control Tokens is subject to certain restrictions and recommendations defined in this section.

Specifically, when an Interface "**may transmit Segments on a CPort**", the following requirements and recommendations hold:

- Interfaces shall transmit Segments on CPorts only when permitted or required to do so by *[MIPI01]*, including Segments carrying Flow Control Tokens, regardless of whether the Segments carry payload data.
- Interfaces shall transmit Segments on CPorts involved in Greybus Connections only when permitted or required to do so by the Greybus Connection Protocol implemented by their respective CPort Users.
- If the UniPro E2EFC feature is enabled on a connected CPort, Interfaces should ensure that Segments carrying Flow Control Tokens are transmitted by that CPort as buffer space becomes available to its CPort User.

Additionally, when an Interface "**shall halt Segment transmission on a CPort**", the Interface shall ensure that the CPort's User shall subsequently neither:

- request the transfer of a Message Fragment to its peer CPort User, nor
- signal its ability to consume more data to its local CPort.

The CPort User may terminate a currently ongoing UniPro Message transmission or complete ongoing flow-control related transactions with its local CPort as a result of the Interface ensuring these conditions hold.

Chapter 9

Special Protocols

This section defines three Protocols, each of which serves a special purpose in a Greybus system.

The first is the *Control Protocol*. Interfaces may provide a CPort whose user implements the Control Protocol. The AP may establish a Connection between one of its Interfaces' CPorts and such CPorts. If it does, the AP may subsequently send Operations on that Connection to perform basic initialization of the Interface, configure it, send it notifications, and otherwise interact with the Interface at a high level. The AP may also use Control Connections while establishing and closing other Connections to CPorts declared in the Interface's *Manifest*.

The second is the *SVC Protocol*, which is used only between the SVC and AP Module. The SVC provides low-level control of the UniPro network. The SVC performs almost all of its activities under direction of the AP Module, and the SVC Protocol is used by the AP Module to exert this control. The SVC also uses this protocol to notify the AP Module of events, such as the insertion or removal of a Module.

The third is the *Bootrom Protocol*, which is used between the AP Module and any other module's bootloader to download firmware executables to the module. When a module's manifest includes a CPort using the Bootrom Protocol, the AP can connect to that CPort and download a firmware executable to the module. Bootrom protocol is deprecated for new designs requiring Firmware download to the Module. The *Firmware Download Protocol* should be used for any new designs.

9.1 Control Protocol

Interfaces with *INTF_TYPE* equal to IFT_GREYBUS shall provide a CPort that responds to the Operations defined in this section. Such a CPort is a *Control CPort*. If an Interface provides a Control CPort, its CPort ID shall be zero.

Such Interfaces shall be prepared to receive Operation requests on that CPort under conditions defined later in this chapter. In particular, this may occur as a result of successful *Interface Activate* and *Interface Resume* Operations, which are defined below in the *SVC Protocol*.

Also using a sequence of SVC Protocol Operations, the AP may establish a Greybus Connection to a Control CPort if it has determined that the Interface is prepared for incoming Operations on that CPort, and the Connection is not already established. Any such Connection is a *Control Connection*. This sequence is defined in *Control Connection Establishment*.

Interfaces are not notified when Control Connections are established.

Only the AP shall send requests on a Control Connection. Other Interfaces shall only send response messages. An Interface shall send a response on a Control Connection only after receiving a request from the AP.

Conceptually, the Operations in the Greybus Control Protocol are:

`int cport_shutdown(u8 phase);`
See *Common Greybus Protocol CPort Shutdown Operation*.

`int version(u8 offer_major, u8 offer_minor, u8 *major, u8 *minor);`
Refer to *Common Greybus Protocol Version Operation*.

`int get_manifest_size(u16 *manifest_size);`
This Operation is used by the AP to discover the size of a module's Interface Manifest. This is used after the SVC has discovered which Module contains the AP. The response to this Operation contains the size of the manifest, which is used by the AP to fetch the manifest later. This operation is only initiated by the AP.

`int get_manifest(u8 *manifest);`
This Operation is used by the AP after the SVC has discovered which Module contains the AP. The response to this Operation contains the manifest of the Module, which is used by the AP to determine the functionality module provides. This operation is only initiated by the AP.

`int connected(u16 cport_id);`
This Operation is used to notify an Interface that a Greybus connection has been established using the indicated CPort. Upon receiving this request, an Interface shall be prepared to receive messages on the indicated CPort. The Interface may send messages over the indicated CPort once it has sent a response to the connected request. This operation is never used for control CPort.

`int disconnecting(u16 cport_id);`
This Operation is used by the AP Module to inform an Interface that the process of disconnecting a previously established Greybus connection has begun.

`int disconnected(u16 cport_id);`
This Operation is used to notify an Interface that a previously established Greybus connection may no longer be used. This operation is never used for control CPort.

`int timesync_enable(u8 count, u64 frame_time, u32 strobe_delay, u32 refclk);`
The AP Module uses this operation to inform the Interface that frame-time is being enabled.

`int timesync_disable(void);`
The AP Module uses this operation to switch off frame-time logic in an Interface.

`int timesync_authoritative(u64 *frame_time);`
The AP Module uses this operation to inform an Interface of the authoritative frame-time reported by the SVC for each TIME_SYNC strobe.

`int timesync_get_last_event(u64 *frame_time);`
The AP Module uses this operation to get the frame-time at the last pulse on the wake pin of a relevant Interface. This operation is used in conjunction with an SVC timesync-ping operation to verify the local time at a given Interface.

`int bundle_version(u8 bundle_id, u8 *major, u8 *minor);`
This Operation is used by the AP to get the version of the Bundle Class implemented by a Bundle.

`void mode_switch(void);`
This Operation can be used by the AP to signal to the Interface that it may reinitialize itself and alter the Bundles it previously described to the AP by sending it an Interface *Manifest*.

`int bundle_suspend(u8 bundle_id);`
This Operation may be used by the AP to request the Bundle to enter a low-power state.

`int bundle_resume(u8 bundle_id);`
This Operation may be used by the AP to request the Bundle to exit the low-power state.

Control Operation Type	Request Value	Response Value
CPort Shutdown	0x00	0x80
Protocol Version	0x01	0x81
Reserved	0x02	0x82
Get Manifest Size	0x03	0x83
Get Manifest	0x04	0x84
Connected	0x05	0x85
Disconnected	0x06	0x86
TimeSync enable	0x07	0x87
TimeSync disable	0x08	0x88
TimeSync authoritative	0x09	0x89
Reserved	0x0a	0x8a
Bundle Version	0x0b	0x8b
Disconnecting	0x0c	0x8c
TimeSync get last event	0x0d	0x8d
Mode Switch	0x0e	N/A
Bundle Suspend	0x0f	0x8f
Bundle Resume	0x10	0x90
Bundle Deactivate	0x11	0x91
Bundle Activate	0x12	0x92
Interface Suspend Prepare	0x13	0x93
Interface Deactivate Prepare	0x14	0x94
Interface Hibernate Abort	0x15	0x95
(all other values reserved)	0x16..0x7e	0x96..0xfe
Invalid	0x7f	0xff

Table 9.1: Control Operation Types

`int bundle_deactivate(u8 bundle_id);`
This Operation may be used by the AP to request that a Bundle be powered off.

`int bundle_activate(u8 bundle_id);`
This Operation may be used by the AP to request that a Bundle be powered on.

`int intf_suspend_prepare(void);`
This Operation may be used by the AP to request the Interface to prepare for the transition to a low-power state.

`int intf_deactivate_prepare(void);`
This Operation may be used by the AP to request the Interface to prepare to be powered down.

`void intf_hibernate_abort(void);`
This Operation may be used by the AP to abort a previously issued Interface Suspend Prepare or Interface Deactivate Prepare request.

9.1.1 Greybus Control Operations

All control Operations are contained within a Greybus control request message. Most of control requests results in a matching response, except mode_switch which is unidirectional. The request and response messages for each control Operation are defined below.

Table 9.1 defines the Greybus Control Protocol Operation types and their values. Both the request type and response type values are shown.

Offset	Field	Size	Value	Description
0	manifest_size	2	Number	Size of the Manifest

Table 9.2: Control Protocol Get Manifest Size Response

9.1.2 Greybus Control CPort Shutdown Operation

The Greybus Control CPort Shutdown Operation is the *Common Greybus Protocol CPort Shutdown Operation* for the Control Protocol.

9.1.3 Greybus Control Protocol Version Operation

The Greybus Control Protocol Version Operation is the *Common Greybus Protocol Version Operation* for the Control Protocol.

Greybus implementations adhering to the Protocol specified herein shall specify the value 0 for the version_major and 1 for the version_minor fields found in this Operation's request and response messages.

9.1.4 Greybus Control Get Manifest Size Operation

The Greybus Control Get Manifest Size Operation is used by the AP to ensure an Interface's *Manifest* is available for retrieval via Greybus. After this Operation is successfully exchanged, the AP may retrieve the Manifest using the *Greybus Control Get Manifest Operation*.

Although the AP may send this request at any time, it should only do so while enumerating an Interface, as defined in *ENUMERATED*. The effect of this Operation under other conditions is unspecified.

Greybus Control Get Manifest Size Request

The Greybus Control Get Manifest Size Request has no payload.

The Greybus Control Get Manifest Size Request is sent by the AP to the Interface in order to request that the Interface ensure its Manifest data structure is available for subsequent retrieval.

If an Interface is being enumerated, the Interface shall ensure an Interface Manifest is available for later retrieval by the AP as a result of receiving this request. It shall then notify the AP of the size of this Manifest in the response, as described below.

Greybus Control Get Manifest Size Response

The Greybus Control Get Manifest Size Response contains a two byte field, manifest_size. If the response status is not GB_OP_SUCCESS, the value of manifest_size is undefined and shall be ignored.

The manifest_size field in the response payload shall contain the size in bytes of the Interface Manifest which may be subsequently retrieved by the AP. If an Interface is being enumerated when it sends this response, the Interface shall not alter the size of this Interface Manifest as long as it continues being enumerated.

9.1.5 Greybus Control Get Manifest Operation

The Greybus Control Get Manifest Operation is used by the AP to retrieve an Interface's *Manifest* via its Control Connection.

Offset	Field	Size	Value	Description
0	manifest	variable	Data	Manifest

Table 9.3: Control Protocol Get Manifest Response

Though the AP may send this request at any time, it should only do so while enumerating an Interface, as defined in *ENUMERATED*. The effect of this Operation under other conditions is unspecified.

Greybus Control Get Manifest Request

The Greybus Control Get Manifest Request has no payload.

If the Interface is being enumerated, its Manifest is available for retrieval by the AP. The Interface shall send it in the response to this request.

Greybus Control Get Manifest Response

The Greybus Control Get Manifest Response contains a block of data that describes the functionality provided by the Interface. The contents of this data are defined in *Manifest*. If the response status is not GB_OP_SUCCESS, the response payload should be empty and shall be ignored.

If the Interface is being enumerated when it sends this response, the size of the Manifest returned by the Interface in this response shall equal the manifest_size field in the preceding Get Manifest Size Response payload. The size is otherwise not specified.

The Interface shall ensure that if it is being enumerated and the response status is GB_OP_SUCCESS, the following shall hold:

1. If the Interface provides CPort Descriptors in the Manifest, then it shall respond to incoming Operation Requests on those CPorts after the AP establishes Greybus Connections using those CPorts as described in *Connection Management*.
2. The Greybus *Protocol*s implemented by the CPort users of any such CPorts shall be as defined in the Manifest.

When the AP receives a successful response, and parses the *Manifest* successfully, the Interface's Lifecycle State is ENUMERATED. The enumeration procedure guarantees that the Interface State is in one of two possible values, as follows:

Sub-State	Value
DETECT	DETECT_ACTIVE
V_SYS	V_SYS_ON
V_CHG	V_CHG_OFF
WAKE	WAKE_UNSET
UNIPRO	UPRO_UP
REFCLK	REFCLK_ON
RELEASE	RELEASE_DEASSERTED
INTF_TYPE	IFT_GREYBUS
ORDER	ORDER_PRIMARY or ORDER_SECONDARY
MAILBOX	MAILBOX_GREYBUS

Table 9.4: Interface States after Successful Greybus Control Get Manifest Response

Offset	Field	Size	Value	Description
0	cport_id	2	Number	CPort that is now connected

Table 9.5: Control Protocol Connected Request

The Interface shall ensure that as long as the Interface State remains this value, that the above list of two conditions in this section shall continue to hold.

The AP and Interface may subsequently, through Protocol-specific means, change the values of some of these sub-states without relaxing these requirements.

9.1.6 Greybus Control Connected Operation

Note: The Control Connected Operation is currently defined under the assumption that all Connections in the Greybus System are between an AP Interface and another, non-AP Interface in the System.

The results in the case of Connections between two Interfaces, neither or both of which are AP Interfaces, are undefined.

The AP may establish Connections between Interfaces in the Greybus System. If the *Interface State* of an Interface has *INTF_TYPE* IFT_GREYBUS, the AP shall only attempt to establish non-Control Connections to that Interface if its Lifecycle State is *ENUMERATED*.

Connection establishment is performed by the AP using a sequence of Operations in the Control and SVC Protocols, as defined in this chapter. A later chapter, *Module and Interface Lifecycles*, provides procedures using these Operations which establish connections in *Connection Management*. As part of these procedures, the AP uses a Greybus Control Connected Operation to notify Interfaces when Connections are established.

Greybus Control Connected Request

The Greybus control connected request supplies the CPort ID on the receiving Interface that has been connected.

The AP should ensure that the CPort ID given by cport_id in the request payload was given in the id field of a *CPort Descriptor* in the Interface's *Manifest*. The results of this Operation under other circumstances are undefined.

Interfaces shall not transmit any UniPro Segments on any CPorts identified in their Manifests' CPort Descriptors before receiving a Control Connected Request indicating that the CPort is now connected, regardless of whether the Segments contain L4 payload.

After receiving this request, the Interface may transmit Segments on the CPort given by cport_id, as described in *Connection Transmission Restrictions*.

Greybus Control Connected Response

The Greybus control connected response message contains no payload.

If the AP receives a Control Connected response with status GB_OP_SUCCESS, it shall store information indicating that the CPort is now connected on that Interface.

The AP may later close the Greybus Connection and disconnect the CPort using a sequence of Operations in the Control and SVC Protocols. This procedure is defined in *Connection Management*, and uses Greybus

Offset	Field	Size	Value	Description
0	cport_id	2	Number	CPort that is being disconnected

Table 9.6: Control Protocol Disconnecting Request

Operations defined in this chapter. If this procedure succeeds, the AP no longer needs to store the information that the CPort is connected.

The AP also no longer needs to store information indicating that a CPort on an Interface is connected if subsequent Operations guarantee that the Interface's Lifecycle State is *ATTACHED*, *ACTIVATED*, *OFF*, or *DETACHED*.

The AP should not send a Control Connected Request to an Interface with a cport_id field if it has stored information indicating that the CPort is connected. If this occurs, the results are undefined.

The AP Interface shall not transmit UniPro Segments to a CPort identified by an Interface Manifest's CPort Descriptors unless it successfully exchanges a Control Connected Operation with the Interface as part of Greybus Connection establishment, as described in *Non-Control Connection Establishment*. After this successful exchange of a Control Connected Operation, the AP Interface may transmit Segments on the CPort at its end of the Connection, as described in *Connection Transmission Restrictions*.

9.1.7 Greybus Control Disconnecting Operation

Note: The Control Disconnected Operation is currently defined under the assumption that all Connections in the Greybus System are between an AP Interface and another, non-AP Interface in the System.

The results in the case of Connections between two Interfaces, neither or both of which are AP Interfaces, are undefined.

After establishing a Greybus Connection from an AP Interface to another Interface, the AP may later use the Greybus Control Disconnecting Operation to notify the Interface that the Connection is being closed, and thus that the CPort will later be disconnected.

Procedures the AP may use to establish and close Greybus Connections are provided in *Connection Management*. Use of this Operation is part of those procedures.

Greybus Control Disconnecting Request

The Greybus Control Disconnecting request supplies the CPort ID on the receiving Interface that is being closed.

After sending this request to notify the Interface that a Connection is closing, the AP Interface may transmit Segments on the CPort at its end of the Connection as defined in *Connection Transmission Restrictions* if one or more of the following conditions hold:

- when issuing responses to requests it has already received on the Connection,
- when exchanging *CPort Shutdown Operations* with the Interface, or
- when transmitting UniPro Flow Control Tokens.

The AP Interface shall otherwise halt Segment transmission on the CPort.

Offset	Field	Size	Value	Description
0	cport_id	2	Number	CPort that is now disconnected

Table 9.7: Control Protocol Disconnected Request

The AP Interface may send a Control Disconnecting Operation with a cport_id field equal to zero when disconnecting a Control Connection, but should not do so if it has stored information indicating that other CPorts on that Interface are connected.

After receiving the request, the Interface may transmit Segments on the CPort at its end of the Connection as defined in *Connection Transmission Restrictions* if one or more of the following conditions hold:

- when issuing responses on the Connection to Operations whose requests it received before the Control Disconnecting Operation Request,
- when exchanging CPort Shutdown Operations with the AP, or
- when transmitting UniPro Flow Control Tokens.

The receiving Interface shall otherwise halt Segment transmission on the CPort.

If the receiving Interface issues any Responses on the Connection to Operations whose Requests it received before this Request, it shall do so before sending the Control Disconnecting Response.

Greybus Control Disconnecting Response

The Greybus Control Disconnecting response message contains no payload.

The response status shall equal GB_OP_SUCCESS.

Before issuing a response to a Disconnecting request, the Interface shall ensure that any further UniPro Messages received on the CPort associated with its side of the Connection are immediately discarded, unless the Messages are well-formed Greybus CPort Shutdown Requests.

9.1.8 Greybus Control Disconnected Operation

Note: The Control Disconnected Operation is currently defined under the assumption that all Connections in the Greybus System are between an AP Interface and another, non-AP Interface in the System.

The results in the case of Connections between two Interfaces, neither or both of which are AP Interfaces, are undefined.

The Greybus Control Disconnected Operation is sent to notify an Interface that a Greybus Connection has been closed. The users of the CPorts at each end of the Connection shall no longer transmit data on their respective CPorts unless a new Connection is established using those CPorts. Any messages received by the Interface on the CPort after the Control Disconnected Request is received shall be discarded, unless a Greybus Connection is later reestablished on that CPort.

Greybus Control Disconnected Request

The Greybus Control Disconnected Request supplies the CPort ID on the receiving Interface for the Greybus Connection which is now closed. The UniPro CPort on the Interface which was at one end of the Connection may subsequently be disconnected by the SVC.

Offset	Field	Size	Value	Description
0	count	1	Number	Number of TIME_SYNC pulses
1	frame_time	8	Number	The initial frame-time to intiailze to
9	strobe_delay	4	Number	Inter-strobe delay in milliseconds
13	refclk	4	Number	The clock rate of the frame-time counter

Table 9.8: Control Protocol TimeSync Enable Request

After receiving the request, the Interface shall perform any implementation-defined procedures required to make the CPort usable if a Greybus Connection is later reestablished on that CPort. The Interface may set local UniPro attributes related to that CPort to implementation-defined values as part of this process. If such procedures are required by the Interface, it shall complete them before sending the response.

Before sending the response, the receiving Interface shall halt Segment transmission on the CPort given by cport_id as described in *Connection Transmission Restrictions*.

Greybus Control Disconnected Response

The Greybus Control Disconnected Response message contains no payload.

The response status shall equal GB_OP_SUCCESS.

After receiving the response, the AP shall halt Segment transmission on the CPort which was at its end of the Connection which is now closed, as defined in *Connection Transmission Restrictions*.

9.1.9 Greybus Control TimeSync Enable Operation

The AP Module uses this operation to inform the Interface of an upcoming pulse-train of TIME_SYNC strobes. The 'count' parameter informs the Interface of how many TIME_SYNC strobes will be issued. The range of the count variable is from 1..4. The 'frame_time' parameter informs the Interface to immediately seeds its frame-time to a value given by the AP. The 'strobe_delay' parameter informs the Interface of the expected delay between each TIME_SYNC strobe. The 'refclk' parameter informs the Interface of the required clock rate to run its frame-time tracking counter at.

A later operation initiated by the AP will inform the Interface of the authoritative frame-time at each TIME_SYNC strobe.

Greybus Control TimeSync Enable Request

Table 9.8 defines the Greybus Control TimeSync Enable Request payload. The request supplies the number of TIME_SYNC strobes to come (count), the initial time (frame_time) the delay between each strobe (strobe_delay) and the required clock rate to run the local timer at (refclk).

Greybus Control TimeSync Enable Response

The Greybus Control Protocol TimeSync Enable response contains no payload.

9.1.10 Greybus Control TimeSync Disable Operation

The AP Module uses this operation to inform an Interface to stop tracking frame-time. The Interface will immediately stop tracking frame-time.

Offset	Field	Size	Value	Description
0	time_sync0	8	Number	Authoritative frame-time at TIME_SYNC0
8	time_sync1	8	Number	Authoritative frame-time at TIME_SYNC1
16	time_sync2	8	Number	Authoritative frame-time at TIME_SYNC2
24	time_sync3	8	Number	Authoritative frame-time at TIME_SYNC3

Table 9.9: Control Protocol TimeSync Authoritative Request

Greybus Control TimeSync Disable Request

The Greybus Control Protocol TimeSync Disable request contains no payload.

Greybus Control TimeSync Disable Response

The Greybus Control Protocol TimeSync Disable response contains no payload.

9.1.11 Greybus Control TimeSync Authoritative Operation

The AP Module uses this operation to inform the Interface of the previous authoritative frame-time at each TIME_SYNC strobe. The AP will store and forward this data to an Interface after interrogating this data from the SVC. Unused entires in the request shall be initialized to zero.

Greybus Control TimeSync Authoritative Request

Table 9.9 defines the Greybus Control TimeSync Authoritative Request payload. The authoritative frame-time at each TIME_SYNC strobe as reported by the SVC to the AP Module is stipulated. Unused slots in the response shall contain zero.

Greybus Control TimeSync Authoritative Response

The Greybus Control Protocol TimeSync Authoritative Response contains no payload.

9.1.12 Greybus Control TimeSync Get Last Event Operation

The AP Module uses this operation to extract the last frame-time from an Interface associated with a wake event.

Greybus Control TimeSync Get Last Event Request

The Greybus Control Protocol TimeSync Get Last Event Request contains no payload.

Greybus Control TimeSync Get Last Event Response

Table 9.1.13 defines the Greybus Control TimeSync Get Last Event Response payload. The frame-time at the last wake event is returned.

Offset	Field	Size	Value	Description
0	frame-time	8	Number	frame-time at the last wake event.

Table 9.10: Control Protocol TimeSync Get Last Event Response

Offset	Field	Size	Value	Description
0	bundle_id	1	Number	Bundle ID

Table 9.11: Control Protocol Bundle Version request

9.1.13 Greybus Control Bundle Version Operation

The AP uses this operation to retrieve the version of the Bundle Class implemented by a Bundle. The version is represented by two 1-byte numbers, major and minor.

The version of a particular Bundle Class advertised by an Interface is the same as the version of the document that defines the Bundle Class and its subprotocols (so for Bundle Classes defined herein, the version is 0.1). In the future, if the Bundle Class specifications are removed from this document, the versions will become independent of the overall Greybus Specification document.

Greybus Control Bundle Version Request

Table 9.11 defines the Greybus Control Bundle Version Request payload. The request contains the ID of the Bundle whose Bundle Class version is to be returned.

Greybus Control Bundle Version Response

Table 9.12 defines the Greybus Control Bundle Version Response payload. The response contains two 1-byte numbers, major and minor.

9.1.14 Greybus Control Mode Switch Operation

The AP can use this Operation to notify the Interface of the following:

- The Control Connection is closed
- The Interface may now alter its Bundles

Although the AP may send this request at any time, it should only do so during the "ms_enter" transition from the *ENUMERATED* Interface *Lifecycle State* to *MODE_SWITCHING*, as defined in *The Interface Lifecycle*. This is described in *Mode Switch Enter (ENUMERATED → MODE_SWITCHING)*. The effect of this Operation under other conditions is unspecified.

Note that the Greybus Control Mode Switch Operation is unidirectional and has no response. This is a necessary consequence of the fact that the AP uses this Operation Request to inform the Interface that

Offset	Field	Size	Value	Description
0	major	1	Number	Major number of the version
1	minor	1	Number	Minor number of the version

Table 9.12: Control Protocol Bundle Version Response

the Control Connection is now closed, since Interfaces shall not transmit data on CPorts whose Greybus Connections are closed.

Instead, when the Interface is ready to signal completion of its handling of this Operation, it shall do so by setting the *MAILBOX* sub-state of its associated Interface State. The SVC shall detect when MAILBOX is set and, other than in certain special circumstances, shall subsequently notify the AP using a *Greybus SVC Interface Mailbox Event Operation*. This indirect mechanism allows the Interface to notify the AP when the processing that results from a Mode Switch Request has completed.

Any timeouts limiting the duration between the receipt of the Mode Switch request and a subsequent MAILBOX write by the Interface are implementation-defined.

Greybus Control Mode Switch Request

The Greybus Control Mode Switch Request contains no payload.

The AP shall send this request only as the final step in the procedure defined below in *Control Connection Closure for ms_enter*. When the Interface receives the request, its Control Connection is now closed.

After receiving the request, the Interface shall perform any implementation-defined procedures required to make the Control CPort usable if a Greybus Connection is later reestablished on that CPort. The Interface may set local UniPro attributes related to that CPort to implementation-defined values as part of these procedures.

The Interface may now release any internal resources it had acquired in response to Control Get Manifest Size or Control Get Manifest Operations. In particular, the Interface may now stop responding to incoming Operation requests on CPorts whose users previously had been configured to implement Greybus Protocols other than the Control Protocol. The effects of the AP subsequently establishing Greybus Connections and attempting to exchange data with any such CPorts are, other than the constraints defined in this version of the Greybus Specification, not specified.

After any such procedures are complete, the Interface shall write the value MAILBOX_GREYBUS to its Interface State's MAILBOX attribute. Before doing so, the Interface shall ensure it can subsequently respond to incoming *Control Protocol* Operation Requests if its Control Connection is reestablished. If the Interface cannot ensure this, it shall not set the MAILBOX state as a result of receiving this request.

9.1.15 Greybus Control Bundle Suspend Operation

The AP may use this Operation to request a Bundle to enter the *BUNDLE_SUSPENDED* state in which all Connections associated with this Bundle are closed by the AP but the Bundle's context may be preserved in an implementation-defined way.

The AP shall close all Connections associated with this Bundle (as described in *Non-Control Connection Closure*) before sending the Bundle Suspend Request.

The Bundle shall be considered *BUNDLE_SUSPENDED* after the AP receives a Response indicating the Operation has completed successfully.

Greybus Control Bundle Suspend Request

Table 9.13 defines the Greybus Control Bundle Suspend Request payload. The Request contains a one-byte Bundle ID corresponding with the Bundle IDs received in the Manifest as described in *Manifest*.

The AP may send this Request to a Bundle which is in the *BUNDLE_ACTIVE* state. An Interface shall send a Response containing the GB_CONTROL_BUNDLE_PM_OK if the AP requests to suspend a Bundle

Offset	Field	Size	Value	Description
0	bundle_id	1	Number	Bundle ID

Table 9.13: Control Protocol Bundle Suspend Request

Offset	Field	Size	Value	Description
0	status	1	Number	Bundle PM status (one of the values defined in Table 9.21)

Table 9.14: Control Protocol Bundle Suspend Response

which is already suspended. Sending this Request to a Bundle which is in the *BUNDLE_OFF* state shall result in the Bundle Suspend Response containing the GB_CONTROL_BUNDLE_PM_NA error code.

Upon reception of this Request the Bundle indicated by the bundle_id field in the Request payload should perform implementation-defined procedures required to enter the *BUNDLE_SUSPENDED* state.

Greybus Control Bundle Suspend Response

Table 9.14 defines the Greybus Control Bundle Suspend Response payload. The Response contains a one-byte status value indicating the result of the Operation. Valid status values are defined in Table 9.21.

The AP shall verify both the Greybus return value and the Bundle PM status upon reception of the Response. Only when the Greybus Operation returns GB_OP_SUCCESS and the Bundle Suspend Response contains GB_CONTROL_BUNDLE_PM_OK may the Bundle be considered suspended. Any other combination indicates an error.

The AP shall re-establish the Connections (as described in *Non-Control Connection Establishment*) if a status code indicating an error was returned in the Response in which case the Bundle shall not be considered suspended.

9.1.16 Greybus Control Bundle Resume Operation

The AP may use this Operation to request a specific Bundle to transition from the *BUNDLE_SUSPENDED* state to the *BUNDLE_ACTIVE* state.

Greybus Control Bundle Resume Request

Table 9.15 defines the Greybus Control Bundle Resume Request payload. The Request contains a one-byte Bundle ID corresponding with the Bundle IDs received in the Manifest as described in *Manifest*.

The AP may send this Request to a Bundle which is in the *BUNDLE_SUSPENDED* state. An Interface shall send a Response containing the GB_CONTROL_BUNDLE_PM_OK if the AP requests to resume a Bundle which is already active. Sending this Request to a Bundle which is in the *BUNDLE_OFF* state shall result in the Bundle Resume Response containing the GB_CONTROL_BUNDLE_PM_NA error code.

Upon reception of this Request the Bundle indicated by the bundle_id field in the Request payload shall perform implementation-defined procedures needed to exit the *BUNDLE_SUSPENDED* state.

Greybus Control Bundle Resume Response

Table 9.16 defines the Greybus Control Bundle Resume Response payload. The Response contains a one-byte return value indicating the result of the Operation. Valid return values are defined in Table 9.21.

Offset	Field	Size	Value	Description
0	bundle_id	1	Number	Bundle ID

Table 9.15: Control Protocol Bundle Resume Request

Offset	Field	Size	Value	Description
0	status	1	Number	Bundle PM status (one of the values defined in Table 9.21)

Table 9.16: Control Protocol Bundle Resume Response

The AP shall verify both the Greybus return value and the Bundle PM status upon reception of the Response. Only when the Greybus Operation returns GB_OP_SUCCESS and the Bundle Resume Response contains GB_CONTROL_BUNDLE_PM_OK may the Bundle be considered active. Any other combination indicates an error.

If the Response does not indicate an error, the AP may establish Greybus Connections on all CPorts associated with this Bundle as described in *Non-Control Connection Establishment*.

9.1.17 Greybus Control Bundle Deactivate Operation

The AP may use this Operation to request a Bundle to enter the *BUNDLE_OFF* state in which all Connections associated with this Bundle are closed by the AP, the underlying hardware is powered off and the Bundle implementation-defined context is lost.

The AP shall close all Connections associated with this Bundle (as described in *Non-Control Connection Closure*) before sending the Bundle Deactivate Request.

The Bundle shall be considered *BUNDLE_OFF* after the AP receives a Response indicating the Operation has completed successfully.

Greybus Control Bundle Deactivate Request

Table 9.17 defines the Greybus Control Bundle Deactivate Request payload. The Request contains a one-byte Bundle ID corresponding with the Bundle IDs received in the Manifest as described in *Manifest*.

The AP may send this Request to a Bundle which is in the *BUNDLE_ACTIVE* state. An Interface shall send a Response containing the GB_CONTROL_BUNDLE_PM_OK if the AP requests to deactivate a Bundle which is already off. Sending this Request to a Bundle which is in the *BUNDLE_SUSPENDED* state shall result in the Bundle Deactivate Response containing the GB_CONTROL_BUNDLE_PM_NA error code.

Upon reception of this Request the Bundle indicated by the bundle_id field in the Request payload should perform implementation-defined procedures required to enter the *BUNDLE_OFF* state.

Offset	Field	Size	Value	Description
0	bundle_id	1	Number	Bundle ID

Table 9.17: Control Protocol Bundle Deactivate Request

Offset	Field	Size	Value	Description
0	status	1	Number	Bundle PM status (one of the values defined in Table 9.21)

Table 9.18: Control Protocol Bundle Deactivate Response

Offset	Field	Size	Value	Description
0	bundle_id	1	Number	Bundle ID

Table 9.19: Control Protocol Bundle Activate Request

Greybus Control Bundle Deactivate Response

Table 9.18 defines the Greybus Control Bundle Deactivate Response payload. The Response contains a one-byte status value indicating the result of the Operation. Valid status values are defined in Table 9.21.

The AP shall verify both the Greybus return value and the Bundle PM status upon reception of the Response. Only when the Greybus Operation returns GB_OP_SUCCESS and the Bundle Deactivate Response contains GB_CONTROL_BUNDLE_PM_OK may the Bundle be considered powered off. Any other combination indicates an error.

The AP shall re-establish the Connections (as described in *Non-Control Connection Establishment*) if a status code indicating an error was returned in the Response in which case the Bundle shall not be considered powered off.

9.1.18 Greybus Control Bundle Activate Operation

The AP may use this Operation to request a specific Bundle to transition from the *BUNDLE_OFF* state to the *BUNDLE_ACTIVE* state.

Greybus Control Bundle Activate Request

Table 9.19 defines the Greybus Control Bundle Activate Request payload. The Request contains a one-byte Bundle ID corresponding with the Bundle IDs received in the Manifest as described in *Manifest*.

The AP may send this Request to a Bundle which is in the *BUNDLE_OFF* state. An Interface shall send a Response containing the GB_CONTROL_BUNDLE_PM_OK if the AP requests to activate a Bundle which is already active. Sending this Request to a Bundle which is in the *BUNDLE_SUSPENDED* state shall result in the Bundle Activate Response containing the GB_CONTROL_BUNDLE_PM_NA error code.

Upon reception of this Request the Bundle indicated by the bundle_id field in the Request payload shall perform implementation-defined procedures needed to exit the *BUNDLE_OFF* state.

Greybus Control Bundle Activate Response

Table 9.20 defines the Greybus Control Bundle Activate Response payload. The Response contains a one-byte return value indicating the result of the Operation. Valid return values are defined in Table 9.21.

The AP shall verify both the Greybus return value and the Bundle PM status upon reception of the Response. Only when the Greybus Operation returns GB_OP_SUCCESS and the Bundle Resume Response contains GB_CONTROL_BUNDLE_PM_OK may the Bundle be considered active. Any other combination indicates an error.

Offset	Field	Size	Value	Description
0	status	1	Number	Bundle PM status (one of the values defined in Table 9.21)

Table 9.20: Control Protocol Bundle Activate Response

Mode	Value	Description
GB_CONTROL_BUNDLE_PM_OK	0x00	Bundle power state change was successful
GB_CONTROL_BUNDLE_PM_INVAL	0x01	Invalid Bundle ID
GB_CONTROL_BUNDLE_PM_BUSY	0x02	Request rejected due to concurrent operations
GB_CONTROL_BUNDLE_PM_FAIL	0x03	Bundle power state change failed due to an internal error
GB_CONTROL_BUNDLE_PM_NA	0x04	Operation not applicable e.g. requested suspend for an already suspended Bundle

Table 9.21: Control Protocol Bundle Power Management Status Values

If the Response does not indicate an error, the AP may establish Greybus Connections on all CPorts associated with this Bundle as described in *Non-Control Connection Establishment*.

9.1.19 Greybus Control Interface Suspend Prepare Operation

The AP uses this Operation during the *Suspend (ENUMERATED → SUSPENDED)* transition to request the Interface to enter a low-power mode after it detects a subsequent UniPro link hibernation.

In this mode, some internal context may be preserved in an implementation-defined way, allowing for a quick transition back to the *ENUMERATED* state.

The Interface Suspend Prepare Request shall not be sent by the AP unless all Bundles associated with this Interface are in the *BUNDLE_SUSPENDED* or *BUNDLE_OFF* state.

There is no Control Interface Resume Prepare Operation - the Resume Operation is handled entirely by the *Greybus SVC Interface Resume Operation*.

Greybus Control Interface Suspend Prepare Request

The Control Interface Suspend Prepare Request has no payload.

Upon reception of this Request the Interface shall verify that it is not already being suspended or powered down, that all Bundles associated with it are in the *BUNDLE_SUSPENDED* or *BUNDLE_OFF* state and that it is subsequently able to detect if its UniPort-M enters the Hibernate state.

If all above conditions are met, the Interface shall respond with the GB_CONTROL_INTF_PM_OK status and ensure that if Hibernate entry occurs, it shall proceed with the Suspend process defined in *Suspend (ENUMERATED → SUSPENDED)*.

The Interface shall still continue to respond to incoming Control Requests when waiting for the UniPort-M Hibernate.

Greybus Control Interface Suspend Prepare Response

Table 9.22 defines the Greybus Control Interface Suspend Response payload. The Response contains a one-byte return value indicating the result of the Operation. Valid return values are defined in Table 9.24.

The AP shall verify both the Greybus return value and the Bundle PM status upon reception of the Response. Only when the Greybus Operation returns GB_OP_SUCCESS and the Interface Suspend Response contains GB_CONTROL_INTF_PM_OK may the AP continue suspending the Interface. Any other combination indicates an error.

Offset	Field	Size	Value	Description
0	status	1	Number	Interface PM status (one of the values in Table 9.24)

Table 9.22: Control Protocol Interface Suspend Response

If the returned PM status is different than GB_CONTROL_INTF_PM_OK, the Interface cannot be suspended at this time. If the returned status code is GB_CONTROL_INTF_PM_BUSY, the Interface is already being suspended or powered down in which case the AP shall not retry.

If the status code is GB_CONTROL_INTF_PM_NA, one or more Bundles are still in the *BUNDLE_ACTIVE* state in which case the AP may retry after making sure all Bundles are suspended or deactivated or abandon the Suspend Operation. If the Operations still fails after a finite, implementation-defined number of retries, then the *Suspend (ENUMERATED → SUSPENDED)* procedure shall be considered as failed.

9.1.20 Greybus Control Interface Deactivate Prepare Operation

The AP uses this Operation during the *Power Down (ENUMERATED → OFF)* transition to request the bridge to power down after it detects a subsequent UniPro link hibernation (see *Power Down (ENUMERATED → OFF)*).

The Interface Deactivate Prepare Request shall not be sent by the AP unless all Bundles associated with this Interface are in the *BUNDLE_OFF* state.

There is no Control Interface Activate Operation - the Activate Operation is handled by the SVC using the *Greybus SVC Interface Activate Operation*.

Greybus Control Interface Deactivate Prepare Request

The Control Interface Deactivate Prepare Request has no payload.

Upon reception of this Request the Interface shall verify that it is not already being powered down or suspended, that all Bundles associated with it are in *BUNDLE_OFF* state and that it is subsequently able to detect if its UniPort-M enters the Hibernate state.

If all above conditions are met, the Interface shall respond with the GB_CONTROL_INTF_PM_OK status and ensure that if Hibernate entry occurs, it shall proceed with the Power Down process defined in *Power Down (ENUMERATED → OFF)*.

The Interface shall still continue to respond to incoming Control Requests when waiting for the UniPort-M Hibernate.

Greybus Control Interface Deactivate Prepare Response

Table 9.23 defines the Greybus Control Interface Deactivate Prepare Response payload. The Response contains a one-byte return value indicating the result of the Operation. Valid return values are defined in Table 9.24.

The AP shall verify both the Greybus return value and the Bundle PM status upon reception of the Response. Only when the Greybus Operation returns GB_OP_SUCCESS and the Interface Deactivate Prepare Response contains GB_CONTROL_INTF_PM_OK may the AP commence with powering down the Interface. Any other combination indicates an error.

If the returned PM status is different than GB_CONTROL_INTF_PM_OK, the Interface cannot be powered down at this time. If the returned status code is GB_CONTROL_INTF_PM_BUSY, the Interface is already being suspended or powered down in which case the AP shall not retry.

Offset	Field	Size	Value	Description
0	status	1	Number	Interface PM status (one of the values in Table 9.24)

Table 9.23: Control Protocol Interface Deactivate Prepare Response

Mode	Value	Description
GB_CONTROL_INTF_PM_OK	0x00	The AP can continue with the Interface power mode change
GB_CONTROL_INTF_PM_BUSY	0x01	Request rejected due to concurrent operations
GB_CONTROL_INTF_PM_NA	0x02	Some bundles associated with this Interface are in a wrong state

Table 9.24: Control Protocol Interface Power Management Return Values

If the status code is GB_CONTROL_INTF_PM_NA, one or more Bundles are still in the *BUNDLE_ACTIVE* or *BUNDLE_SUSPENDED* state in which case the AP may retry after making sure all Bundles are suspended or deactivated or abandon the Deactivate Operation. If the Operation still fails after a finite, implementation-defined number of retries, then the AP may continue the *Power Down (ENUMERATED → OFF)* procedure, which will result in a forceful power down.

9.1.21 Greybus Control Interface Hibernate Abort Operation

The AP may use this Operation to abort a previous Control Interface Suspend or Control Interface Deactivate Prepare Operation.

Greybus Control Interface Hibernate Abort Request

The Greybus Control Interface Hibernate Abort Request has no payload.

The AP shall not send this request to an Interface which is not currently being suspended or powered down. The AP shall also not send this Request to an Interface for which it already requested the SVC to hibernate the UniPro link.

Upon reception of this Request the Interface shall stop waiting for the UniPort-M Hibernate and undo any implementation-defined procedures it performed in order to prepare for the power state transition.

This Operation halts both the Suspend and Power Down process.

Greybus Control Interface Hibernate Abort Response

The Greybus Control Interface Hibernate Abort Response has no payload.

Upon reception of this Response the AP may re-establish any Non-Control Connections it may have closed before issuing the Suspend or Deactivate Request.

9.2 SVC Protocol

The AP Module is required to provide a CPort that uses the SVC Protocol on an Interface. The AP Module does not have a control connection, but instead implements the SVC protocol using the reserved Control CPort ID. At initial power-on, the SVC sets up a UniPro connection from one of its CPorts to the AP Module Interface's SVC CPort.

The SVC has direct control over and responsibility for the *Frame*, including detecting when modules are present, configuring the UniPro switch, powering module Interfaces, providing the frame-time and attaching and detaching modules. The AP Module controls the Frame through operations sent over the SVC connection. And the SVC informs the AP Module about Frame events (such as the presence of a new module, or notification of changing power conditions).

Conceptually, the operations in the Greybus SVC Protocol are:

`int cport_shutdown(u8 phase);`
See *Common Greybus Protocol CPort Shutdown Operation*.

`int version(u8 offer_major, u8 offer_minor, u8 *major, u8 *minor);`
Refer to *Common Greybus Protocol Version Operation*.

`int svc_hello(u16 frame_generation, u16 frame_variant, u8 intf_id);`
This Operation is used at initial power-on, sent by the SVC to inform the AP of its environment. After version negotiation, it is the next operation initiated by the SVC sent at initialization. The descriptor describes details of the Frame's environment such as number, placement, and features of interface blocks, etc.

`int dme_peer_get(u8 intf_id, u16 attribute, u16 selector, u16 *result_code, u32 *value);`
This Operation is used by the AP to direct the SVC to perform a UniPro DME peer get on its behalf. The SVC returns the value of the DME attribute requested.

`int dme_peer_set(u8 intf_id, u16 attribute, u16 selector, u32 value, u16 *result_code);`
This Operation is used by the AP to direct the SVC to perform a UniPro DME peer set on its behalf.

`int route_create(u8 intf1_id, u8 dev1_id, u8 intf2_id, u8 dev2_id);`
This Operation is used by the AP to direct the SVC to create a route for UniPro traffic between two interfaces.

`int route_destroy(u8 intf1_id, u8 intf2_id);`
This Operation is used by the AP to direct the SVC to destroy a route for UniPro traffic between two interfaces.

`int intf_device_id(u8 intf_id, u8 device_id);`
This operation is used by the AP Module to request that the SVC associate a Device ID with the given Interface.

`int intf_hotplug(u8 intf_id, u32 ddbl1_mfr_id, u32 ddbl1_prod_id, u32 ara_vend_id, u32 ara_prod_id, u64 se`
This operation is deprecated, and should not be used in new designs. See *Boot (ATTACHED → ACTIVATED)* and *Mode Switch Exit (MODE_SWITCHING → ENUMERATED)*.

`int intf_hotunplug(u8 intf_id);`
This operation is deprecated, and should not be used in new designs. See the *Greybus SVC Module Removed Operation*.

`int intf_reset(u8 intf_id);`
The SVC sends this to inform the AP Module that an active Interface needs to be reset. This might happen when the SVC has detected an error on the link, for example.

`int intf_set_power_mode(u8 intf_id, struct unipro_link_cfg *cfg);`
The AP sends this to the SVC to request that a UniPro power mode change be applied to an Interface.

`int connection_create(u8 intf1_id, u16 cport1_id, u8 intf2_id, u16 cport2_id, u8 tc, u8 flags);`
The AP Module uses this operation to request the SVC set up a UniPro connection between CPorts on two Interfaces.

`int connection_destroy(u8 intf1_id, u16 cport1_id, u8 intf2_id, u16 cport2_id);`
The AP Module uses this operation to request the SVC tear down a previously created connection.

`int timesync_enable(u8 count, u64 frame_time, u32 strobe_delay, u32 refclk);`
The AP Module uses this operation to request the SVC to enable frame-time tracking.

`int timesync_disable(void);`
The AP Module uses this operation to request the SVC stop tracking frame-time. The SVC will immediately stop tracking frame-time.

`int timesync_authoritative(void);`
The AP Module uses this operation to request the SVC to send the authoritative frame-time at each TIME_SYNC strobe.

`int timesync_wake_pins_acquire(u32 strobe_mask);`
The AP Module uses this operation to request the SVC to take control of a bit-mask of wake lines associated with the bit-mask of Interface IDs specified by the strobe_mask parameter. This is done to establish an initial state on the relevant wake lines prior to generating timesync related events.

`int timesync_wake_pins_release(void);`
The AP Module uses this operation to request the SVC to release any wake lines currently reserved for time-sync operations.

`int timesync_ping(u64 *frame_time);`
The AP Module uses this operation to request the SVC to generate a single pulse on a bit-mask of wake lines communicated to SVC by a prior timesync_wake_pins_acquire() operation. SVC will return the authoritative frame-time of the timesync_ping() to the AP Module in the response phase of the operation.

`int module_eject(u8 primary_intf_id);`
The AP Module uses this operation to request the SVC to perform the necessary action to eject a Module having the given primary interface id.

`int ping(void);`
An exchange of Messages with empty Greybus payloads.

`int pwrmon_rail_count_get(u8 *rail_count);`
The AP uses this operation to retrieve the number of power rails for which power measurements are available.

`int pwrmon_rail_names_get(u8 **rails_buf);`
The AP uses this operation to retrieve the list of names of all supported power rails.

`int pwrmon_sample_get(u8 rail_id, u8 type, u8 *result, u32 *measurement);`
The AP uses this operation to retrieve a single measurement (current, voltage or power) for a single rail.

`int pwrmon_intf_sample_get(u8 intf_id, u8 type, u8 *result, u32 *measurement);`
The AP uses this operation to retrieve a single measurement (current, voltage or power) for the specified interface.

`int power_down(void);`
The AP uses this operation to power down the SVC and all the devices it controls.

`int module_inserted(u8 primary_intf_id, u8 intf_count, u16 flags);`
The SVC uses this operation to notify the AP Module of the presence of a newly inserted Module. It sends the request after it has determined the size and position of the Module in the Frame.

`int module_removed(u8 primary_intf_id);`
The SVC uses this operation to notify the AP Module that a Module that was previously the subject of a Greybus SVC Module Inserted Operation has been removed.

`int intf_vsys_enable(u8 intf_id, u8 *result);`
The AP uses this Operation to request the SVC to set Interface State intf_id's *V_SYS* to V_SYS_ON.

`int intf_vsys_disable(u8 intf_id, u8 *result);`
The AP uses this Operation to request the SVC to set Interface State intf_id's *V_SYS* to V_SYS_OFF.

`int intf_refclk_enable(u8 intf_id, u8 *result);`
The AP uses this Operation to request the SVC to set Interface State intf_id's *REFCLK* to REFCLK_ON.

`int intf_refclk_disable(u8 intf_id, u8 *result);`
The AP uses this Operation to request the SVC to set Interface State intf_id's *REFCLK* to REFCLK_OFF.

`int intf_unipro_enable(u8 intf_id, u8 *result);`
The AP uses this Operation to request the SVC to set Interface State intf_id's *UNIPRO* to UPRO_DOWN.

`int intf_unipro_disable(u8 intf_id, u8 *result);`
The AP uses this Operation to request the SVC to set Interface State intf_id's *UNIPRO* to UPRO_OFF.

`int intf_activate(u8 intf_id, u8 *intf_type);`
The AP uses this Operation to request that the SVC attempt to activate an Interface for communication via Greybus.

`int intf_resume(u8 intf_id);`
The AP uses this Operation to request that the SVC attempt to resume an Interface which is in a low power mode into a state where it can again communicate via Greybus.

`int intf_mailbox_event(u8 intf_id, u16 result_code, u32 mailbox);`
The SVC uses this Operation to inform the AP that an Interface State's *MAILBOX* has changed value.

`int intf_oops(u8 intf_id, u8 reason);`
The SVC uses this Operation to inform the AP that an Interface has experienced a fatal error.

`int intf_vchg_enable(u8 intf_id, u8 *result);`
The AP uses this Operation to request the SVC to set *V_CHG* sub-state of *Interface States* for intf_id to V_CHG_ON.

`int intf_vchg_disable(u8 intf_id, u8 *result);`
The AP uses this Operation to request the SVC to set *V_CHG* sub-state of *Interface States* for intf_id to V_CHG_OFF.

int `intf_set_vsys_power_limit`(u8*intf_id*, u16*mw*)
The AP Module uses this Operation to set a limit for the power delivered to an Interface.

9.2.1 Greybus SVC Operations

All SVC Operations are contained within a Greybus SVC request message. Every SVC request results in a matching response. The request and response messages for each SVC Operation are defined below.

Table 9.25 defines the Greybus SVC Protocol Operation types and their values. Both the request type and response type values are shown.

9.2.2 Greybus SVC Protocol Operation Status

The SVC Protocol defines a common set of status values which are embedded in some Operation Response payload fields, and are defined in Table 9.26. These status values are used to signal errors specific to SVC Protocol.

SVC Operation Type	Request Value	Response Value
CPort Shutdown	0x00	0x80
Protocol Version	0x01	0x81
SVC Hello	0x02	0x82
Interface Device ID	0x03	0x83
Interface Hotplug (deprecated)	0x04	0x84
Interface Hot Unplug (deprecated)	0x05	0x85
Interface Reset	0x06	0x86
Connection Create	0x07	0x87
Connection Destroy	0x08	0x88
DME Peer Get	0x09	0x89
DME Peer Set	0x0a	0x8a
Route Create	0x0b	0x8b
Route Destroy	0x0c	0x8c
TimeSync Enable	0x0d	0x8d
TimeSync Disable	0x0e	0x8e
TimeSync Authoritative	0x0f	0x8f
Interface Set Power Mode	0x10	0x90
Module Eject	0x11	0x91
Reserved	0x12	N/A
Ping	0x13	0x93
Power Monitor Get Rail Count	0x14	0x94
Power Monitor Get Rail Names	0x15	0x95
Power Monitor Get Sample	0x16	0x96
Power Monitor Interface Get Sample	0x17	0x97
TimeSync Wake Pins Acquire	0x18	0x98
TimeSync Wake Pins Release	0x19	0x99
TimeSync Ping	0x1a	0x9a
Power Down	0x1d	0x9d
Reserved	0x1e	0x9e
Module Inserted	0x1f	0x9f
Module Removed	0x20	0xa0
Interface V_SYS Enable	0x21	0xa1
Interface V_SYS Disable	0x22	0xa2
Interface REFCLK Enable	0x23	0xa3
Interface REFCLK Disable	0x24	0xa4
Interface UNIPRO Enable	0x25	0xa5
Interface UNIPRO Disable	0x26	0xa6
Interface Activate	0x27	0xa7
Interface Resume	0x28	0xa8
Interface Mailbox Event	0x29	0xa9
Interface Oops	0x2a	0xaa
Interface V_CHG Enable	0x2b	0xab
Interface V_CHG Disable	0x2c	0xac
Interface Set V_SYS Power Limit	0x2d	0xad
Reserved	0x30	0xb0
Reserved	0x31	N/A
Reserved	0x32	0xb2
(all other values reserved)	0x33..0x7e	0xb3..0xfe
Invalid	0x7f	0xff

Table 9.25: SVC Operation Types

Status	Value	Meaning
GB_SVC_OP_SUCCESS	0x00	SVC Protocol Operation completed successfully
GB_SVC_OP_UNKNOWN_ERROR	0x01	Unknown error occured
GB_SVC_INTF_NOT_DETECTED	0x02	DETECT is not DETECT_ACTIVE
GB_SVC_INTF_NO_UPRO_LINK	0x03	UNIPRO is not UPRO_UP
GB_SVC_INTF_UPRO_NOT_DOWN	0x04	UNIPRO is not UPRO_DOWN
GB_SVC_INTF_UPRO_NOT_HIBERNATED	0x05	UNIPRO is not UPRO_HIBERNATE
GB_SVC_INTF_NO_V_SYS	0x06	V_SYS is not V_SYS_ON
GB_SVC_INTF_V_CHG	0x07	V_CHG is V_CHG_ON
GB_SVC_INTF_WAKE_BUSY	0x08	WAKE is not WAKE_UNSET
GB_SVC_INTF_NO_REFCLK	0x09	REFCLK is not REFCLK_ON
GB_SVC_INTF_RELEASING	0x0a	RELEASE is RELEASE_ASSERTED
GB_SVC_INTF_NO_ORDER	0x0b	ORDER is ORDER_UNKNOWN
GB_SVC_INTF_MBOX_SET	0x0c	MAILBOX is not MAILBOX_NONE
GB_SVC_INTF_BAD_MBOX	0x0d	Interface set MAILBOX to illegal value
GB_SVC_INTF_OP_TIMEOUT	0x0e	SVC Interface operation timed out
GB_SVC_PWRMON_OP_NOT_PRESENT	0x0f	Measurable power rails are not present
GB_SVC_PWRMON_ERROR	0x10	Error occurred while configuring power rail
Reserved	0x11 to 0xff	Reserved for future use

Table 9.26: SVC Protocol Status Values

9.2.3 Greybus SVC CPort Shutdown Operation

The Greybus SVC CPort Shutdown Operation is the *Common Greybus Protocol CPort Shutdown Operation* for the SVC Protocol.

9.2.4 Greybus SVC Protocol Version Operation

The Greybus SVC Protocol Version Operation is the *Common Greybus Protocol Version Operation* for the SVC Protocol.

Greybus implementations adhering to the Protocol specified herein shall specify the value 0 for the version_major and 1 for the version_minor fields found in this Operation's request and response messages.

9.2.5 Greybus SVC Hello Operation

The Greybus SVC Hello Operation is sent by the SVC to the AP at power-on to inform the AP of its environment.

Greybus SVC Hello Request

Table 9.27 defines the Greybus SVC Hello Request payload. This Operation is used at initial power-on, sent by the SVC to inform the AP of its environment. After version negotiation, it is the next Operation sent by the SVC sent at initialization. The descriptor describes details of the *Frame* environment and location of the AP interface.

Before sending the SVC Hello Request, the SVC shall ensure that all *Interface States* in the Greybus System are either *ATTACHED* or *DETACHED*.

Offset	Field	Size	Value	Description
0	frame_variant	2	Number	Frame Variant within the Generation
2	intf_id	1	Number	AP Interface ID

Table 9.27: SVC Protocol SVC Hello Request

Offset	Field	Size	Value	Description
0	intf_id	1	Number	Interface ID
1	attr	2	Number	UniPro DME Attribute
3	selector	2	Number	UniPro DME selector

Table 9.28: SVC Protocol DME Peer Get Request

Greybus SVC Hello Response

The Greybus SVC Hello response contains no payload.

During the initialization of a Greybus System, after receiving a successful SVC Hello Response from the AP, the SVC shall attempt to exchange a sequence of *Module Inserted* Operations with the AP.

9.2.6 Greybus SVC DME Peer Get Operation

The Greybus SVC DME Peer Get Operation is sent by the AP to the SVC to direct the SVC to perform a UniPro DME Peer Get on an Interface.

Greybus SVC DME Peer Get Request

Table 9.28 defines the Greybus SVC DME Peer Get Request payload. This request may be sent by the AP to query specific attributes located in the UniPro stack of an Interface. The SVC returns the value of the DME attribute requested.

Upon receiving the request, the SVC shall check that the *Interface State* with ID intf_id has DETECT equal to DETECT_ACTIVE, and UNIPRO equal to UPRO_UP.

If these conditions do not hold, the SVC cannot satisfy the request, and shall send a response signaling an error as described below. The SVC shall take no further action related to such an unsatisfiable request beyond sending the response.

Otherwise, the SVC shall attempt to retrieve the value of the UniPro DME attribute with Attribute ID given by the attr field, with selector index given by the selector field.

Greybus SVC DME Peer Get Response

Table 9.29 defines the Greybus SVC DME Peer Get Operation Response payload. If the *Greybus Operation Status* is not GB_OP_SUCCESS, the values of the response payload fields are undefined and shall be ignored.

If the status field in the Operation Response payload is not GB_SVC_OP_SUCCESS, values in all other fields of the Operation Response payload are undefined and shall be ignored. The SVC shall return the following errors in the status field of the Operation Response payload depending on the sub-state values of the *Interface States* with Interface ID given by intf_id in the request payload:

- If DETECT is not DETECT_ACTIVE, the response shall have status GB_SVC_INTF_NOT_DETECTED.

Offset	Field	Size	Value	Description
0	status	1	Number	*Greybus SVC Protocol Operation Status*
1	result_code	2	Number	UniPro DME Peer Get ConfigResultCode
3	attr_value	4	Number	UniPro DME Peer Get DME Attribute value

Table 9.29: SVC Protocol DME Peer Get Response

Offset	Field	Size	Value	Description
0	intf_id	1	Number	Interface ID
1	attr	2	Number	UniPro DME Attribute
3	selector	2	Number	UniPro DME selector
5	value	4	Number	UniPro DME Attribute value to set

Table 9.30: SVC Protocol DME Peer Set Request

- If UNIPRO is not UPRO_UP, the response shall have status GB_SVC_INTF_NO_UPRO_LINK.

If during the handling of the request, the SVC is unable to exchange the UniPro frames required to retrieve a ConfigResultCode or attribute value from the peer identified in the request, the status field in Operation Response payload shall be GB_SVC_OP_UNKNOWN_ERROR. When this occurs, the value of the UNIPRO sub-state for the Interface identified in the request is unpredictable.

If the *Greybus Operation Status* is GB_OP_SUCCESS and the status field in Operation Response payload is GB_SVC_OP_SUCCESS, the Greybus DME Peer Get response contains the ConfigResultCode as defined in the UniPro specification, as well as the value of the attribute, if applicable.

9.2.7 Greybus SVC DME Peer Set Operation

The Greybus SVC DME Peer Set Operation is sent by the AP to the SVC to direct the SVC to perform a UniPro DME_PEER_SET on an Interface.

Greybus SVC DME Peer Set Request

Table 9.30 defines the Greybus SVC DME Peer Set Request payload. This request may be sent by the AP to set specific attributes located in the UniPro stack of an Interface.

Upon receiving the request, the SVC shall check that the *Interface State* with ID intf_id has DETECT equal to DETECT_ACTIVE, and UNIPRO equal to UPRO_UP.

If these conditions do not hold, the SVC cannot satisfy the request, and shall send a response signaling an error as described below. The SVC shall take no further action related to such an unsatisfiable request beyond sending the response.

Otherwise, the SVC shall attempt to set the value of the UniPro DME attribute with Attribute ID given by the attr field, with selector index given by the selector field, to the value given by the value field.

Greybus SVC DME Peer Set Response

Table 9.31 defines the Greybus SVC DME Peer Set Response payload. If the *Greybus Operation Status* is not GB_OP_SUCCESS, the values of the response payload fields are undefined and shall be ignored.

If the status field in the Operation Response payload is not GB_SVC_OP_SUCCESS, values in all other fields of the Operation Response payload are undefined and shall be ignored. The SVC shall return the

Offset	Field	Size	Value	Description
0	status	1	Number	*Greybus SVC Protocol Operation Status*
1	result_code	2	Number	UniPro DME Peer Set ConfigResultCode

Table 9.31: SVC Protocol DME Peer Set Response

Offset	Field	Size	Value	Description
0	intf1_id	1	Number	First Interface
1	dev1_id	1	Number	First Interface Device ID
2	intf2_id	1	Number	Second Interface
3	dev2_id	1	Number	Second Interface Device ID

Table 9.32: SVC Protocol Route Create Request

following errors in the status field of the Operation Response payload depending on the sub-state values of the *Interface States* with Interface ID given by intf_id in the request payload:

- If DETECT is not DETECT_ACTIVE, the response shall have status GB_SVC_INTF_NOT_DETECTED.
- If UNIPRO is not UPRO_UP, the response shall have status GB_SVC_INTF_NO_UPRO_LINK.

If during the handling of the request, the SVC is unable to exchange the UniPro frames required to retrieve a ConfigResultCode or attribute value from the peer identified in the request, the status field in Operation Response payload shall be GB_SVC_OP_UNKNOWN_ERROR. When this occurs, the value of the UNIPRO sub-state for the Interface identified in the request is unpredictable.

If the *Greybus Operation Status* is GB_OP_SUCCESS and the status field in Operation Response payload is GB_SVC_OP_SUCCESS, the Greybus DME Peer Set response contains the ConfigResultCode for the attribute write as defined in the UniPro specification.

9.2.8 Greybus SVC Route Create Operation

The Greybus SVC Protocol Route Create Operation allows the AP Module to request a route be established for UniPro traffic between two Interfaces.

While handling this Operation request, the SVC may attempt to create a *route* within the Frame. This is a necessary condition for UniPro Messages to subsequently be exchanged between the UniPorts attached to the Interface Blocks identified by the request.

However, creation of a route is not a sufficient condition for Message exchange. In order to exchange UniPro Messages between the two Interfaces, a successful *Greybus SVC Connection Create Operation* between the two interfaces is required as well. Additional Operations are required to establish a Greybus Connection, as described in *Connection Management*.

Greybus SVC Route Create Request

Table 9.32 defines the Greybus SVC Route Create request payload. The request supplies the Interface IDs and Device IDs of two Interfaces to be connected.

Upon receiving the request, the SVC shall check that the *Interface States* with IDs intf1_id and intf2_id have DETECT equal to DETECT_ACTIVE, and UNIPRO equal to UPRO_UP.

Offset	Field	Size	Value	Description
0	status	1	Number	*Greybus SVC Protocol Operation Status*

Table 9.33: SVC Protocol Route Create Response

Offset	Field	Size	Value	Description
0	intf1_id	1	Number	First Interface
1	intf2_id	1	Number	Second Interface

Table 9.34: SVC Protocol Route Destroy Request

If these conditions do not hold, the SVC cannot satisfy the request, and shall send a response signaling an error as described below. The SVC shall take no further action related to such an unsatisfiable request beyond sending the response.

Otherwise, the SVC shall attempt to create the specified route.

Greybus SVC Route Create Response

Table 9.33 defines the Greybus SVC Route Create Response payload. If the *Greybus Operation Status* is not GB_OP_SUCCESS, the value of the Response payload field is undefined and shall be ignored.

The SVC shall return the following errors in the status field of the Operation Response payload depending on the sub-state values of the *Interface States* with Interface ID given by intf1_id and intf2_id in the Request payload.

- If DETECT is not DETECT_ACTIVE in both Interface States, the response shall have status GB_SVC_INTF_NOT_DETECTED.
- If DETECT is DETECT_ACTIVE in both Interface States, and UNIPRO is not UPRO_UP in both Interface States, the response shall have status GB_SVC_INTF_NO_UPRO_LINK.

Regardless of the Response status value, the Greybus SVC Route Create Operation shall have no effect on either the UNIPRO sub-state of either Interface identified by the request, or the value of any of the UniPro DME attributes for the Interfaces identified by the request.

9.2.9 Greybus SVC Route Destroy Operation

The Greybus SVC Protocol Route Destroy Operation allows the AP Module to request a route be torn down for UniPro traffic between two Interfaces.

While handling this Operation, the SVC may tear down a previously created *route* within the Frame. This is a sufficient condition for preventing subsequent UniPro Messages from being exchanged between the UniPorts attached to the Interface Blocks identified by the request; however, additional Operations are required to completely release resources acquired during Greybus Connection establishment, as described in *Connection Management*.

Greybus SVC Route Destroy Request

Table 9.34 defines the Greybus SVC Route Destroy request payload. The request supplies the Interface IDs of two Interfaces between which the route should be destroyed.

Upon receiving the request, the SVC shall attempt to destroy the specified route.

Offset	Field	Size	Value	Description
0	intf_id	1	Number	Interface ID whose Device ID is being assigned
1	device_id	1	Number	5-bit UniPro Device ID for Interface

Table 9.35: SVC Protocol Device ID Request

Greybus SVC Route Destroy Response

The Greybus SVC Protocol Route Destroy response contains no payload.

Regardless of the response status value, the Greybus SVC Route Destroy Operation shall have no effect on either the UNIPRO sub-state of either Interface identified by the request, or the value of any of the UniPro DME attributes for the Interfaces identified by the request.

9.2.10 Greybus SVC Interface Device ID Operation

The Greybus SVC Interface Device ID Operation is used by the AP Module to request the SVC associate a device id with an Interface. The device id is used by the UniPro switch to determine how packets should be routed through the network. The AP Module is responsible for managing the mapping between Interfaces and UniPro device ids.

Greybus supports 5-bit UniPro Device IDs. Device ID 0 and 1 are reserved for the SVC and primary AP Interface respectively.

The AP shall manage Device IDs of any attached Modules using this operation during *Connection Management*.

Greybus SVC Interface Device ID Request

Table 9.35 defines the Greybus SVC Interface Device ID Request payload.

The Greybus SVC Interface Device ID Request shall only be sent by the AP Module to the SVC. It supplies the 5-bit Device ID that the SVC will associate with the indicated Interface. The AP Module can remove the association of an Interface with a Device ID by setting the device_id field in the request payload to zero. The AP shall not assign a (non-zero) Device ID to an Interface that the SVC has already associated with an Interface, and shall not clear the Device ID of an Interface that has no Device ID assigned.

Note that assigning a Device ID to an Interface does not cause the SVC to set up any routes for that Device ID. Routes are set up only as needed when a connection involving a Device ID are created, and removed when an Interface's last connection is destroyed.

Upon receiving the request, the SVC shall check that the *Interface State* with ID intf_id has DETECT equal to DETECT_ACTIVE, and UNIPRO equal to UPRO_UP.

If these conditions do not hold, the SVC cannot satisfy the request, and shall send a response signaling an error as described below. The SVC shall take no further action related to such an unsatisfiable request beyond sending the response.

Otherwise, the SVC shall attempt to set the UniPro Device ID of the UniPort connected to corresponding Interface Block to device_id, and to mark the UniPro Device ID as valid. This sequence may change the values of UniPro DME attributes on the UniPort the Interface Block identified in the request.

Offset	Field	Size	Value	Description
0	status	1	Number	*Greybus SVC Protocol Operation Status*

Table 9.36: SVC Protocol Interface Device Id Response

Greybus SVC Interface Device ID Response

Table 9.36 defines the Greybus SVC Interface Device ID Response payload. If the Response message header has *Greybus Operation Status* not equal to GB_OP_SUCCESS, the value of the Response payload field is undefined and shall be ignored.

The SVC shall return the following errors in the status field of the Operation Response payload depending on the sub-state values of the *Interface State* with Interface ID given by intf_id in the Request payload.

- If DETECT is not DETECT_ACTIVE, the response shall have status GB_SVC_INTF_NOT_DETECTED.
- If UNIPRO is not UPRO_UP, the response shall have status GB_SVC_INTF_NO_UPRO_LINK.

If the SVC fails to set the Device ID due to an error on a UniPro link, the status field in the Operation Response payload shall be GB_SVC_OP_UNKNOWN_ERROR. When this occurs, the value of the Device ID, as well as its validity, are unpredictable, as is the value of the UNIPRO sub-state of the *Interface State* with Interface ID given by the intf_id in Request payload.

9.2.11 Greybus SVC Interface Hotplug Operation (Deprecated)

Note: This operation is deprecated, and should not be used in new designs.

Boot (ATTACHED → ACTIVATED) and *Mode Switch Exit (MODE_SWITCHING → ENUMERATED)* should be used for any new designs.

When the SVC first detects that a module is present on an Interface, it sends an Interface Hotplug Request to the AP Module. The hotplug request is sent after the Interface's UniPro link has been established. The request includes some additional information known by the SVC about the discovered Interface (such as the vendor and product ID).

Greybus SVC Interface Hotplug Request

Table 9.37 defines the Greybus SVC Interface Hotplug Request payload.

The Greybus SVC hotplug request is sent only by the SVC to the AP Module. The Interface ID informs the AP Module which Interface now has a module present, and supplies information (such as the vendor and model numbers) the SVC knows about the Interface. Exactly one hotplug event shall be sent by the SVC for a module when it has been inserted (or if it was found to be present at initial power-on).

Greybus SVC Interface Hotplug Response

The Greybus SVC hotplug response message contains no payload.

Offset	Field	Size	Value	Description
0	intf_id	1	Number	Interface that now has a module present
1	ddbl1_mfr_id	4	Number	UniPro DDB Level 1 Manufacturer ID
5	ddbl1_prod_id	4	Number	UniPro DDB Level 1 Product ID
9	ara_vend_id	4	Number	Ara Vendor ID
13	ara_prod_id	4	Number	Ara Product ID
17	serial_number	8	Number	Module serial number that uniquely identifies modules with same ARA VID/PIDs

Table 9.37: SVC Protocol Hotplug Request

Offset	Field	Size	Value	Description
0	intf_id	1	Number	Interface that no longer has an attached module

Table 9.38: SVC Protocol Hot Unplug Request

9.2.12 Greybus SVC Interface Hot Unplug Operation (Deprecated)

Note: This operation is deprecated, and should not be used in new designs.

The *Greybus SVC Module Removed Operation* should be used for any new designs.

The SVC sends this to the AP Module to tell it that an Interface that was previously the subject of an Interface Hotplug Operation is no longer present. The SVC sends exactly one hot unplug event, for the Interface, to the AP when this occurs.

Greybus SVC Interface Hot Unplug Request

Table 9.38 defines the Greybus SVC Interface Hot Unplug Request payload.

The Greybus SVC hot unplog request is sent only by the SVC to the AP Module. The Interface ID informs the AP which Interface no longer has a module attached to it. The SVC shall ensure the hotplug event for the Interface has been successfully delivered to the AP Module before sending a hot unplug.

Greybus SVC Interface Hot Unplug Response

The Greybus SVC hot unplug response message contains no payload.

9.2.13 Greybus SVC Interface Reset Operation

The SVC sends this to the AP Module to request it reset the indicated link.

Greybus SVC Interface Reset Request

Table 9.39 defines the Greybus SVC Interface Reset Request payload.

The Greybus SVC Interface Reset Request is sent only by the SVC to the AP Module. The Interface ID informs the AP Module which Interface needs to be reset.

Offset	Field	Size	Value	Description
0	intf_id	1	Number	Interface to reset

Table 9.39: SVC Protocol Reset Request

Offset	Field	Size	Value	Description
0	intf_id	1	Number	Interface whose power mode to change
1	hs_series	1	Number	Frequency series in high speed mode; see Table 9.41
2	tx_mode	1	Number	Power mode for TX; see Table 9.42
3	tx_gear	1	Number	Gear for TX lanes
4	tx_nlanes	1	Number	Number of active TX lanes
5	tx_amplitude	1	Number	TX signal amplitude; see Table 9.43
6	tx_hs_equalizer	1	Number	HS TX signal de-emphasis; see Table 9.44
7	rx_mode	1	Number	Power mode for RX; see Table 9.42
8	rx_gear	1	Number	Gear for RX lanes
9	rx_nlanes	1	Number	Number of active RX lanes
10	flags	1	Bit mask	See Table 9.45
11	quirks	4	Bit mask	See Table 9.46
15	local_l2timerdata	24	Number	L2 timer configuration data for power mode change (local peer)
39	remote_l2timerdata	24	Number	L2 timer configuration data for power mode change (remote peer)

Table 9.40: SVC Protocol Interface Set Power Mode Request

Greybus SVC Interface Reset Response

The Greybus SVC Interface Reset response message contains no payload.

9.2.14 Greybus SVC Interface Set Power Mode Operation

The AP sends this to the SVC to request that it change the UniPro power mode for the UniPro link on an Interface.

The AP may use this Operation while an *Interface* is *ENUMERATED* to manage various features of the Link established between the Switch and the attached Module.

The AP shall additionally use this Operation in order to perform *Power Management* and certain *Error Handling* transitions in *The Interface Lifecycle*.

Greybus SVC Interface Set Power Mode Request

Table 9.40 defines the Greybus SVC Interface Set Power Mode Request payload.

The request message payload contains the interface ID for which the AP requests the power mode change, fields specifying the power mode change to apply, and a structure containing implementation-specific configuration information associated with the power mode change.

The hs_series field in the request payload allows the AP to control which rate series is used when either direction of the link is in high speed mode. The values of the hs_series field are defined in Table 9.41.

The tx_mode and rx_mode fields in the request payload allow the AP to specify a UniPro power mode for each direction of the link. The values of these fields, along with the corresponding modes, are specified in Table 9.42.

Frequency Series	Value	Description
(Reserved)	0	(Reserved for future use)
A	1	High speed series A
B	2	High speed series B
(All other values reserved)	3-255	(Reserved for future use)

Table 9.41: High Speed Frequency Series

Mode	Value	Description
(Reserved)	0x00	(Reserved for future use)
UNIPRO_FAST_MODE	0x01	Fast (HS) mode
UNIPRO_SLOW_MODE	0x02	Slow (PWM) mode
(Reserved)	0x03	(Reserved for future use)
UNIPRO_FAST_AUTO_MODE	0x04	Fast auto mode
UNIPRO_SLOW_AUTO_MODE	0x05	Slow auto mode
(Reserved)	0x06	(Reserved for future use)
UNIPRO_MODE_UNCHANGED	0x07	Leave mode unchanged
(Reserved)	0x08-0x10	(Reserved for future use)
UNIPRO_HIBERNATE_MODE	0x11	Hibernate mode
UNIPRO_OFF_MODE	0x12	Link is off
(Reserved)	0x13-0xFF	(Reserved for future use)

Table 9.42: |unipro| power modes

The tx_amplitude field in the request payload allows the AP to specify the TX path signal amplitude of a UniPro link. It applies to both local and remote peers. The values of this field, along with the corresponding modes, are specified in Table 9.43.

The tx_hs_equalizer field in the request payload allows the AP to specify a de-emphasis value for the TX path of a UniPro link. It applies to both local and remote peers. It is only relevant in high speed (HS) mode, and ignored in slow (PWM) mode. The values of this field, along with the corresponding modes, are specified in Table 9.44.

The flags field in the request payload is a bit mask which allows the AP to request the SVC to update extra UniPro power mode settings. The mask values for the flags field are defined in Table 9.45.

The quirks field in the request payload is a bit mask which allows the AP to request behavior from the SVC which may deviate in some way from the UniPro specification. The mask values for the quirks field are defined in Table 9.46.

The local_l2timerdata and remote_l2timerdata fields in the request payload allow the AP to configure L2 timer values of the UniPro link. local_l2timerdata and remote_l2timerdata fields apply respectively to the local and remote peers of the UniPro link. The content of this structure is defined in the UniPro specification version 1.6, Table 102. All integer values in Table 102 are stored as 16-bit little-endian values.

Mode	Value	Description
(Reserved)	0x0	(Reserved for future use)
SMALL_AMPLITUDE	0x01	Select small TX signal amplitude
LARGE_AMPLITUDE	0x02	Select large TX signal amplitude
(all other values reserved)	0x03-0xFF	(Reserved for future use)

Table 9.43: TX path signal amplitudes

Mode	Value	Description
NO_DE_EMPHASIS	0x0	Disable de-emphasis on HS TX path
SMALL_DE_EMPHASIS	0x01	Enable 3.5dB de-emphasis on HS TX path
LARGE_DE_EMPHASIS	0x02	Enable 6dB de-emphasis on HS TX path
(all other values reserved)	0x03-0xFF	(Reserved for future use)

Table 9.44: HS TX signal de-emphasis modes

Mode	Value	Description
RX_TERMINATION	0x01	Enable RX-direction termination
TX_TERMINATION	0x02	Enable TX-direction termination
LINE_RESET	0x04	Request Line Reset
(Reserved)	0x08	(Reserved for future use)
(Reserved)	0x10	(Reserved for future use)
SCRAMBLING	0x20	Always set HS series
(all other values reserved)	0x40-0x80	(Reserved for future use)

Table 9.45: Flags for SVC Interface Set Power Mode Request

If one or more of the following list of conditions holds, the SVC shall transmit a Greybus SVC Interface Set Power Mode Response message with status byte GB_OP_INVALID. The SVC shall make no changes to the link's power mode in any of these cases.

1. The request's hs_series field does not lie within the table of values given in Table 9.41.
2. The request's tx_mode or rx_mode field is not one of the values given in Table 9.42.
3. The request's tx_mode, rx_mode, tx_gear, rx_gear, tx_nlanes, rx_nlanes, tx_amplitude and tx_hs_equalizer do not collectively lie within the ranges defined by the UniPro specification.
4. The request's quirks field contains bits set which are reserved for future use or not supported by the SVC.

Upon receipt of a Greybus SVC Interface Set Power Mode Request, the SVC shall determine if the intf_id field in the request payload is valid, by determining if there is a UniPro link associated with the Interface given by intf_id, and whether that UniPro link is up. If so, the SVC shall attempt to change the power mode of the UniPro link at the given interface. If not, the SVC shall transmit a Greybus SVC Interface Set Power Mode Response message with the *Greybus Operation Status* in the Response message header set to GB_OP_INVALID. The SVC shall make no changes to the link's power mode in this case.

The tx_mode and rx_mode fields in the Greybus SVC Interface Set Power Mode Request determine the UniPro Power Modes of the link's transmit and receive directions, respectively. The transmit and receive directions are defined with respect to the UniPort attached to the UniPro switch. For example, tx_mode determines the UniPro power mode of the transmitter which is attached to the UniPro switch at the Interface given by intf_id; tx_mode does not refer to the transmitter within the switch itself.

If either of tx_mode or rx_mode equals UNIPRO_HIBERNATE_MODE, both shall equal UNIPRO_HIBERNATE_MODE. Under this condition, the following fields in the request payload

Mode	Value	Description
SVC_PWRM_QUIRK_HSSER	0x00000001	Always set HS series
(all other values reserved)	0x00000002-0x80000000	(Reserved for future use)

Table 9.46: Quirks for SVC Interface Set Power Mode Request

shall be ignored: hs_series, tx_gear, tx_nlanes, tx_amplitude, tx_hs_equalizer, rx_gear, rx_nlanes, flags, quirks, local_l2timerdata, remote_l2timerdata.

When reconfiguring the link power mode as a result of receiving a Greybus SVC Interface Set Power Mode Request, the SVC shall set the UniPro PA_HSSeries attribute for the link according to the hs_series field in the request payload, as defined by Table 9.41.

If the SVC_PWRM_QUIRK_HSSER bit is set in the quirks field of the request payload, the SVC shall perform this setting regardless of whether either tx_mode or rx_mode is UNIPRO_FAST_MODE or UNIPRO_FAST_AUTO_MODE. If SVC_PWRM_QUIRK_HSSER is unset, the SVC shall set PA_HSSeries if and only if one of tx_mode or rx_mode is UNIPRO_FAST_MODE or UNIPRO_FAST_AUTO_MODE.

The tx_gear and rx_gear attributes specify the gear settings for the transmit and receive directions in the new power mode configuration. The valid values for the tx_gear and rx_gear fields depend respectively on the values of tx_mode and rx_mode.

If tx_mode or rx_mode is UNIPRO_FAST_MODE or UNIPRO_FAST_AUTO_MODE, then the valid values for tx_gear or rx_gear, respectively, are one, two, and three.

If tx_mode or rx_mode is UNIPRO_SLOW_MODE or UNIPRO_SLOW_AUTO_MODE, then the valid values for tx_gear or rx_gear, respectively, are the range of integers between one and seven.

If tx_mode or rx_mode is UNIPRO_MODE_UNCHANGED, direction-specific parameters (tx_gear, tx_nlanes, SVC_PWRM_TXTERMINATION or rx_gear, rx_nlanes, SVC_PWRM_RXTERMINATION, respectively) will be ignored.

Upon receiving the request, the SVC shall check that the *Interface State* with ID intf_id has DETECT equal to DETECT_ACTIVE, and has a UNIPRO sub-state equal to UPRO_UP or UPRO_HIBERNATE.

If these conditions do not hold, the SVC shall send a response signaling an error as described below. The SVC shall take not attempt to reconfigure any UniPro links as a result of receiving such a request.

Otherwise, the SVC shall attempt to reconfigure the power mode for the UniPro link identified by the request.

When reconfiguring the link power mode as a result of receiving a Greybus SVC Interface Set Power Mode Request, the link's transmitter and/or receiver power mode shall be set to the given configuration. The *Greybus Operation Status* in the Response message header of the response to a Greybus SVC Interface Set Power Mode Request shall not be used to check the result of the power mode change operation. It shall only be used to indicate the result of the Greybus communication only. If the *Greybus Operation Status* in the Response message header of the response to a Greybus SVC Interface Set Power Mode Request is different than GB_OP_SUCCESS, it shall indicate that an error occurred and that the power mode change could not be initiated; the targeted link shall be in the same state as before the request was issued. If the *Greybus Operation Status* in the Response message header of response to a Greybus SVC Interface Set Power Mode Request is GB_OP_SUCCESS, it shall indicate that there was no Greybus communication error detected (Request and Response were successfully exchanged). However, it shall not also be considered as a successful power mode change. The status and pwr_change_result_code fields as respectively described in Table 9.47 shall be used for that unique purpose. In other words, if and only if the *Greybus Operation Status* in the Response message header is GB_OP_SUCCESS and the status field in the Greybus SVC Interface Set Power Mode Response payload as described in Table 9.47 is GB_SVC_OP_SUCCESS, the pwr_change_result_code field in the Response payload indicates the actual result of the power mode change request.

Greybus SVC Interface Set Power Mode Response

Table 9.47 defines the Greybus SVC Interface Set Power Mode Response payload. If the Response message header has the *Greybus Operation Status* not equal to GB_OP_SUCCESS, the values of the Response payload fields are undefined and shall be ignored.

Offset	Field	Size	Value	Description
0	status	1	Number	*Greybus SVC Protocol Operation Status*
1	pwr_change_result_code	1	Number	UniPro PowerChangeResultCode

Table 9.47: SVC Protocol Interface Set Power Mode Response

PowerChangeResultCode	Value	Description
PWR_OK	0	The request was accepted.
PWR_LOCAL	1	The local request was successfully applied.
PWR_REMOTE	2	The remote request was successfully applied.
PWR_BUSY	3	The request was aborted due to concurrent requests.
PWR_ERROR_CAP	4	The request was rejected because the requested configuration exceeded the Links capabilities.
PWR_FATAL_ERROR	5	The request was aborted due to a communication problem. The Link may be inoperable.
(All other values reserved)	6-255	(Reserved for future use)

Table 9.48: PowerChangeResultCode Values

If the status field in the Operation response payload as described in Table 9.47 is not GB_SVC_OP_SUCCESS, the value in the pwr_change_result_code field of the Response payload is undefined and shall be ignored. The SVC shall return the following errors in the status field of the Operation Response payload depending on the sub-state values of the *Interface State* with Interface ID given by intf_id in the Request payload:

- If DETECT is not DETECT_ACTIVE, the response shall have status GB_SVC_INTF_NOT_DETECTED.
- If UNIPRO is not UPRO_UP or UPRO_HIBERNATE, the response shall have status GB_SVC_INTF_NO_UPRO_LINK.

If the Response message header has the *Greybus Operation Status* equal to GB_OP_SUCCESS and the status field in the Operation Response payload is GB_SVC_OP_SUCCESS, the pwr_change_result_code field in the Greybus Interface Set Power Mode response message contains a PowerChangeResultCode as defined by the UniPro specification, version 1.6, Table 9. The pwr_change_result_code field indicates a successful Operation or describes the reason for the Operation failure. The values of the pwr_change_result_code field are defined in Table 9.48.

9.2.15 Greybus SVC Connection Create Operation

The AP Module sends this Operation to the SVC to request that it establish a UniPro connection between the two indicated CPorts. The SVC uses each (intf_id, cport_id) pair to determine the UniPro (DeviceID_Enc, CPortID_Enc) it represents. It is an error to attempt to create a connection using a CPort that is already in use in another connection.

Greybus SVC Connection Create Request

Table 9.49 defines the Greybus SVC Connection Create Request payload.

The Greybus SVC connection create request is sent only by the AP Module to the SVC. The first Interface ID and first CPort ID define one end of the connection to be established, and the second Interface ID and CPort ID define the other end.

CPort flags can be specified as a bitwise-or of flags in *flags*, and are defined in table 9.50.

Upon receiving the request, the SVC shall check that the *Interface States* with IDs intf1_id and intf2_id both have DETECT equal to DETECT_ACTIVE, and UNIPRO equal to UPRO_UP.

Offset	Field	Size	Value	Description
0	intf1_id	1	Number	First Interface
1	cport1_id	2	Number	CPort on first Interface
3	intf2_id	1	Number	Second Interface
4	cport2_id	2	Number	CPort on second Interface
6	tc	1	Traffic class	UniPro traffic class
7	flags	1	Connection flags	UniPro connection flags

Table 9.49: SVC Protocol Connection Create Request

Value	Flag	Description
0x01	E2EFC	Enable UniPro End-to-End Flow Control
0x02	CSD_N	Disable UniPro Controlled Segment Dropping
0x04	CSV_N	Disable UniPro CPort Safety Valve

Table 9.50: SVC Protocol Connection Create Request Flags

If these conditions do not hold, the SVC cannot satisfy the request, and shall send a response signaling an error as described below. The SVC shall take no further action related to such an unsatisfiable request beyond sending the response.

Otherwise, the SVC shall attempt to establish a UniPro connection between the CPort with ID cport1_id on Interface intf1_id, and CPort with ID cport2_id on Interface intf2_id. The SVC shall attempt to establish the connection using the Traffic Class and CPort features given by the tc and flags field in the request, respectively. This sequence may change the values of UniPro DME attributes on the UniPorts attached to each Interface Block identified in the request.

Greybus SVC Connection Create Response

Table 9.51 defines the Greybus SVC Connection Create Response. If the Response message header has the *Greybus Operation Status* not equal to GB_OP_SUCCESS, the value of the status field in the Operation Response payload is undefined and shall be ignored.

The SVC shall return the following errors in the status field of the Operation Response payload depending on the sub-state values of the *Interface States* with Interface IDs given by intf1_id and intf2_id in the Request payload:

- If DETECT is not DETECT_ACTIVE in both Interface States, the response shall have status GB_SVC_INTF_NOT_DETECTED.
- If DETECT is DETECT_ACTIVE in both Interface States, and UNIPRO is not UPRO_UP in both Interface States, the response shall have status GB_SVC_INTF_NO_UPRO_LINK.

If the SVC fails to establish a UniPro connection between the two Interfaces due to an I/O or protocol error on the UniPro links, the status field in Operation Response payload shall equal GB_SVC_OP_UNKNOWN_ERROR. When this occurs, the values of the UniPro DME attributes of one or both of the Interfaces is unpredictable, as are the values of the UNIPRO sub-state of the *Interface States* with Interface IDs given by intf1_id and intf2_id in Request payload.

9.2.16 Greybus SVC Connection Destroy Operation

The AP Module sends this to the SVC to request that a connection that was previously set up by a Connection Create Operation be torn down. The AP Module shall have sent Disconnected Control Operations to the

Offset	Field	Size	Value	Description
0	status	1	Number	*Greybus SVC Protocol Operation Status*

Table 9.51: SVC Protocol Connection Create Response

Offset	Field	Size	Value	Description
0	intf1_id	1	Number	First Interface
1	cport1_id	2	Number	CPort on first Interface
3	intf2_id	1	Number	Second Interface
4	cport2_id	2	Number	CPort on second Interface

Table 9.52: SVC Protocol Connection Destroy Request

two Interfaces prior to this call. It is an error to attempt to destroy a connection more than once.

Greybus SVC Connection Destroy Request

Table 9.52 defines the Greybus SVC Connection Destroy Request payload.

The Greybus SVC connection destroy request is sent only by the AP Module to the SVC. The two (Interface ID, CPort ID) pairs define the connection to be destroyed.

Upon receiving the request, the SVC shall check that the *Interface States* with IDs intf1_id and intf2_id both have DETECT equal to DETECT_ACTIVE, and UNIPRO equal to UPRO_UP.

If these conditions do not hold, the SVC cannot satisfy the request, and shall send a response signaling an error as described below. The SVC shall take no further action related to such an unsatisfiable request beyond sending the response.

Otherwise, the SVC shall attempt to disable the UniPro connection between the CPort with ID cport1_id on Interface intf1_id, and CPort with ID cport2_id on Interface intf2_id. This sequence may change the values of UniPro DME attributes on the UniPorts attached to each Interface Block identified in the request.

Greybus SVC Connection Destroy Response

Table 9.53 defines the Greybus SVC Connection Destroy Response payload. If the Response message header has the *Greybus Operation Status* not equal to GB_OP_SUCCESS, the value in the status field in the Operation Response payload is undefined and shall be ignored.

The SVC shall return the following errors in the status field of the Operation Response payload depending on the sub-state values of the *Interface States* with Interface IDs given by intf1_id and intf2_id in the request payload:

- If DETECT is not DETECT_ACTIVE in both Interface State, the response shall have status GB_SVC_INTF_NOT_DETECTED.
- If DETECT is DETECT_ACTIVE for both Interface States, and UNIPRO is not UPRO_UP in both Interface States, the response shall have status GB_SVC_INTF_NO_UPRO_LINK.

If the SVC fails to destroy the UniPro connection between the two Interfaces due to an I/O or protocol error on the UniPro links, the status field in Operation Response payload shall equal GB_SVC_OP_UNKNOWN_ERROR. When this occurs, the values of the UniPro DME attributes of one or both of the Interfaces is unpredictable, as are the values of the UNIPRO sub-state of the *Interface States* with Interface IDs given by intf1_id and intf2_id in Request payload.

Offset	Field	Size	Value	Description
0	status	1	Number	*Greybus SVC Protocol Operation Status*

Table 9.53: SVC Protocol Connection Destroy Response

Offset	Field	Size	Value	Description
0	count	1	Number	Number of TIME_SYNC pulses
1	frame_time	8	Number	The initial frame-time to intiailze to
9	strobe_delay	4	Number	Inter-strobe delay in milliseconds
13	refclk	4	Number	The clock rate of the frame-time counter

Table 9.54: SVC Protocol TimeSync Enable Request

9.2.17 Greybus SVC TimeSync Enable Operation

The AP Module uses this operation to request the SVC to enable frame-time tracking. After a successful timesync_enable operation the SVC will generate a pulse-train of 'count' logical TIME_SYNC strobes to the bitmask of WAKE_DETECT lines indicated by a previously communicated set of Interfaces. A delay of 'strobe_delay' microseconds will be applied between each TIME_SYNC strobe. The range of the count variable is from 1..4. The 'frame_time' parameter informs the Interface to immediately seeds its frame-time to a value given by the AP. 'frame-time. The 'refclk' parameter informs the SVC of the required clock rate to run its frame-time tracking counter at.

Greybus SVC TimeSync Enable Request

Table 9.54 defines the Greybus SVC TimeSync Enable Request payload. The request supplies the number of TIME_SYNC strobes to perform (count), the initial frame-time (frame_time), the delay between each strobe (strobe_delay) and the required clock-rate for frame-time (refclk).

Greybus SVC TimeSync Enable Response

The Greybus SVC Protocol TimeSync Enable response contains no payload.

If the Response message header status field *Greybus Operation Status* is not equal to GB_OP_SUCCESS the AP shall immediately issue a *Greybus SVC TimeSync Disable Operation* to the set of Interfaces previously indicated in the 'strobe_mask' field of the *Greybus SVC TimeSync Wake Pins Acquire Operation*. The AP shall then issue a *Greybus SVC TimeSync Wake Pins Release Operation* to the SVC.

If the Response message header status field *Greybus Operation Status* is equal to GB_OP_SUCCESS the SVC shall set the *TimeSync Pulse* sub-state for the indicated set of Interfaces to WAKE_ASSERTED and WAKE_DEASSERTED repeatedly to indicate 'count' number of *TimeSync Pulse* events. The SVC may send the response before initiating or completing the set of *TimeSync Pulse* events.

9.2.18 Greybus SVC TimeSync Disable Operation

The AP Module uses this operation to request the SVC stop tracking frame-time. The SVC will immediately stop tracking frame-time.

Offset	Field	Size	Value	Description
0	time_sync0	8	Number	Authoritative frame-time at TIME_SYNC0
8	time_sync1	8	Number	Authoritative frame-time at TIME_SYNC1
16	time_sync2	8	Number	Authoritative frame-time at TIME_SYNC2
24	time_sync3	8	Number	Authoritative frame-time at TIME_SYNC3

Table 9.55: SVC Protocol TimeSync Enable Response

Greybus SVC TimeSync Disable Request

The Greybus SVC Protocol TimeSync Disable request contains no payload.

Greybus SVC TimeSync Disable Response

The Greybus SVC Protocol TimeSync Disable response contains no payload. The SVC shall always return GB_OP_SUCCESS to this Operation. This Greybus Operation does not affect any Interface sub-states.

9.2.19 Greybus SVC TimeSync Authoritative Operation

The AP Module uses this operation to request the SVC to send the authoritative frame-time at each TIME_SYNC strobe. The SVC will return the authoritative frame-time at each TIME_SYNC in the response phase of this operation. Unused entires in the response frame shall be initialized to zero.

Greybus SVC TimeSync Authoritative Request

The Greybus SVC Protocol TimeSync Authoritative Request contains no payload.

Greybus SVC TimeSync Authoritative Response

Table 9.55 defines the Greybus SVC TimeSync Authoritative Response payload. The response specifies the authoritative frame-time at each TIME_SYNC strobe. Unused slots in the response shall contain zero. If the Response message header status field *Greybus Operation Status* is not equal to GB_OP_SUCCESS the values in the Operation Response payload are undefined and shall be ignored. This Greybus Operation does not affect any Interface sub-states.

9.2.20 Greybus SVC TimeSync Wake Pins Acquire Operation

The AP Module uses this operation to request the SVC to take ownership-of and to establish an initial state on a set of wake lines associated with the indicated bit-mask of Interface IDs specified by the strobe_mask parameter in the Request phase of the Operation.

The SVC will take control of the wake lines specified in the Request and set the outputs to logical 0.

Greybus SVC TimeSync Wake Pins Acquire Request

Table 9.56 defines the Greybus SVC TimeSync Wake Pins Acquire Request payload. The request supplies the bit-mask (strobe_mask) of Interface IDs which should have their wake pins set to output with logical state 0.

Offset	Field	Size	Value	Description
0	strobe_mask	4	Number	Bit-mask of Interface IDs SVC should allocate as outputs

Table 9.56: SVC Protocol TimeSync Wake Pins Acquire Request

Greybus SVC TimeSync Wake Pins Acquire Response

The Greybus SVC Protocol TimeSync Wake Pins Acquire Response contains no payload.

If the Response message header status field *Greybus Operation Status* is equal to GB_OP_SUCCESS then the SVC shall set the the *TimeSync Pulse* sub-state for the indicated set of Interfaces to WAKE_UNSET. After this Operation completes the *WAKE Pulse* shall be re-interpreted as a *TimeSync Pulse* subject to the restrictions defined in the hardware model.

If the Response message header status field *Greybus Operation Status* is not equal to GB_OP_SUCCESS the AP shall abandon further TimeSync activities.

9.2.21 Greybus SVC TimeSync Wake Pins Release Operation

The AP Module uses this operation to request the SVC to release ownership of any previously allocated wake pins. The SVC shall release all pins allocated for wake purposes in a previous successful Greybus SVC TimeSync Wake Pins Acquire operation.

Greybus SVC TimeSync Wake Pins Release Request

The Greybus SVC Protocol TimeSync Wake Pins Release request contains no payload.

Greybus SVC TimeSync Wake Pins Release Response

The Greybus SVC Protocol TimeSync Wake Pins Release Response contains no payload. The SVC shall always return GB_OP_SUCCESS to this Operation. Before completion of this Operation the the SVC shall set the *TimeSync Pulse* sub-state for the set of Interfaces previously indicated in the *Greybus SVC TimeSync Wake Pins Acquire Operation* to WAKE_UNSET. After this Operation completes the *TimeSync Pulse* shall be re-interpreted as a *WAKE Pulse* subject to the restrictions defined in the hardware-model.

9.2.22 Greybus SVC TimeSync Ping Operation

The AP Module uses this Operation to request the SVC to send a single TimeSync event on a bitmask of wake pins which must have previously been allocated via Greybus SVC TimeSync Wake Pins Acquire.

On receipt of this request the SVC will immediately generate a single pulse and capture the authoritative frame-time; this frame-time will then be returned in the response phase of the TimeSync Ping Operation.

Greybus SVC TimeSync Ping Request

The Greybus SVC Protocol TimeSync Ping Request contains no payload.

Offset	Field	Size	Value	Description
0	frame-time	8	Number	Authoritative frame-time at ping event

Table 9.57: SVC Protocol TimeSync Ping Response

Offset	Field	Size	Value	Description
0	primary_intf_id	1	Number	Module location

Table 9.58: SVC Protocol Module Eject Request

Greybus SVC TimeSync Ping Response

Table 9.57 defines the Greybus SVC TimeSync Ping Response payload. The response specifies the authoritative frame-time at the ping event generated. If the Response message header status field *Greybus Operation Status* is not equal to GB_OP_SUCCESS the values in the Operation Response payload are undefined and shall be ignored. This Greybus Operation does not affect any Interface sub-states.

9.2.23 Greybus SVC Module Eject Operation

The Greybus SVC Module Eject operation is sent by the AP Module to request the SVC to execute the necessary actions to eject a Module from the Frame.

Although the AP may send this Operation's request at any time following a successful *Greybus SVC Hello Operation*, the AP should ensure that the *Interface Lifecycle State* of each of the Interfaces in the attached Module is either *ATTACHED* or *OFF* before doing so. Otherwise, the effect on the Greybus System is equivalent to a *Forcible Removal (Any → DETACHED)* of the Module, and may otherwise disrupt the operation of the System.

Greybus SVC Module Eject Request

The Greybus SVC Module Eject Request is defined in Table 9.58. The primary_intf_id field in the request payload contains the Interface ID of the Primary Interface to the Module which the SVC shall eject from the Frame.

The SVC shall not perform any checking of the Interface State with ID given by the primary_intf_id field beyond ensuring it is a valid Interface ID.

After receiving the request, the SVC shall set the *RELEASE* sub-state for that Interface State to RELEASE_ASSERTED before sending a response back to the AP. The SVC may send the result before setting RELEASE back to RELEASE_DEASSERTED; that is, the RELEASE pulse may end after the AP has already received the response.

Greybus SVC Module Eject Response

The Greybus SVC Module Eject response message contains no payload.

As described in *RELEASE*, a RELEASE pulse is only an attempt to eject the Module. The Module may still be in the MODULE_ATTACHED state after the AP receives the result. Furthermore, the RELEASE pulse may fail to eject the Module.

If the release pulse is successful, the AP will receive a subsequent notification from the SVC in the form of a *Greybus SVC Module Removed Operation* request.

Offset	Field	Size	Value	Description
0	rail_count	1	Number	Number of power rails

Table 9.59: SVC Power Monitor Get Rail Count Response

9.2.24 Greybus SVC Ping Operation

The Greybus SVC Ping Operation is an exchange of Messages, neither of which contains payload data.

Greybus SVC Ping Request

The Greybus SVC Ping Request Message contains no payload.

Greybus SVC Ping Response

The Greybus SVC Ping Response Message contains no payload. The status byte in the Response *Message Header* shall be GB_OP_SUCCESS.

9.2.25 Greybus SVC Power Monitor Get Rail Count Operation

The Greybus SVC Power Monitor Get Rail Count operation retrieves the number of power rails for which power measurement is supported.

Greybus SVC Power Monitor Get Rail Count Request

The Greybus SVC Power Monitor Get Rail Count request is sent from the AP only. It has no payload.

Greybus SVC Power Monitor Get Rail Count Response

The Greybus SVC Power Monitor Get Rail Count response contains a 1-byte field 'rail_count'. The maximum supported number of rails is 254, 255 (0xff) is an invalid value. The rail count can equal 0 in which case no rail can be measured by the SVC.

9.2.26 Greybus SVC Power Monitor Get Rail Names Operation

The Greybus SVC Power Monitor Get Rail Names operation requests the names of all power rails for which power measurement is supported.

Greybus SVC Power Monitor Get Rail Names Request

The Greybus SVC Power Monitor Get Rail Names request is sent from the AP only. It has no payload.

Offset	Field	Size	Value	Description
0	status	1	Number	*Greybus SVC Protocol Operation Status*
1	rail_1_name	32	String	Rail #1 name
33	rail_2_name	32	String	Rail #2 name
(...)				

Table 9.60: SVC Power Monitor Get Rail Names Response

Offset	Field	Size	Value	Description
0	rail_id	1	Number	ID of the rail that shall be measured
1	type	1	Number	Measurement type indicator (*Greybus SVC Power Monitor Get Sample Type Indicators*)

Table 9.61: SVC Power Monitor Get Sample Request

Greybus SVC Power Monitor Get Rail Names Response

Table 9.60 defines the Greybus SVC Power Monitor Get Rail Names Response payload. If the Response message header has the *Greybus Operation Status* not equal to GB_OP_SUCCESS, the values in the Operation Response payload are undefined and shall be ignored.

Otherwise, If the status field in the Operation Response payload is not GB_SVC_OP_SUCCESS, values in all other fields of the Operation Response payload are undefined and shall be ignored.

The Greybus SVC Power Monitor Get Rail Names Response payload is comprised of human-readable names for rails that support voltage, current and power measurement. Each name consists of a fixed 32-byte sub-buffer containing a rail name padded with zero bytes. A rail name is comprised of a subset of *[US-ASCII]* characters: lower- and upper-case alphanumerics and the character '_'. A rail name is 1-32 bytes long; a 32-byte name has no pad bytes.

The number of these buffers shall be exactly the number returned by a prior Greybus SVC Power Monitor Get Rail Name Count operation.

If there are no measurable power rails on the platform, the status field in the Operation Response payload shall be set to GB_SVC_PWRMON_OP_NOT_PRESENT.

Each rail has an implicit 'Rail ID' which is equal to its position in the array of rail names returned by this response. The rail whose name is first in the array shall have Rail ID 0, the second shall have Rail ID 1, and so on. Despite using numeric IDs, the rail names returned by this operation are guaranteed to be unique.

9.2.27 Greybus SVC Power Monitor Get Sample Operation

The Greybus SVC Power Monitor Get Sample operation shall be used by the AP to retrieve a single measurement.

Greybus SVC Power Monitor Get Sample Request

The Greybus SVC Power Monitor Get Sample request is sent from the AP only. It contains the ID of the rail and the measurement type (current, voltage, power).

Greybus SVC Power Monitor Get Sample Type Indicators

Measurement type	Brief Description	Value
GB_SVC_PWRMON_TYPE_INVALID	Invalid request value	0x00
GB_SVC_PWRMON_TYPE_CURR	Current measurement in microamps (uA)	0x01
GB_SVC_PWRMON_TYPE_VOL	Voltage measurement in microvolts (uV)	0x02
GB_SVC_PWRMON_TYPE_PWR	Power measurement in microwatts (uW)	0x03
	(all other values reserved)	0x04..0xFF

Table 9.62: SVC Power Monitor measurement types

Offset	Field	Size	Value	Description
0	result	1	Number	Result code (*Greybus SVC Power Monitor Get Sample Result Codes*)
1	measurement	4	Number	Measured value

Table 9.63: SVC Power Monitor Get Sample Response

Greybus SVC Power Monitor Get Sample Response

The Greybus SVC Power Monitor Get Sample response contains a 1-byte result code and the measured value in a 4-byte unsigned integer. Units in which the retrieved values are represented are as follows: microvolts for voltage, microamps for current and microwatts for power.

Greybus SVC Power Monitor Get Sample Result Codes

9.2.28 Greybus SVC Power Monitor Interface Get Sample Operation

The Greybus SVC Power Monitor Interface Get Sample operation shall be used by the AP to retrieve a single measurement for the given interface.

Unlike the Greybus SVC Power Monitor Get Sample operation it does not require any preceding data exchange nor any prior knowledge about the power rails layout. It retrieves a single power supply measurement of the interface.

Greybus SVC Power Monitor Interface Get Sample Request

The Greybus SVC Power Monitor Interface Get Sample Request can only be sent from the AP. It contains a 1-byte interface ID and 1-byte measurement type (voltage, current, power).

Result code	Brief Description	Value
GB_SVC_PWRMON_GET_SAMPLE_OK	Measurement OK	0x00
GB_SVC_PWRMON_GET_SAMPLE_INVAL	Invalid ID provided in request	0x01
GB_SVC_PWRMON_GET_SAMPLE_NOSUPP	Measurement not supported for this ID	0x02
GB_SVC_PWRMON_GET_SAMPLE_HWERR	Internal hardware error	0x03
	(all other values reserved)	0x04..0xFF

Table 9.64: SVC Power Monitor Get Sample result codes

Offset	Field	Size	Value	Description
0	intf_id	1	Number	ID of the interface
1	type	1	Number	Measurement type indicator (*Greybus SVC Power Monitor Get Sample Type Indicators*)

Table 9.65: SVC Power Monitor Interface Get Sample Request

Offset	Field	Size	Value	Description
0	result	1	Number	Result code (*Greybus SVC Power Monitor Get Sample Result Codes*)
1	measurement	4	Number	Measured value

Table 9.66: SVC Power Monitor Interface Get Sample Response

Greybus SVC Power Monitor Interface Get Sample Response

The Greybus SVC Power Monitor Interface Get Sample response contains a 1-byte operation result code and the measured value in a 4-byte unsigned integer. Units in which the retrieved values are represented are as follows: microvolts for voltage, microamps for current and microwatts for power.

9.2.29 Greybus SVC Power Down Operation

The Greybus SVC Power Down operation shall be used by the AP to request the SVC to forcibly power down all the devices under its control and then put itself in power down mode. Prior to issuing such operation, the AP shall close all Greybus communication with all interfaces and then power all interfaces down.

When the SVC Power Down operation completes, the Greybus subsystem is no more operational: hotplug detection is unavailable, no Greybus communication with any interface is possible, and SVC is unable to process any new Greybus operation or event.

The SVC shall be reset to recover from this state.

Greybus SVC Power Down Request

The Greybus SVC Power Down request message contains no payload.

Greybus SVC Power Down Response

The Greybus SVC Power Down response message contains no payload.

9.2.30 Greybus SVC Module Inserted Operation

The Greybus SVC Module Inserted request is sent by the SVC to the AP Module to indicate that a new Module has been inserted into the Frame, as well as during initialization of a Greybus System, to inform the AP of Modules which were already attached to the Frame.

Greybus SVC Module Inserted Request

Table 9.67 defines the Greybus SVC Module Inserted request payload. The request specifies the location of the *Primary Interface* to the newly inserted Module in the primary_intf_id field. It also specifies the number of Interfaces covered by the Module in the intf_count field; this includes the Primary Interface, plus the total

Offset	Field	Size	Value	Description
0	primary_intf_id	1	Number	Module location
1	intf_count	1	Number	Number of Interfaces covered by Module
2	flags	2	Number	See Table 9.68

Table 9.67: SVC Protocol Module Inserted Request

Flag	Value	Description
NO_PRIMARY_INTERFACE	0x1	No Primary Interface to Module detected

Table 9.68: Flags for SVC Module Inserted Request

number of *Secondary Interface*s to the Module, if any. The size of a Module (the value of the intf_count field in the Module Inserted request payload) is thus always one or more.

The flags field in the request payload is a bit mask which allows the SVC to notify the AP of additional conditions associated with the insertion event. The mask values for the flags field are defined in Table 9.68.

The NO_PRIMARY_INTERFACE mask for the flags field allows the SVC to notify the AP when an error has occurred, and no Primary Interface to the Module was detected.

If the NO_PRIMARY_INTERFACE flag is set in the Module Inserted Request, the intf_count field shall equal one. The Interface State with Interface ID primary_intf_id shall have *ORDER* equal to ORDER_SECONDARY.

If the NO_PRIMARY_INTERFACE flag is not set in the Module Inserted request, then the Interface State with Interface ID primary_intf_id has *ORDER* equal to ORDER_PRIMARY. If intf_count is greater than one, all Interface States with IDs from (primary_intf_id + 1) through (primary_intf_id + intf_count - 1), inclusive, have ORDER equal to ORDER_SECONDARY.

In all cases, regardless of the value of the flags field, every Interface identified by the request is in the *ATTACHED Lifecycle State*. After sending the response to this request, the AP may thus subsequently attempt to *enumerate* these Interfaces.

Additionally, the entire Module has transitioned to the MODULE_ATTACHED state, as described in *Module Attach*.

The consequences of boot and enumeration when the NO_PRIMARY_INTERFACE flag is set are unspecified.

During the initialization of a Greybus System, following a successful *Greybus SVC Hello Operation*, the SVC shall attempt to exchange Module Inserted Operations with the AP for each attached Module.

Unless an error occurs, there is a unique Primary Interface to each Module attached to the Frame. The number of Operations exchanged during initialization is thus at least the number of *Interface States* that are *ATTACHED* and whose *ORDER* is ORDER_PRIMARY. The primary_intf_id fields in these requests shall be the Interface IDs of the Interface States whose ORDER is ORDER_PRIMARY.

There may be additional Secondary Interfaces to each of these Modules. The intf_count field in each such request shall thus equal one plus the number of consecutive Interface States in the Greybus System whose ORDER is ORDER_SECONDARY, starting from the Primary Interfaces to each attached Module, up to the final Interface Block in the *Slot*. This follows from the definitions of the ORDER sub-state and the intf_count request field.

The SVC may also send additional Module Inserted Requests with the NO_PRIMARY_INTERFACE flag set, as described above.

Offset	Field	Size	Value	Description
0	primary_intf_id	1	Number	Module location

Table 9.69: SVC Protocol Module Removed Request

Greybus SVC Module Inserted Response

The Greybus SVC Module Inserted response message contains no payload.

9.2.31 Greybus SVC Module Removed Operation

The Greybus SVC Module Removed request is sent by the SVC to the AP Module. It supplies the Interface ID for the Primary Interface to the Module that is no longer present. The Interface ID shall have been the subject of a previous *Greybus SVC Module Inserted Operation*.

Greybus SVC Module Removed Request

Table 9.69 defines the Greybus SVC Module Removed request payload. The request specifies the Primary Interface ID for the Module that is no longer present.

Using the most recent Module Inserted Operation on the SVC protocol whose primary_intf_id field equaled the primary_intf_id field in this request, the SVC notified the AP that one or more *Interfaces* were *ATTACHED*.

The current Lifecycle States of each of these Interfaces can be determined as follows.

- If the Interface was *ATTACHED* or *OFF*, then the Interface is now *DETACHED*.
- Otherwise, a forcible removal has occurred, as described in the *DETECT* section. When this occurs, the Interface's Lifecycle State is unpredictable.

Following a forcible removal, the AP and SVC shall proceed as described in *Forcible Removal (Any → DETACHED)*.

The Module is now in the MODULE_DETACHED state, as described in *Module Detach*.

Greybus SVC Module Removed Response

The Greybus SVC Module Removed response message contains no payload.

9.2.32 Greybus SVC Interface V_SYS Enable Operation

The AP uses this Operation to request the SVC to set an *Interface State's V_SYS* to V_SYS_ON.

Though the AP may send this request at any time, the AP should only do so as part of the "boot" and "reboot" transitions in the *Interface Lifecycle* state machine, as described in *Boot (ATTACHED → ACTIVATED)* and *Reboot (OFF → ACTIVATED)*.

The SVC shall not set V_SYS to V_SYS_ON except as a result of receiving a Greybus V_SYS Enable Request.

Offset	Field	Size	Value	Description
0	intf_id	1	Interface ID	Interface ID

Table 9.70: SVC Protocol Interface V_SYS Enable Request

Offset	Field	Size	Value	Description
0	result_code	1	Number	Result Code

Table 9.71: SVC Protocol Interface V_SYS Enable Response

Greybus SVC Interface V_SYS Enable Request

Table 9.70 defines the Greybus SVC Interface V_SYS Enable Request payload.

The SVC, on receiving this request, shall attempt to set the V_SYS sub-state of the Interface State specified by the intf_id field to V_SYS_ON.

Greybus SVC Interface V_SYS Enable Response

Table 9.71 defines the Greybus SVC Interface V_SYS Enable Response payload. The Operation Response payload contains a one-byte result_code field.

The *Greybus Operation Status* in the Operation Response message header shall not be used to determine the value of V_SYS sub-state after the response is received. It shall only be used to indicate the result of the Greybus communication. If the Greybus SVC Interface V_SYS Enable Response message header has the *Greybus Operation Status* value different than GB_OP_SUCCESS, a Greybus communication error has occurred; the V_SYS sub-state identified in the Operation Request shall not have changed as a result of processing the Request. If the Greybus SVC Interface V_SYS Enable Response message header has the *Greybus Operation Status* equal to GB_OP_SUCCESS, it shall indicate that no Greybus communication error was detected.

However, a *Greybus Operation Status* in the Response message header equal to GB_OP_SUCCESS alone does not imply the intended V_SYS is now V_SYS_ON. When the Response message header has the *Greybus Operation Status* equal to GB_OP_SUCCESS, the value of V_SYS may be determined given the result_code field in the Operation Response payload, as described in Table 9.72. In particular, V_SYS is V_SYS_ON if the Response message header has *Greybus Operation Status* equal to GB_OP_SUCCESS and the result_code in the Operation Response payload is V_SYS_OK. V_SYS shall not have changed value as a result of processing the Request in any other combination of these two fields.

9.2.33 Greybus SVC Interface V_SYS Disable Operation

The AP uses this Operation to request the SVC to set an *Interface State's V_SYS* to V_SYS_OFF.

Result Code	Value	Description
V_SYS_OK	0	V_SYS enable/disable operation was successful.
(Reserved)	1	(Reserved for future use)
V_SYS_FAIL	2	V_SYS enable/disable was attempted and failed.
(Reserved)	3-255	(Reserved for future use)

Table 9.72: Interface V_SYS Enable and Interface V_SYS Disable result_code

Offset	Field	Size	Value	Description
0	intf_id	1	Interface ID	Interface ID

Table 9.73: SVC Protocol Interface V_SYS Disable Request

Offset	Field	Size	Value	Description
0	result_code	1	Number	Result Code

Table 9.74: SVC Protocol Interface V_SYS Disable Response

Though the AP may send this request at any time, the AP should only do so under one of the following conditions:

- during the "power_down" and "early_power_down" transitions in the *Interface Lifecycle* state machine, as described in *Power Down (ENUMERATED → OFF)* and *Early Power Down (ACTIVATED → OFF)*.
- during the "forcible_removal" transition in the Interface Lifecycle state machine, as described in *Forcible Removal (Any → DETACHED)*.

The SVC shall set V_SYS to V_SYS_OFF without having received an Interface V_SYS Disable Request only under the conditions specified in *V_SYS*.

Greybus SVC Interface V_SYS Disable Request

Table 9.73 defines the Greybus SVC Interface V_SYS Disable Request payload.

The SVC, on receiving this request, shall attempt to set the V_SYS sub-state of the Interface State specified by the intf_id field to V_SYS_OFF.

Greybus SVC Interface V_SYS Disable Response

Table 9.74 defines the Greybus SVC Interface V_SYS Disable Response payload. The Operation Response payload contains a one-byte result_code field.

The meaning of the *Greybus Operation Status* in the Operation Response message header and the result_code in the Operation Response payload are analogous to the corresponding *Greybus Operation Status* in the Interface V_SYS Enable Response message header and the result_code field in the Interface V_SYS Enable Operation Response payload.

That is, the *Greybus Operation Status* of the Operation Response message header shall only be used to indicate the result of the Greybus communication, exactly as described in *Greybus SVC Interface V_SYS Enable Response*.

Similarly, when the Interface V_SYS Disable Response message header has the *Greybus Operation Status* equal to GB_OP_SUCCESS, the value of V_SYS may be determined given the result_code field in the Operation Response payload, as described in Table 9.72. In particular, V_SYS is V_SYS_OFF if Response message header has the *Greybus Operation Status* equal to GB_OP_SUCCESS and the result_code field in the Operation Response payload is V_SYS_OK. V_SYS shall not have changed value as a result of processing the Request in any other combination of these two fields.

Offset	Field	Size	Value	Description
0	intf_id	1	Interface ID	Interface ID

Table 9.75: SVC Protocol Interface REFCLK Enable Request

Offset	Field	Size	Value	Description
0	result_code	1	Number	Result Code

Table 9.76: SVC Protocol Interface REFCLK Enable Response

9.2.34 Greybus SVC Interface REFCLK Enable Operation

The AP uses this Operation to request the SVC to set an *Interface State's REFCLK* to REFCLK_ON.

Though the AP may send this request at any time, the AP should only do so under one of the following conditions:

- during the "boot" and "reboot" transitions in the *Interface Lifecycle* state machine, as described in *Boot (ATTACHED → ACTIVATED)* and *Reboot (OFF → ACTIVATED)*.
- while the Interface is *ENUMERATED*, if REFCLK is REFCLK_OFF and the AP has determined using application-specific means that REFCLK should be set to REFCLK_ON.
- if the Interface is *ENUMERATED* and REFCLK is REFCLK_OFF, during the "ms_enter" transition in the Interface Lifecycle state machine, as described in *Mode Switch Enter (ENUMERATED → MODE_SWITCHING)*.
- during the "resume" transition in the Interface Lifecycle state machine, as described in *Resume (SUSPENDED → ENUMERATED)*.

The SVC shall not set REFCLK to REFCLK_ON except as a result of receiving a Greybus REFCLK Enable Request.

Greybus SVC Interface REFCLK Enable Request

Table 9.75 defines the Greybus SVC Interface REFCLK Enable Request payload.

The SVC, on receiving this request, shall attempt to set the REFCLK sub-state of the Interface State specified by the intf_id field to REFCLK_ON.

Greybus SVC Interface REFCLK Enable Response

Table 9.76 defines the Greybus SVC Interface REFCLK Enable Response payload. The Operation Response payload contains a one-byte result_code field.

The *Greybus Operation Status* in the Operation Response message header shall not be used to determine the value of REFCLK sub-state after the response is received. It shall only be used to indicate the result of the Greybus communication. If the Greybus SVC Interface REFCLK Enable Response message header has the *Greybus Operation Status* value different than GB_OP_SUCCESS, a Greybus communication error has occurred; the REFCLK sub-state identified in the Operation Request shall not have changed as a result of processing the request. If the Greybus SVC Interface REFCLK Enable Response message header has the *Greybus Operation Status* equal to GB_OP_SUCCESS, it shall indicate that no Greybus communication error was detected.

Result Code	Value	Description
REFCLK_OK	0	REFCLK enable/disable operation was successful.
(Reserved)	1	(Reserved for future use)
REFCLK_FAIL	2	REFCLK enable/disable was attempted and failed.
(Reserved)	3-255	(Reserved for future use)

Table 9.77: Interface REFCLK Enable and Interface REFCLK Disable result_code

Offset	Field	Size	Value	Description
0	intf_id	1	Interface ID	Interface ID

Table 9.78: SVC Protocol Interface REFCLK Disable Request

However, a *Greybus Operation Status* in the Response message header equal to GB_OP_SUCCESS alone does not imply the intended REFCLK is now REFCLK_ON. When the Response message header has the *Greybus Operation Status* equal to GB_OP_SUCCESS, the value of REFCLK may be determined given the result_code field in the Operation Response payload, as described in Table 9.77. In particular, REFCLK is REFCLK_ON if the Response message header has the *Greybus Operation Status* equal to GB_OP_SUCCESS and the result_code in the Operation Response payload is REFCLK_OK. REFCLK shall not have changed value as a result of processing the request in any other combination of these two fields.

9.2.35 Greybus SVC Interface REFCLK Disable Operation

The AP uses this Operation to request the SVC to set an *Interface State's REFCLK* to REFCLK_OFF.

Though the AP may send this request at any time, the AP should only do so under one of the following conditions:

- during "early_power_down" transition in the *Interface Lifecycle* state machine, as described in *Early Power Down (ACTIVATED → OFF)*.
- while the Interface is *ENUMERATED*, if REFCLK is REFCLK_ON and the AP has determined using application-specific means that REFCLK should be set to REFCLK_OFF.
- if the Interface is *ENUMERATED* and REFCLK is REFCLK_ON, during the "power_down" transition in the Interface Lifecycle state machine, as described in *Power Down (ENUMERATED → OFF)*.
- if the Interface is *ENUMERATED* and REFCLK is REFCLK_ON, during the "suspend" transition in the Interface Lifecycle state machine, as described in *Suspend (ENUMERATED → SUSPENDED)*.
- during the "forcible_removal" transition in the Interface Lifecycle state machine, as described in *Forcible Removal (Any → DETACHED)*.

The SVC shall set REFCLK to REFCLK_OFF without having received an Interface REFCLK Disable Request only under the conditions specified in *REFCLK*.

Greybus SVC Interface REFCLK Disable Request

Table 9.78 defines the Greybus SVC Interface REFCLK Disable Request payload.

The SVC, on receiving this request, shall attempt to set the REFCLK sub-state of the Interface State specified by the intf_id field to REFCLK_OFF.

Offset	Field	Size	Value	Description
0	result_code	1	Number	Result Code

Table 9.79: SVC Protocol Interface REFCLK Disable Response

Greybus SVC Interface REFCLK Disable Response

Table 9.79 defines the Greybus SVC Interface REFCLK Disable Response payload. The Operation Response payload contains a one-byte result_code field.

The meaning of the *Greybus Operation Status* in the Operation Response message header and the result_code in the Operation Response payload are analogous to the corresponding *Greybus Operation Status* in the Interface REFCLK Enable Response message header and the result_code field in the Interface REFCLK Enable Operation Response payload.

That is, the *Greybus Operation Status* of the Operation Response message header shall only be used to indicate the result of the Greybus communication, exactly as described in *Greybus SVC Interface REFCLK Enable Response*.

Similarly, when the Interface REFCLK Disable Response message header has the *Greybus Operation Status* equal to GB_OP_SUCCESS, the value of REFCLK may be determined given the result_code field in the Operation Response payload, as described in Table 9.77. In particular, REFCLK is REFCLK_OFF if the Response message header has *Greybus Operation Status* equal to GB_OP_SUCCESS and the result_code field in the Operation Response payload is REFCLK_OK. REFCLK shall not have changed value as a result of processing the Request in any other combination of these two fields.

9.2.36 Greybus SVC Interface UNIPRO Enable Operation

The AP uses this Operation to request the SVC to set an *Interface State's UNIPRO* to UPRO_DOWN.

Note: Important:

1. This operation will *not* result in UNIPRO being UPRO_UP. The UniPro state machine requires communication between peers before entering the state modeled by the UPRO_UP value of a Greybus Interface State's UNIPRO sub-state. The process by which UNIPRO transitions from UPRO_DOWN to UPRO_UP is described in *Boot (ATTACHED → ACTIVATED)*.
2. There are additional UNIPRO sub-state values which similarly are not reachable using this operation alone.

Though the AP may send this request at any time, the AP should only do so during the "boot" and "reboot" transitions in the *Interface Lifecycle* state machine, as described in *Boot (ATTACHED → ACTIVATED)* and *Reboot (OFF → ACTIVATED)*.

The SVC shall not set UNIPRO to UPRO_DOWN except as a result of receiving a Greybus UNIPRO Enable Request.

Greybus SVC Interface UNIPRO Enable Request

Table 9.80 defines the Greybus SVC Interface UNIPRO Enable Request payload.

The SVC, on receiving this request, shall check the UNIPRO sub-state of the Interface State with Interface ID intf_id. If UNIPRO is not UPRO_OFF, the SVC shall not attemp to change the UNIPRO sub-state value.

Offset	Field	Size	Value	Description
0	intf_id	1	Interface ID	Interface ID

Table 9.80: SVC Protocol Interface UNIPRO Enable Request

Offset	Field	Size	Value	Description
0	result_code	1	Number	Result Code

Table 9.81: SVC Protocol Interface UNIPRO Enable Response

The SVC shall signal an error to the AP in the response as described below, and shall take no further action related to this request.

If UNIPRO is UPRO_OFF, the SVC shall attempt to set the UNIPRO sub-state of the Interface State specified by the intf_id field to UPRO_DOWN.

Greybus SVC Interface UNIPRO Enable Response

Table 9.81 defines the Greybus SVC Interface UNIPRO Enable Response payload. The Operation Response payload contains a one-byte result_code field.

The *Greybus Operation Status* in the Operation Response message header shall not be used to determine the value of UNIPRO sub-state after the response is received. It shall only be used to indicate the result of the Greybus communication. If the Greybus SVC Interface UNIPRO Enable Response message header has the *Greybus Operation Status* value different than GB_OP_SUCCESS, a Greybus communication error has occurred; the UNIPRO sub-state identified in the Operation Request shall not have changed as a result of processing the Request. If the Greybus SVC Interface UNIPRO Enable Response message header has the *Greybus Operation Status* equal to GB_OP_SUCCESS, it shall indicate that no Greybus communication error was detected.

However, a *Greybus Operation Status* in the Response message header equal to GB_OP_SUCCESS alone does not imply the intended UNIPRO is now UNIPRO_DOWN. When the Response message header has *Greybus Operation Status* equal to GB_OP_SUCCESS, the value of UNIPRO may be determined given the result_code field in the Operation Response payload, as described in Table 9.82. In particular, if the Response message header has *Greybus Operation Status* equal to GB_OP_SUCCESS:

- UNIPRO is UPRO_DOWN if the result_code is UPRO_OK.
- UNIPRO shall not have changed value if the result_code is UPRO_NOT_OFF.
- UNIPRO is unpredictable if the result_code is UPRO_FAIL.

Result Code	Value	Description
UPRO_OK	0	UNIPRO enable/disable operation was successful.
(Reserved)	1	(Reserved for future use)
UPRO_FAIL	2	UNIPRO enable/disable was attempted and failed.
UPRO_NOT_OFF	3	UNIPRO was not UPRO_OFF, attempt to set to UPRO_DOWN was not made.
(Reserved)	4-255	(Reserved for future use)

Table 9.82: Interface UNIPRO Enable and Interface UNIPRO Disable result_code

Offset	Field	Size	Value	Description
0	intf_id	1	Interface ID	Interface ID

Table 9.83: SVC Protocol Interface UNIPRO Disable Request

Offset	Field	Size	Value	Description
0	result_code	1	Number	Result Code

Table 9.84: SVC Protocol Interface UNIPRO Disable Response

9.2.37 Greybus SVC Interface UNIPRO Disable Operation

The AP uses this Operation to request the SVC to set an *Interface State's UNIPRO* to UPRO_OFF.

Though the AP may send this request at any time, the AP should only do so under one of the following conditions:

- during "early_power_down" transition in the *Interface Lifecycle* state machine, as described in *Early Power Down (ACTIVATED → OFF)*.
- during the "power_down" transition in the Interface Lifecycle state machine, as described in *Power Down (ENUMERATED → OFF)*.
- during the "forcible_removal" transition in the Interface Lifecycle state machine, as described in *Forcible Removal (Any → DETACHED)*.

The SVC shall set UNIPRO to UPRO_OFF without having received an Interface UNIPRO Disable Request only under the conditions specified in *UNIPRO*.

Greybus SVC Interface UNIPRO Disable Request

Table 9.83 defines the Greybus SVC Interface UNIPRO Disable Request payload.

The SVC, on receiving this request, shall attempt to set the UNIPRO sub-state of the Interface State specified by the intf_id field to UPRO_OFF.

Greybus SVC Interface UNIPRO Disable Response

Table 9.84 defines the Greybus SVC Interface UNIPRO Disable Response payload. The Operation Response payload contains a one-byte result_code field.

The meaning of the *Greybus Operation Status* in the Operation Response message header and the result_code in the Operation Response payload are analogous to the corresponding *Greybus Operation Status* in the Interface UNIPRO Enable Response message header and the result_code field in the Interface UNIPRO Enable Operation Response payload.

That is, the *Greybus Operation Status* of the Operation Response message header shall only be used to indicate the result of the Greybus communication, exactly as described in *Greybus SVC Interface UNIPRO Enable Response*.

Similarly, when the Interface UNIPRO Disable Response message header has the *Greybus Operation Status* equal to GB_OP_SUCCESS, the value of UNIPRO may be determined given the result_code field in the Operation Response payload, as described in Table 9.82. In particular, if the Response message header has the *Greybus Operation Status* equal to GB_OP_SUCCESS:

- UNIPRO is UPRO_OFF if the result_code is UPRO_OK.

Offset	Field	Size	Value	Description
0	intf_id	1	Interface ID	Interface to activate

Table 9.85: SVC Protocol Interface Activate Request

- UNIPRO is unpredictable if the result_code is UPRO_FAIL.

9.2.38 Greybus SVC Interface Activate Operation

The Greybus SVC Interface Activate Operation allows the AP to request the SVC to "activate" an Interface by initializing it and determining if it is capable of communication via Greybus.

More precisely, use of this Operation is the final step in a sequence of Greybus Operations which are used when transitioning an *Interface* to the *ACTIVATED* Interface *Lifecycle State*, as defined in *The Interface Lifecycle*.

Though the AP may send this request at any time, the AP should only do so during the "boot" and "reboot" transitions in the Interface Lifecycle state machine as defined in *Boot (ATTACHED → ACTIVATED)* and *Reboot (OFF → ACTIVATED)*. The effect of sending this request under other conditions is unspecified.

The SVC shall not send this Operation request.

Greybus SVC Interface Activate Request

Table 9.85 defines the Greybus SVC Interface Activate Request payload.

Upon receiving this request, the SVC shall check the following sub-states of the *Interface State* with ID intf_id have these values:

- *DETECT* is DETECT_ACTIVE
- *V_SYS* is V_SYS_ON
- *V_CHG* is V_CHG_OFF
- *WAKE* is WAKE_UNSET
- *UNIPRO* is UPRO_DOWN
- *REFCLK* is REFCLK_ON
- *RELEASE* is RELEASE_DEASSERTED
- *ORDER* is ORDER_PRIMARY or ORDER_SECONDARY
- *MAILBOX* is MAILBOX_NONE

If any of these conditions does not hold, the SVC shall send a response to the AP signaling an error as described below. The SVC shall take no further action related to such a request beyond sending the response.

Otherwise, the only Interface sub-state whose value is not constrained is *INTF_TYPE*.

The SVC and Module shall now activate the Interface by following these steps in the order specified.

If this sequence completes successfully, INTF_TYPE is one of IFT_DUMMY, IFT_UNIPRO, or IFT_GREYBUS, and the Interface's *Lifecycle State* is consequently *ACTIVATED*. If this sequence fails, INTF_TYPE is IFT_UNKNOWN, and the SVC shall signal an error to the AP in the response, as described below.

This sequence is also depicted in *Boot (ATTACHED → ACTIVATED)* and *Reboot (OFF → ACTIVATED)*.

1. If the SVC is notified that UNIPRO is UPRO_LSS at any time, immediately proceed to step 6.
2. The SVC shall initiate a *WAKE pulse* for a duration greater than or equal to the WAKE Pulse Cold Boot Threshold.
3. After the WAKE Pulse completes, the SVC shall start a timer, for an implementation-defined duration.

   If the SVC detects the timer has expired and UNIPRO is UPRO_DOWN, the activation sequence is complete. The SVC shall set *INTF_TYPE* to IFT_DUMMY. The Interface is ACTIVATED, as described above. When this occurs, immediately proceed to step 9.
4. Since DETECT is DETECT_ACTIVE, a Module is attached to the Interface Block. If the attached Module's Interface is capable of communication via UniPro, it shall detect when the WAKE Pulse duration equals the WAKE Pulse Cold Boot Threshold, and perform an internal reset sequence to its initial state.

   Note that the Interface may draw power from the Frame, and make use of the reference clock supplied by the Frame, during this initialization, since V_SYS and REFCLK are respectively V_SYS_ON and REFCLK_ON.
5. If the Interface is capable of UniPro communications, it shall set UNIPRO to UPRO_LSS during its initialization sequence.

   As stated in *UNIPRO*, the SVC shall be notified if UNIPRO is set to UPRO_LSS, and if UNIPRO remains UPRO_LSS for too long, UNIPRO autonomously becomes UPRO_DOWN.

   Note that the Interface cannot set MAILBOX unless UNIPRO is UPRO_UP.
6. If the SVC receives the notification that UNIPRO is UPRO_LSS following any previous step, the SVC shall attempt to set UNIPRO to UPRO_UP, and start another timer, for another implementation defined duration.

   If the SVC detects this timer has expired and *MAILBOX* is MAILBOX_NONE, the activation sequence is complete. The SVC shall set INTF_TYPE to IFT_UNIPRO. The Interface is ACTIVATED, as described above. When this occurs, immediately proceed to step 9.
7. If the Interface is notified that UNIPRO is UPRO_UP and supports Greybus communications, it may set MAILBOX to MAILBOX_GREYBUS. The Interface shall not set MAILBOX to any other value.

   Before setting MAILBOX, the Interface shall ensure that the *Greybus Interface Attributes* are set to their correct values and are available for retrieval, if they are supported.

   If the Interface sets MAILBOX, it shall subsequently respond to incoming *Control Protocol* Operation Requests as defined in that section if the appropriate CPort is connected and used for Greybus communication.
8. As stated in *MAILBOX*, the SVC can detect if the MAILBOX value has changed, and if so, to what value.

   If this occurs and MAILBOX is MAILBOX_GREYBUS, the SVC shall set INTF_TYPE to IFT_GREYBUS. The SVC shall then attempt to clear the mailbox attribute by setting its value to zero, setting MAILBOX to MAILBOX_NONE as a result. If the SVC is unable to do so, the results are undefined. Immediately proceed to step 9.

   If this occurs and MAILBOX is not MAILBOX_GREYBUS, the SVC shall set INTF_TYPE to IFT_UNKNOWN, and signal an error to the AP as described below.
9. Regardless of the path to reach this step, the INTF_TYPE sub-state is now set. The activation sequence is complete.

Offset	Field	Size	Value	Description
0	status	1	Number	*Greybus SVC Protocol Operation Status*
1	intf_type	1	INTF_TYPE	*INTF_TYPE* of activated Interface

Table 9.86: SVC Protocol Interface Activate Response

If the Interface is ACTIVATED, the SVC shall now send a successful response. Otherwise, it shall signal an error in the response.

Greybus SVC Interface Activate Response

Table 9.86 defines the Greybus SVC Interface Activate Operation Response payload. If the Response message header has *Greybus Operation Status* not equal to GB_OP_SUCCESS, the values in the Operation Response payload are undefined and shall be ignored.

After receiving the request, the SVC first checks various sub-states before starting the activation sequence. If any of these checks fail, the SVC shall signal errors to the AP in the Operation Response payload by setting the status field of the Operation Response payload as follows.

- If DETECT was not DETECT_ACTIVE, the status is GB_SVC_INTF_NOT_DETECTED.
- Otherwise, if V_SYS was not V_SYS_ON, the status is GB_SVC_INTF_NO_V_SYS.
- Otherwise, if V_CHG was not V_CHG_OFF, the status is GB_SVC_INTF_V_CHG.
- Otherwise, if WAKE was not WAKE_UNSET, the status is GB_SVC_INTF_WAKE_BUSY.
- Otherwise, if UNIPRO was not UPRO_DOWN, the status is GB_SVC_INTF_UPRO_NOT_DOWN.
- Otherwise, if REFCLK was not REFCLK_ON, the status is GB_SVC_INTF_NO_REFCLK.
- Otherwise, if RELEASE was not RELEASE_DEASSERTED, the status is GB_SVC_INTF_RELEASING.
- Otherwise, if ORDER was ORDER_UNKNOWN, the status is GB_SVC_INTF_NO_ORDER.
- Otherwise, if MAILBOX was not MAILBOX_NONE, the status is GB_SVC_INTF_MBOX_SET.

Also as described above, INTF_TYPE may be IFT_UNKNOWN due to the Interface having set MAILBOX to an illegal value. If this occurred, the SVC shall signal an error to the AP by setting the status field in the Operation Response payload to GB_SVC_INTF_BAD_MBOX.

If the Interface is *ACTIVATED* and no other errors occur, the SVC shall set the *Greybus Operation Status* in the Response message header to GB_OP_SUCCESS and the status field of the Operation Response payload to GB_SVC_OP_SUCCESS. In this case, the intf_type field in the Operation Response payload contains the numeric value of the INTF_TYPE as defined in *INTF_TYPE*.

9.2.39 Greybus SVC Interface Resume Operation

The Greybus SVC Interface Resume Operation allows the AP to request the SVC to "resume" an Interface which was previously *SUSPENDED*, allowing it to later be *ENUMERATED*.

More precisely, use of this Operation is one step in a sequence of Greybus Operations which are used when transitioning an *Interface* to the *ENUMERATED* Interface *Lifecycle State* from the *SUSPENDED* Lifecycle State, as defined in *The Interface Lifecycle*.

Offset	Field	Size	Value	Description
0	intf_id	1	Interface ID	Interface to resume

Table 9.87: SVC Protocol Interface Resume Request

Though the AP may send this request at any time, the AP should only do so during the "resume" transition in the Interface Lifecycle state machine as defined in *Resume (SUSPENDED → ENUMERATED)*. The effect of sending this request under other conditions is unspecified.

The SVC shall not send this Operation request.

Greybus SVC Interface Resume Request

Table 9.87 defines the Greybus SVC Interface Resume Request payload.

Upon receiving this request, the SVC shall check the following sub-states of the *Interface State* with ID intf_id have these values:

- *DETECT* is DETECT_ACTIVE
- *V_SYS* is V_SYS_ON
- *WAKE* is WAKE_UNSET
- *UNIPRO* is UPRO_HIBERNATE
- *REFCLK* is REFCLK_ON
- *RELEASE* is RELEASE_DEASSERTED
- *INTF_TYPE* is IFT_GREYBUS
- *ORDER* is ORDER_PRIMARY or ORDER_SECONDARY
- *MAILBOX* is MAILBOX_NONE

If any of these conditions does not hold, the SVC shall send a response to the AP signaling an error as described below. The SVC shall take no further action related to such a request beyond sending the response.

The SVC and Module shall now resume the Interface by following these steps in the order specified.

This sequence is also depicted in *Resume (SUSPENDED → ENUMERATED)*.

1. If the SVC detects at any time that *MAILBOX* is MAILBOX_GREYBUS, the resume sequence has succeeded. Go directly to step 6.
2. The SVC shall initiate a *WAKE Pulse* for a duration less than the WAKE Pulse Cold Boot Threshold.

   After the WAKE Pulse, the SVC shall delay in this step for an implementation-defined duration to allow the Interface to prepare for the sequence to continue.
3. Since INTF_TYPE is IFT_GREYBUS, the Interface is capable of UniPro and Greybus communication.

   The Interface shall detect the WAKE Pulse, and that its duration was less than the Wake Pulse Cold Boot Threshold. As a result, it shall perform an implementation-specific resume sequence. This sequence shall ensure that the Interface receives a notification if the SVC attempts to set UNIPRO to UPRO_UP.

   Note that the Interface may draw power from the Frame, and make use of the reference clock supplied by the Frame, during this resume sequence, since V_SYS and REFCLK are respectively V_SYS_ON and REFCLK_ON.

4. As described in *UNIPRO*, the SVC can attempt to set UNIPRO to UPRO_UP, and shall be notified if the attempt succeeds or fails.

   The SVC shall now attempt to set UNIPRO to UPRO_UP, and delay until it is notified whether the attempt succeeds or fails.

   If the attempt succeeds, the SVC sets a timer for an implementation-defined duration. If the SVC detects this timer has expired and *MAILBOX* is MAILBOX_NONE, the resume sequence has failed. The SVC shall signal an error to the AP as described below. Go directly to step 6.

   If the attempt fails, the resume sequence has failed. The SVC shall signal an error to the AP as described below. Go directly to step 6.

5. As described above, the Interface shall also be notified that UNIPRO has successfully been set to UNIPRO_UP. When this occurs, the Interface shall set MAILBOX to MAILBOX_GREYBUS. The Interface shall not set MAILBOX to any other value.

   After setting MAILBOX, the Interface shall subsequently respond to incoming *Control Protocol* Operation Requests as defined in that section if the appropriate CPort is connected and used for Greybus communication.

   The SVC shall detect the new value of MAILBOX. The SVC shall then attempt to clear the mailbox attribute by setting its value to zero, setting MAILBOX to MAILBOX_NONE as a result. If the SVC is unable to do so, the results are undefined.

   Otherwise, the resume sequence has succeeded. The SVC shall signal this success to the AP in the response to this request as described below.

6. The resume sequence is now complete, and has succeeded or failed. The SVC shall signal completion and either success or failure to the AP as described below.

Greybus SVC Interface Resume Response

Table 9.88 defines the Greybus SVC Interface Resume Response payload. If the Response message header has *Greybus Operation Status* not equal to GB_OP_SUCCESS, the value in the Operation Response payload is undefined and shall be ignored.

After receiving the request, the SVC first checked various sub-states before starting the resume sequence. If any of these checks fail, the SVC shall signal errors to the AP in the Operation Response payload by setting the status field of the Operation Response payload as follows.

- If DETECT was not DETECT_ACTIVE, the status is GB_SVC_INTF_NOT_DETECTED.
- Otherwise, if V_SYS was not V_SYS_ON, the status is GB_SVC_INTF_NO_V_SYS.
- Otherwise, if WAKE was not WAKE_UNSET, the status is GB_SVC_INTF_WAKE_BUSY.
- Otherwise, if UNIPRO was not UPRO_HIBERNATE, the status is GB_SVC_INTF_UPRO_NOT_HIBERNATED.
- Otherwise, if REFCLK was not REFCLK_ON, the status is GB_SVC_INTF_NO_REFCLK.
- Otherwise, if RELEASE was not RELEASE_DEASSERTED, the status is GB_SVC_INTF_RELEASING.
- Otherwise, if ORDER was ORDER_UNKNOWN, the status is GB_SVC_INTF_NO_ORDER.
- Otherwise, if MAILBOX was not MAILBOX_NONE, the status is GB_SVC_INTF_MBOX_SET.

If a protocol error occurs due to erroneous Interface behavior which writes a different value than MAILBOX_GREYBUS to MAILBOX, the SVC shall set the status field in Operation Response payload to GB_SVC_INTF_BAD_MBOX.

Offset	Field	Size	Value	Description
0	status	1	Number	*Greybus SVC Protocol Operation Status*

Table 9.88: SVC Interface Resume Response

Offset	Field	Size	Value	Description
0	intf_id	1	Interface ID	Interface State whose MAILBOX was set
1	result_code	2	Number	UniPro ConfigResultCode
3	mailbox	4	Number	MAILBOX value

Table 9.89: SVC Protocol Interface Mailbox Event Request

If the resume sequence failed because the SVC detected in step 4 that MAILBOX was MAILBOX_NONE, the SVC shall set the status field in the Operation response payload to GB_SVC_INTF_OP_TIMEOUT.

If the resume sequence failed because the SVC was notified in step 4 that the attempt to set UPRO to UPRO_UP failed, the SVC shall set the status field in the Operation Response payload to GB_SVC_INTF_NO_UPRO_LINK.

If the resume sequence succeeded and no other errors occurred, the SVC shall set the *Greybus Operation Status* in the Response message header to GB_OP_SUCCESS and set the status field in Operation Response payload to GB_SVC_OP_SUCCESS.

9.2.40 Greybus SVC Interface Mailbox Event Operation

The Greybus SVC Interface Mailbox Event Operation allows the SVC to inform the AP that the *MAILBOX* of an *Interface State* has changed value.

Though this can occur at other times, it carries special meaning during the "ms_exit" transition from the *MODE_SWITCHING* Interface *Lifecycle State* to *ENUMERATED*, as defined in *The Interface Lifecycle*. This is described in *Mode Switch Exit (MODE_SWITCHING → ENUMERATED)*.

Though an Interface can set the MAILBOX sub-state at other times, it should only do so when explicitly required to do so by the Greybus Specification. If a MAILBOX changes value due to other circumstances, the SVC shall send this Operation Request subject to the restrictions described below, with results that are implementation-defined.

The AP shall not send this Operation Request.

Greybus SVC Interface Mailbox Event Request

Table 9.89 defines the Greybus SVC Interface Mailbox Event Request payload.

As described in *MAILBOX*, under certain circumstances, an Interface can set the MAILBOX sub-state for that Interface State. This event can be detected by the SVC, and the SVC can subsequently read the value written.

If the SVC detects such an attribute write, it shall attempt to send an SVC Interface Mailbox Event Request to the AP if none of the following conditions hold:

1. The SVC is currently activating that Interface, as described in *Greybus SVC Interface Activate Request*.
2. The SVC is currently resuming that Interface, as described in *Greybus SVC Interface Resume Request*.
3. The Interface is an *AP Interface*.

Offset	Field	Size	Value	Description
0	intf_id	1	Number	Interface to reset
1	reason	1	Number	*Greybus SVC Interface Oops Reason*

Table 9.90: SVC Protocol Interface Oops Request

If any of the above conditions hold, the SVC shall not attempt to send a Mailbox Event Request to the AP as a result of detecting that the attribute was written.

The SVC shall attempt to exchange a Mailbox Event Request Operation with the AP by sending this request under any other circumstances. If the Operation fails, the SVC may take no further action as a result of detecting the mailbox attribute write.

Before sending the request, the SVC shall:

1. Attempt to read the mailbox attribute, storing the ConfigResultCode as defined in the UniPro specification, as well the mailbox attribute value if the read is successful.
2. If the mailbox attribute is read successfully, the SVC shall clear it by setting its value to zero, thus setting the MAILBOX Interface State sub-state to MAILBOX_NONE.

   If the SVC is unable to successfully clear the attribute, the results are undefined.

The SVC shall then send the request. The intf_id field in the request payload shall equal the MAILBOX Interface State's Interface ID. The result_code field in the request payload shall equal the previously stored ConfigResultCode. The mailbox field in the request payload shall be zero if the read failed, and otherwise shall equal the mailbox attribute's value.

Greybus SVC Interface Mailbox Event Response

The Greybus SVC Interface Mailbox Event Response has no payload.

9.2.41 Greybus SVC Interface Oops Operation

The SVC sends this to the AP Module to notify it that an Interface has experienced a fatal error.

Greybus SVC Interface Oops Request

Table 9.90 defines the Greybus SVC Interface Oops Request payload.

The Greybus SVC Interface Oops Request shall be sent only by the SVC to the AP Module. The Interface ID informs the AP Module which Interface has experienced a fatal error. The reason field specifies the cause of the fatal error. Presently, the only cause is GB_OOPS_REASON_OVERCURRENT, which indicates the Interface has caused an overconsumption event on the V_SYS power bus.

Greybus SVC Interface Oops Reason

Table 9.91 defines the constants used to indicate the reason in an Interface Oops Request.

Greybus SVC Interface Oops Response

The Greybus SVC Interface Oops Response Message contains no payload.

reason	Brief Description	Value
GB_OOPS_REASON_INVALID	Invalid reason value	0x00
GB_OOPS_REASON_OVERCURRENT	Interface shut down due to V_SYS overconsumption	0x01
	(all other values reserved)	0x02..0xff

Table 9.91: SVC Oops reasons

Offset	Field	Size	Value	Description
0	intf_id	1	Interface ID	Interface ID

Table 9.92: SVC Protocol Interface V_CHG Enable Request

9.2.42 Greybus SVC Interface V_CHG Enable Operation

The AP uses this Operation to request the SVC to set an *V_CHG* sub-state of *Interface State's* to V_CHG_ON.

The SVC shall not set V_CHG to V_CHG_ON except as a result of receiving a Greybus V_CHG Enable Request.

Greybus SVC Interface V_CHG Enable Request

Table 9.92 defines the Greybus SVC Interface V_CHG Enable Request payload.

The SVC, on receiving this request, shall attempt to set the V_CHG sub-state of the Interface State specified by the intf_id field to V_CHG_ON.

Greybus SVC Interface V_CHG Enable Response

Table 9.93 defines the Greybus SVC Interface V_CHG Enable Response payload. The Operation Response payload contains a one-byte result_code field.

The *Greybus Operation Status* in the Operation Response message header shall not be used to determine the value of V_CHG sub-state after the response is received. It shall only be used to indicate the result of the Greybus communication. If the Greybus SVC Interface V_CHG Enable Response message header has the *Greybus Operation Status* value different than GB_OP_SUCCESS, a Greybus communication error has occurred; the V_CHG sub-state identified in the Operation Request shall not have changed as a result of processing the Request. If the Greybus SVC Interface V_CHG Enable Response message header has the *Greybus Operation Status* equal to GB_OP_SUCCESS, it shall indicate that no Greybus communication error was detected.

However, a *Greybus Operation Status* in the Response message header equal to GB_OP_SUCCESS alone does not imply the intended V_CHG is now V_CHG_ON. When the Response message header has the *Greybus Operation Status* equal to GB_OP_SUCCESS, the value of V_CHG may be determined given the result_code field in the Operation Response payload, as described in Table 9.94. In particular, V_CHG is V_CHG_ON if the Response message header has *Greybus Operation Status* equal to GB_OP_SUCCESS and the result_code

Offset	Field	Size	Value	Description
0	result_code	1	Number	Result Code

Table 9.93: SVC Protocol Interface V_CHG Enable Response

Result Code	Value	Description
V_CHG_OK	0	V_CHG enable/disable operation was successful.
V_CHG_FAIL	1	V_CHG enable/disable was attempted and failed.
(Reserved)	2-255	(Reserved for future use)

Table 9.94: Interface V_CHG Enable and Interface V_CHG Disable result_code

Offset	Field	Size	Value	Description
0	intf_id	1	Interface ID	Interface ID

Table 9.95: SVC Protocol Interface V_CHG Disable Request

in the Operation Response payload is V_CHG_OK. V_CHG shall not have changed value as a result of processing the Request in any other combination of these two fields.

9.2.43 Greybus SVC Interface V_CHG Disable Operation

The AP uses this Operation to request the SVC to set an *V_CHG* sub-state of *Interface State's* to V_CHG_OFF.

The SVC shall set V_CHG to V_CHG_OFF without having received an Interface V_CHG Disable Request only under the conditions specified in *V_CHG*.

Greybus SVC Interface V_CHG Disable Request

Table 9.95 defines the Greybus SVC Interface V_CHG Disable Request payload.

The SVC, on receiving this request, shall attempt to set the V_CHG sub-state of the Interface State specified by the intf_id field to V_CHG_OFF.

Greybus SVC Interface V_CHG Disable Response

Table 9.96 defines the Greybus SVC Interface V_CHG Disable Response payload. The Operation Response payload contains a one-byte result_code field.

The meaning of the *Greybus Operation Status* in the Operation Response message header and the result_code in the Operation Response payload are analogous to the corresponding *Greybus Operation Status* in the Interface V_CHG Enable Response message header and the result_code field in the Interface V_CHG Enable Operation Response payload.

That is, the *Greybus Operation Status* of the Operation Response message header shall only be used to indicate the result of the Greybus communication, exactly as described in *Greybus SVC Interface V_CHG Enable Response*.

Similarly, when the Interface V_CHG Disable Response message header has the *Greybus Operation Status* equal to GB_OP_SUCCESS, the value of V_CHG may be determined given the result_code field in the

Offset	Field	Size	Value	Description
0	result_code	1	Number	Result Code

Table 9.96: SVC Protocol Interface V_CHG Disable Response

Offset	Field	Size	Value	Description
0	intf_id	1	Number	Interface whose power consumption to limit
1	mw	2	Number	Power limit to be set in milli-watts

Table 9.97: SVC Protocol Interface Set V_SYS Power Limit Request

Offset	Field	Size	Value	Description
0	status	1	Number	*Greybus SVC Protocol Operation Status*

Table 9.98: SVC Protocol Interface Set V_SYS Power Limit Response

Operation Response payload, as described in Table 9.94. In particular, V_CHG is V_CHG_OFF if Response message header has the *Greybus Operation Status* equal to GB_OP_SUCCESS and the result_code field in the Operation Response payload is V_CHG_OK. V_CHG shall not have changed value as a result of processing the Request in any other combination of these two fields.

9.2.44 Greybus SVC Interface Set V_SYS Power Limit Operation

The AP Module uses this Operation to set a limit to the power delivered to an Interface. If this Operation is successfully exchanged, and the Interface draws more power through the V_SYS power bus than the amount specified in the Request, the Frame shall disable power delivery to the Interface by setting V_SYS to V_SYS_OFF. The SVC shall detect this event, and subsequently exchange a *Greybus SVC Interface Oops Operation*, with the AP Module in order to notify the AP Module about the overconsumption event. If this Operation fails the Interface power limit shall be set to the default limit and an error shall be sent to the AP.

Greybus SVC Interface Set V_SYS Power Limit Request

Table 9.97 defines the Greybus SVC Interface Set Power Limit Request payload.

The Greybus SVC Interface Set V_SYS Power Limit Request shall be sent only by the AP Module to the SVC. The Interface ID informs the SVC of the Interface it has to impose the power limit on. The mw field specifies the power limit in milli-watts.

Greybus SVC Interface Set V_SYS Power Limit Response

Table 9.98 defines the Greybus SVC Interface Set V_SYS Power Limit Response.

The Response contains the status of the Operation. For more information refer to *Greybus SVC Protocol Operation Status*.

9.3 Bootrom Protocol

Note: The Bootrom Protocol is deprecated for new Greybus implementations. The *Firmware Download Protocol* should be used instead.

While the Bootrom Protocol supports downloading *Interface Firmware* via Greybus, it lacks support for other features provided by the *Firmware Management Protocol* and other related Protocols, such as:

- Proper *Connection Management*
- Downloading *Interface Backend Firmware*
- Indicating to an Interface that it should store downloaded firmware on a non-volatile medium for later use

However, an implementation of this Protocol is part of a Greybus implementation which can no longer be changed. Because of this, AP Modules should maintain legacy compatibility for this protocol.

The Greybus Bootrom Protocol may be used by an Interface to download Interface Firmware via UniPro when the Interface does not have suitable Interface Firmware already available.

If an Interface implements this Greybus Protocol, the following additional requirements or exceptions hold:

- Any *Manifest* the Interface transmits to the AP via the *Control Protocol* shall contain exactly one CPort Descriptor with id field different than zero. The protocol field in that CPort Descriptor shall equal "Bootrom" (0x15), as described in Table 6.10.

  As a special exception, the Manifest may also contain one additional CPort Descriptor with id field equal to zero. This descriptor, if present, shall be ignored when received by the AP, along with any Bundle Descriptors it refers to, if any.

- The Interface shall implement the *Greybus Interface Attributes*. The value of the Ara Initialization Status attribute shall be set to one of 0x00000006 or 0x00000009 before any time the Interface sets the value of *MAILBOX*.

- If the AP detects one of these reserved Ara Initialization Status attribute values has been set, it shall not enable UniPro End-to-End Flow Control on any Connections it establishes with the Interface.

The Operations in the Greybus Bootrom Protocol are:

`int version(u8 offer_major, u8 offer_minor, u8 *major, u8 *minor);`
Refer to *Common Greybus Protocol Version Operation*.

`int ap_ready(void);`
The AP may send this Request to the Interface to confirm that the AP is now ready to receive Requests over the Connection, and the Interface can start the firmware download process. Until this Request is received by the Interface, it shall not send any Requests on the Connection.

`int firmware_size(u8 stage, u32 *size);`
The Interface requests from the AP the size of the Interface Firmware to load, specifying the stage of the boot sequence for which the Interface is requesting firmware. The AP then locates a suitable firmware blob, and associates that firmware blob with the requested boot stage until it next receives a Firmware Size Request, and responds with the blob's size in bytes, which must be nonzero.

`int get_firmware(u32 offset, u32 size, void *data);`
The Interface requests a finite stream of bytes in the firmware blob from the AP, passing its current offset into the firmware blob, and the size of the stream it currently needs. The AP responds with exactly the number of bytes requested, taken from the firmware blob currently associated with this Connection at the specified offset.

`int ready_to_boot(u8 status);`
The Interface implementing the Protocol requests permission from the AP to jump into the firmware blob it has loaded. The Request sent to the AP includes a status indicating whether the retrieved firmware blob is valid and secure, valid but insecure, or invalid. The AP decides whether to permit the module to boot in its current condition: if so, it sends a success code in its Response's status byte, otherwise, it sends an error code in its Response's status byte.

Bootrom Operation Type	Request Value	Response Value
Invalid	0x00	0x80
Protocol Version	0x01	0x81
Firmware Size	0x02	0x82
Get Firmware	0x03	0x83
Ready to Boot	0x04	0x84
AP Ready	0x05	0x85
(all other values reserved)	0x06..0x7e	0x86..0xfe
Invalid	0x7f	0xff

Table 9.99: Bootrom Operation Types

9.3.1 Greybus Bootrom Operations

Table 9.99 describes the Greybus Bootrom Operation Types and their values. A Message Type consists of an Operation Type combined with a flag (0x80) indicating whether the Operation is a Request or a Response.

9.3.2 Greybus Bootrom Protocol Version Operation

The Greybus Bootrom Protocol Version Operation is the *Common Greybus Protocol Version Operation* for the Bootrom Protocol.

Greybus implementations adhering to the Protocol specified herein shall specify the value zero (0) for the version_major and one (1) for the version_minor fields found in this Operation's Request and Response messages.

The Greybus Bootrom Protocol definition shall not change the required values for the version_major or version_minor fields in the future. This Protocol's Operations are fixed and shall not change in future versions of the Greybus Specification.

9.3.3 Greybus Bootrom Protocol AP Ready Operation

The Greybus Bootrom Protocol AP Ready Operation allows the AP to indicate that it is ready to receive Requests from the Interface over the Bootrom Connection.

Greybus Bootrom Protocol AP Ready Request

The Greybus Bootrom AP Ready Request Message has no payload.

Before receiving this Request, the Interface shall not send any Requests on the Bootrom Connection. After receiving this Request, the Interface may send Requests on the Bootrom Connection.

Greybus Bootrom Protocol AP Ready Response

The Greybus Bootrom AP Ready Response Message has no payload.

9.3.4 Greybus Bootrom Firmware Size Operation

The Greybus Bootrom Firmware Size Operation allows the Interface to submit a boot stage to the AP, so that the AP can associate a firmware blob with that boot stage and respond with its size. The AP keeps

Offset	Field	Size	Value	Description
0	stage	1	2	Stage is fixed to two.

Table 9.100: Bootrom Protocol Firmware Size Request

Offset	Field	Size	Value	Description
0	size	4	Number	Size of the blob in bytes

Table 9.101: Bootrom Protocol Firmware Size Response

the firmware blob associated with the boot stage until it receives another Firmware Size Request on the same Connection, but is not required to send identical firmware blobs in Response to different Requests with identical boot stages, even to the same Interface.

The boot stage parameter is fixed as a result of this Protocol's deprecation.

Greybus Bootrom Firmware Size Request

Table 9.100 defines the Greybus Bootrom Firmware Size Request payload. The Request supplies the boot stage of the Interface implementing the Protocol. The stage shall equal two.

Greybus Bootrom Firmware Size Response

Table 9.101 defines the Greybus Firmware Size Response payload. The Response supplies the size of the firmware blob which the AP has made available to the Interface for download.

9.3.5 Greybus Bootrom Get Firmware Operation

The Greybus Bootrom Get Firmware Operation allows the Interface to retrieve a stream of bytes at an offset within the firmware blob from the AP. The AP responds with the requested number of bytes from the Connection's associated firmware blob at the requested offset, or with an error status without payload if no firmware blob has yet been associated with this Connection or if the requested stream size exceeds the firmware blob's size minus the requested offset.

Greybus Bootrom Get Firmware Request

Table 9.102 defines the Greybus Bootrom Get Firmware Request payload. The Request specifies an offset into the firmware blob, and the size of the stream of bytes requested. The stream size requested must be less than or equal to the size given by the most recent Firmware Size Response (*Greybus Bootrom Firmware Size Response*) minus the offset; when it is not, the AP shall signal an error in its Response. The Interface is responsible for tracking its offset into the firmware blob as needed.

Offset	Field	Size	Value	Description
0	offset	4	Number	Offset into the firmware blob
4	size	4	Number	Size of the byte stream requested

Table 9.102: Bootrom Protocol Get Firmware Request

Offset	Field	Size	Value	Description
0	data	*size*	Data	Data from the firmware blob

Table 9.103: Bootrom Protocol Get Firmware Response

Offset	Field	Size	Value	Description
0	status	1	Number	*Greybus Bootrom Ready to Boot Firmware Blob Status*

Table 9.104: Bootrom Protocol Ready to Boot Request

Greybus Bootrom Get Firmware Response

Table 9.103 defines the Greybus Bootrom Get Firmware Response payload. The Response includes the stream of bytes requested by the Interface. In the case that the AP cannot fulfill the Request, such as when the requested stream size was greater than the total size of the firmware blob, it shall signal an error in the status byte of the Response header.

9.3.6 Greybus Bootrom Ready to Boot Operation

The Greybus Bootrom Ready To Boot Operation allows the requesting Interface to notify the AP that it has successfully loaded the Connection's currently associated firmware blob, and is able to execute that blob, as well as indicate the status of its firmware blob. The AP shall then send a Response empty of payload, indicating via the header's status byte whether or not it permits the Interface to continue booting.

The Interface shall send a Ready To Boot Request only when it has successfully loaded a firmware blob and can execute that firmware.

Greybus Bootrom Ready to Boot Request

Table 9.104 defines the Greybus Bootrom Ready To Boot Request payload. The Request gives the security status of its firmware blob.

Before sending this Request, the Interface should ensure that all outstanding *Get Firmware* Operation Requests it has sent have received Responses from the AP. The Interface should also not transmit any additional UniPro Segments with nonempty L4 payload on any Connection after those containing this Request payload. The effect of sending this Request under other conditions are undefined.

Greybus Bootrom Ready to Boot Firmware Blob Status

Table 9.105 defines the constants by which the Interface can indicate the status of its firmware blob to the AP in a Greybus Bootrom Ready to Boot Request.

Firmware Blob Status	Brief Description	Status Value
BOOT_STATUS_INVALID	Firmware blob could not be validated	0x00
BOOT_STATUS_INSECURE	Firmware blob is valid but insecure	0x01
BOOT_STATUS_SECURE	Firmware blob is valid and secure	0x02
	(Reserved Range)	0x03..0xFF

Table 9.105: Bootrom Ready to Boot Firmware Blob Statuses

Greybus Bootrom Ready to Boot Response

The Greybus Bootrom Ready to Boot Response has no payload.

In the case that the AP forbids the Interface from booting, it shall signal an error in the status byte of the Response Message's header. Otherwise, the status byte shall equal GB_OP_SUCCESS, indicating permission to boot.

Before sending the Response, the AP should ensure that all outstanding Control Protocol Requests to the Interface have received Responses. The effect of sending this Request under other conditions is undefined.

Provided that the recommendations for the Interface and the AP defined in this Protocol are followed, the Request and Response of the single Ready to Boot Operation exchanged between the Interface and the AP are the final UniPro Messages exchanged between the two.

When this occurs, the Interface may execute the downloaded firmware blob previously retrieved using this Protocol, and the following is permitted as a special case exception to restrictions made elsewhere in this Specification.

1. The Interface may treat its Control and Bootrom Connections as though they had been closed as described in *Connection Management*.
2. The Interface may, at most once, make a new Manifest available for retrieval to the AP, and thus send different Response payloads to the *Greybus Control Get Manifest Size Operation* and *Greybus Control Get Manifest Operation* Requests, should new Requests on the Control Connection be received later.

   The new Manifest shall not contain any CPort Descriptors whose protocol field equals "Bootrom" (0x15).
3. The Interface shall set the Ara Initialization Status attribute to a value different than 0x00000006 or 0x00000009.
4. The Interface may subsequently set *MAILBOX* to MAILBOX_GREYBUS, causing the SVC to exchange a *Greybus SVC Interface Mailbox Event Operation* with the AP. If the Interface does so, it shall:
   - ensure that if its Control CPort is subsequently reconnected, UniPro Flow Control Tokens shall subsequently be transmitted to the AP as buffer space for receiving Control Protocol Requests becomes available, and
   - subsequently respond to incoming *Control Protocol* Operation Requests as defined in that section if the Control CPort is connected and used for Greybus communication.
5. The AP should, after exchanging the Interface Mailbox Event Operation with the SVC, attempt to release system resources associated with the Control and Bootrom Connections to the Interface.
6. The AP should then attempt to open a Control Connection with the Interface, and retrieve its Manifest once more.

This sequence, when possible, is a **Legacy Mode Switch**. Though the Interface remains in the ENUMERATED Interface Lifecycle State throughout a Legacy Mode Switch and afterwards, its Manifest may change at most once as a result.

Chapter 10

Device Class Connection Protocols

This section defines a group of Protocols whose purpose is to provide a device abstraction for functionality commonly found on mobile handsets. Modules which implement at least one of the Protocols defined in this section, and which do not implement any of the Protocols defined below in *Bridged PHY Connection Protocols*, are said to be *device class conformant*.

Note: Two UniPro-based protocols will take the place of device class Protocol definitions in this section:

- MIPI CSI-3: for camera Modules
- JEDEC UFS: for storage Modules

10.1 Audio Protocol

This section defines the operations used on connections implementing the Greybus Audio Protocol. This Protocol allows an AP Module to manage audio devices present on a Module. The Protocol is strongly influenced by the *Advanced Linux Sound Architecture* (ALSA) and is designed to fit closely with it.

There are two types of Audio Connections defined by the Greybus Audio Protocol: *Audio Management Connections* and *Audio Data Connections*. Audio Management Connections are used to communicate management related operations. Audio Data Connections are used to stream audio data. All Greybus Audio Protocol operations except for the *Greybus Audio Send Data Operation* are sent over an Audio Management Connection. There shall be at least one Audio Data Connection associated with each Audio Management Connection.

The audio data shall be generated using *Pulse-Code Modulation*.

10.1.1 Required Functionality and Controls

A Greybus Audio Module shall have at least one endpoint (e.g., speaker, microphone, headphone jack, headset jack). There are two types of endpoints, input and output endpoints. Input endpoints are used when converting sounds into digital audio data that are sent to an AP Module (e.g., microphone). Output endpoints are used when converting digital audio data received from an AP Module into sounds (e.g., speaker). Some endpoints are used for both (e.g., headset jack).

Each endpoint shall support stereo audio data even when the underlying hardware does not. When the underlying hardware does not support stereo audio data, the module shall make the necessary conversions in order to support it. Exactly how that is done is left to the audio manufacturer.

Additionally, all endpoints shall support volume and mute controls for each channel.

10.1.2 Extended Functionality and Controls

A Greybus Audio Module may support functionality and controls that are far more elaborate than the required set. These extended features shall be supported by the AP Module downloading a matching MSP with the necessary support. How this is done is out of the scope of this document.

10.1.3 Audio Management Operations

The operations in the Greybus Audio Protocol are:

`int cport_shutdown(u8 phase);`
See *Common Greybus Protocol CPort Shutdown Operation*.

`int get_topology_size(u16 *descriptor_size);`
Returns the size of the audio device's topology data structure.

`int get_topology(struct gb_audio_descriptor *descriptor);`
Returns a data structure containing the audio device's supported Digital Audio Interfaces (DAIs), controls, widget, and how the DAIs and widgets can be connected.

```
int get_control(u8 control_id,
struct gb_audio_control_element_value *value);
```
Returns the current value of the specified control.

```
int set_control(u8 control_id,
struct gb_audio_control_element_value *value);
```
Sets a control to the specified value.

`int enable_widget(u8 widget_id);`
Enables the specified widget.

`int disable_widget(u8 widget_id);`
Disables the specified widget.

`int get_pcm(u16 data_cport, u64 *format, u32 *rate, u8 *channels u8 sig_bits);`
Returns the current PCM values of the specified DAI.

`int set_pcm(u16 data_cport, u64 format, u32 rate, u8 channels u8 ig_bits);`
Sets the PCM values of the specified DAI.

`int set_tx_data_size(u16 data_cport, u16 size);`
Sets the number of bytes in the audio data portion of Greybus audio messages going from the AP Module to the Audio Module.

`int get_tx_delay(u16 data_cport, u32 *delay);`
Returns the delay from the time the Audio Module receives the first Greybus Audio Messages until the first sound can be heard in microseconds.

`int activate_tx(u16 data_cport);`
Requests that the Audio Module begin accepting Greybus audio messages and output them on the configured audio widget.

`int deactivate_tx(u16 data_cport);`
Requests that the Audio Module stop accepting Greybus audio messages and stop outputting them on the configured audio endpoint.

`int set_rx_data_size(u16 data_cport, u16 size);`
Sets the number of bytes in the audio data portion of Greybus audio messages going from the Audio Module to the AP Module.

`int get_rx_delay(u16 data_cport, u32 *delay);`
Returns the delay from the time the Audio Module first receives a Activate RX Message until the first Greybus audio message is sent in microseconds (given the current PCM and RX data size configuration).

`int activate_rx(u16 data_cport);`
Requests that the Audio Module begin capturing audio data and sending it to the AP Module.

`int deactivate_rx(u16 data_cport);`
Requests that the Audio Module stop capturing audio data and sending it to the AP Module.

`int jack_event(u8 widget_id, u8 widget_type, u8 *event);`
Reports a jack related event to the AP Module.

`int button_event(u8 widget_id, u8 button_id, u8 *event);`
Reports a jack related event to the AP Module.

`int streaming_event(u16 data_cport, u8 *event);`
Reports a streaming related event to the AP Module.

`int send_data(u64 timestamp, u32 size, u8 *data);`
Sends an integer number of audio samples over an Audio Data Connection.

10.1.4 Greybus Audio Management Message Types

Table 10.1 describes the Greybus audio operation types and their values. A message type consists of an operation type combined with a flag (0x80) indicating whether the operation is a request or a response.

10.1.5 Greybus Audio CPort Shutdown Operation

The Greybus Audio CPort Shutdown Operation is the *Common Greybus Protocol CPort Shutdown Operation* for the Audio Protocol.

10.1.6 Greybus Audio Get Topology Size Operation

The Greybus Audio Get Topology Size operation allows the requester to determine the number of bytes required to hold the topology information structure returned by the *Greybus Audio Get Topology Operation*. If this operation fails, no further operations related to Greybus Audio shall occur.

Greybus Audio Get Topology Size Request

The Greybus Audio Get Topology Size request message has no payload.

Audio Operation Type	Request Value	Response Value
CPort Shutdown	0x00	0x80
Reserved	0x01	0x81
Get Topology Size	0x02	0x82
Get Topology	0x03	0x83
Get Control	0x04	0x86
Set Control	0x05	0x87
Enable Widget	0x06	0x88
Disable Widget	0x07	0x89
Get PCM	0x08	0x84
Set PCM	0x09	0x85
Set TX Data Size	0x0a	0x8a
Get TX Delay	0x0b	0x8b
Activate TX	0x0c	0x8c
Deactivate TX	0x0d	0x8d
Set RX Data Size	0x0e	0x8e
Get RX Delay	0x0f	0x8f
Activate RX	0x10	0x90
Deactivate RX	0x11	0x91
Jack Event	0x12	0x92
Button Event	0x13	0x93
Streaming Event	0x14	0x94
Send Data	0x15	0x95
(all other values reserved)	0x16..0x7e	0x96..0xfe
Invalid	0x7f	0xff

Table 10.1: Audio Operation Types

Offset	Field	Size	Value	Description
0	size	2	Number	Number of bytes of topology data

Table 10.2: Audio Get Topology Size Response

Offset	Field	Size	Value	Description
0	num_dais	1	Number	Number of DAI structures
1	num_controls	1	Number	Number of control structures
2	num_widgets	1	Number	Number of widget structures
3	num_routes	1	Number	Number of route structures
4	size_dais	4	Number	Size of audio_dais
8	size_controls	4	Number	Size of audio_controls
12	size_widgets	4	Number	Size of audio_widgets
16	size_routes	4	Number	Size of audio_routes
20	jack_type	4	Bit Mask	*Greybus Audio jack type Flag Bits*
24	dai[1]	120	Structure	*Greybus Audio DAI Structure*
...	...	120	Structure	*Greybus Audio DAI Structure*
24+120*(I-1)	dai[I]	120	Structure	*Greybus Audio DAI Structure*
24+size_dais	control[1]	XX	Structure	*Greybus Audio Control Structure*
...	...	XX	Structure	*Greybus Audio Control Structure*
24+size_dais+XX*(J-1)	control[J]	XX	Structure	*Greybus Audio Control Structure*
24+size_dais+size_controls	widget[1]	YY	Structure	*Greybus Audio Widget Structure*
...	...	YY	Structure	*Greybus Audio Widget Structure*
24+size_dais+size_controls+YY*(K-1)	widget[K]	YY	Structure	*Greybus Audio Widget Structure*
24+size_dais+size_controls+size_widgets	route[1]	4	Structure	*Greybus Audio Route Structure*
...	...	4	Structure	*Greybus Audio Route Structure*
24+size_dais+size_controls+size_widgets+4*(L-1)	route[L]	4	Structure	*Greybus Audio Route Structure*

Table 10.3: Audio Get Topology Response

Greybus Audio Get Topology Size Response

Table 10.2 describes the Greybus Audio Get Topology Size response. The response payload contains a two-byte value defining the number of bytes in the topology information structure returned by *Greybus Audio Get Topology Operation*. If the value returned is 0 no further operations related to Greybus Audio shall follow.

10.1.7 Greybus Audio Get Topology Operation

The Greybus Audio Get Topology operation allows the requester to retrieve audio topology information from an Audio Module. If this operation fails, no further operations related to Greybus Audio shall occur.

Greybus Audio Get Topology Request

The Greybus Audio Get Topology request message has no payload.

Greybus Audio Get Topology Response

Table 10.3 describes the Greybus Audio Get Topology response. The response payload contains a set of fixed size fields and a variable number of DAI, control, widget, and route structures.

Symbol	Brief Description	Mask Value
GB_AUDIO_JACK_HEADPHONE	Headphone	0x0000001
GB_AUDIO_JACK_MICROPHONE	Mic	0x0000002
GB_AUDIO_JACK_HEADSET	Headphone+Mic	0x0000003
GB_AUDIO_JACK_LINEOUT	Lineout	0x0000004
GB_AUDIO_JACK_MECHANICAL	Mechanical jack	0x0000008
GB_AUDIO_JACK_VIDEOOUT	Videoout	0x0000010
GB_AUDIO_JACK_AVOUT	Lineout+Videoout	0x0000014
GB_AUDIO_JACK_LINEIN	LineIn	0x0000020
GB_AUDIO_JACK_OC_HPHL	HPHL	0x0000040
GB_AUDIO_JACK_OC_HPHR	HPHR	0x0000080
GB_AUDIO_JACK_MICROPHONE2	Mic2	0x0000200
GB_AUDIO_JACK_ANC_HEADPHONE	Headphone+Mic+Mic2	0x0000203
GB_AUDIO_JACK_BTN_0	Media key	0x4000000
GB_AUDIO_JACK_BTN_1	Voicecommand key	0x2000000
GB_AUDIO_JACK_BTN_2	Volumeup key	0x1000000
GB_AUDIO_JACK_BTN_3	Volumedown key	0x0800000

Table 10.4: Audio Jack type Flag Bits

Offset	Field	Size	Value	Description
0	name	32	UTF-8	DAI Name
32	cport	2	Number	CPort for DAI Data Connection
34	capture	43	Structure	*Greybus Audio PCM Structure*
77	playback	43	Structure	*Greybus Audio PCM Structure*

Table 10.5: Audio DAI Structure

Greybus Audio jack type Flag Bits

Table 10.4 describes the audio jack types.

Greybus Audio DAI Structure

Table 10.5 describes the structure containing DAI information for Audio Modules.

Greybus Audio PCM Structure

Table 10.6 describes the structure containing PCM information for Audio Modules.

Offset	Field	Size	Value	Description
0	stream_name	32	UTF-8	Stream Name
32	formats	4	Bit Mask	*Greybus Audio Format Flags Bits*
36	rates	4	Bit Mask	*Greybus Audio Rate Flags Bits*
40	chan_min	1	Number	Minimum number of channels
41	chan_max	1	Number	Maximum number of channels
42	sig_bits	1	Number	Number of bits of content

Table 10.6: Audio PCM Structure

Symbol	Brief Description	Mask Value
GB_AUDIO_PCM_FMT_S8	Eight bit signed PCM data	0x00000001
GB_AUDIO_PCM_FMT_U8	Eight bit unsigned PCM data	0x00000002
GB_AUDIO_PCM_FMT_S16_LE	Sixteen bit signed PCM data, little endian	0x00000004
GB_AUDIO_PCM_FMT_U16_LE	Sixteen bit unsigned PCM data, little endian	0x00000008
GB_AUDIO_PCM_FMT_S16_BE	Sixteen bit signed PCM data, big endian	0x00000010
GB_AUDIO_PCM_FMT_U16_BE	Sixteen bit unsigned PCM data, big endian	0x00000020
GB_AUDIO_PCM_FMT_S24_LE	Twenty-four bit signed PCM data, little endian	0x00000040
GB_AUDIO_PCM_FMT_U24_LE	Twenty-four bit unsigned PCM data, little endian	0x00000080
GB_AUDIO_PCM_FMT_S24_BE	Twenty-four bit signed PCM data, big endian	0x00000100
GB_AUDIO_PCM_FMT_U24_BE	Twenty-four bit unsigned PCM data, big endian	0x00000200
GB_AUDIO_PCM_FMT_S32_LE	Thirty-two bit signed PCM data, little endian	0x00000400
GB_AUDIO_PCM_FMT_U32_LE	Thirty-two bit unsigned PCM data, little endian	0x00000800
GB_AUDIO_PCM_FMT_S32_BE	Thirty-two bit signed PCM data, big endian	0x00001000
GB_AUDIO_PCM_FMT_U32_BE	Thirty-two bit unsigned PCM data, big endian	0x00002000

Table 10.7: Audio Format Flag Bits

Symbol	Brief Description	Mask Value
GB_AUDIO_PCM_RATE_5512	5512 samples per second	0x00000001
GB_AUDIO_PCM_RATE_8000	8000 samples per second	0x00000002
GB_AUDIO_PCM_RATE_11025	11025 samples per second	0x00000004
GB_AUDIO_PCM_RATE_16000	16000 samples per second	0x00000008
GB_AUDIO_PCM_RATE_22050	22050 samples per second	0x00000010
GB_AUDIO_PCM_RATE_32000	32000 samples per second	0x00000020
GB_AUDIO_PCM_RATE_44100	44100 samples per second	0x00000040
GB_AUDIO_PCM_RATE_48000	48000 samples per second	0x00000080
GB_AUDIO_PCM_RATE_64000	64000 samples per second	0x00000100
GB_AUDIO_PCM_RATE_88200	88200 samples per second	0x00000200
GB_AUDIO_PCM_RATE_96000	96000 samples per second	0x00000400
GB_AUDIO_PCM_RATE_176400	176400 samples per second	0x00000800
GB_AUDIO_PCM_RATE_192000	192000 samples per second	0x00001000

Table 10.8: Audio Rate Flag Bits

Greybus Audio Format Flags Bits

Table 10.7 describes the audio data formats.

Greybus Audio Rate Flags Bits

Table 10.8 describes the audio data rates.

Greybus Audio Control Structure

Table 10.9 describes the structure containing control information for Audio Modules.

Greybus Audio Control Iface Type

Table 10.10 describes the audio control interface type.

Offset	Field	Size	Value	Description
0	name	32	UTF-8	Control Name
32	id	1	Number	Control ID
33	iface	1	Number	*Greybus Audio Control Iface Type*
34	data_cport	2	Number	Data CPort
36	access	4	Bit Mask	*Greybus Audio Control Access Rights Flags*
40	count	1	Number	Number of elements of this type
41	count_values	1	Number	Number of values (max=2, L/R)
42	info	XX	Structure	*Greybus Audio Control Element Info Structure*

Table 10.9: Audio Control Structure

Symbol	Brief Description	Mask Value
GB_AUDIO_IFACE_CARD	Global control	0x01
GB_AUDIO_IFACE_HWDEP	Hardware depedent device	0x02
GB_AUDIO_IFACE_MIXER	Mixer device	0x03
GB_AUDIO_IFACE_PCM	PCM device	0x04
GB_AUDIO_IFACE_RAWMIDI	Raw MIDI device	0x05
GB_AUDIO_IFACE_TIMER	Timer device	0x06
GB_AUDIO_IFACE_SEQUENCER	Sequencer device	0x07

Table 10.10: Audio Control Interface Type

Greybus Audio Control Access Rights Flags

Table 10.11 describes the audio control access rights.

Greybus Audio Control Element Info Structure

Table 10.12 describes the structure containing control element information for Audio Modules.

Greybus Audio Control Element Type

Table 10.13 describes the audio control element type.

Greybus Audio Control Element Value Range Union

Table 10.14 describes the union containing control element value ranges for Audio Modules.

Greybus Audio Control Element Integer Value Range Structure

Table 10.15 describes the structure containing a control element integer value range for Audio Modules.

Symbol	Brief Description	Mask Value
GB_AUDIO_ACCESS_READ	Read access	0x01
GB_AUDIO_ACCESS_WRITE	Write access	0x02

Table 10.11: Audio Control Access Rights Flag Bits

Offset	Field	Size	Value	Description
0	type	1	Bit Mask	*Greybus Audio Control Element Type*
1	dimen[1]	2	Number	First dimension
...	...	2	Number	...
7	dimen[4]	2	Number	Fourth dimension
9	value	XX	Union	*Greybus Audio Control Element Value Range Union*

Table 10.12: Audio Control Element Info Structure

Symbol	Brief Description	Mask Value
GB_AUDIO_CTL_ELEM_TYPE_BOOLEAN	Boolean	0x01
GB_AUDIO_CTL_ELEM_TYPE_INTEGER	32-bit Integer	0x02
GB_AUDIO_CTL_ELEM_TYPE_ENUMERATED	Enumerated type	0x03
GB_AUDIO_CTL_ELEM_TYPE_INTEGER64	64-bit Integer	0x06

Table 10.13: Audio Control Elemente Type

Offset	Field	Size	Value	Description
0	integer	12	Structure	*Greybus Audio Control Element Integer Value Range Structure*
0	integer64	24	Structure	*Greybus Audio Control Element Integer64 Value Range Structure*
0	enumerated	xxx	Structure	*Greybus Audio Control Element Enumerated Value Range Structure*

Table 10.14: Audio Control Element Value Range Union

Offset	Field	Size	Value	Description
0	min	4	Number	Minimum value
4	max	4	Number	Maximum value
8	step	4	Number	Increment amount

Table 10.15: Audio Control Element Integer Value Range Structure

Offset	Field	Size	Value	Description
0	min	8	Number	Minimum value
8	max	8	Number	Maximum value
16	step	8	Number	Increment amount

Table 10.16: Audio Control Element Integer64 Value Range Structure

Offset	Field	Size	Value	Description
0	items	4	Number	Number of items
4	names_length	2	Number	Length of names field
6	names	XX	UTF-8	Enumerated type names

Table 10.17: Audio Control Element Enumerated Value Range Structure

Greybus Audio Control Element Integer64 Value Range Structure

Table 10.16 describes the structure containing a control element integer64 value for range Audio Modules.

Greybus Audio Control Element Enumerated Value Range Structure

Table 10.17 describes the structure containing a control element enumerated value for range Audio Modules.

Greybus Audio Widget Structure

Table 10.18 describes the structure containing widget information for Audio Modules.

Greybus Audio Widget Type

Table 10.19 describes the audio widget type.

Greybus Audio Widget State

Table 10.20 describes the audio widget state.

Offset	Field	Size	Value	Description
0	name	32	UTF-8	Widget Name
32	name	32	UTF-8	Widget Stream Name
64	id	1	Number	Widget ID
65	type	1	Number	*Greybus Audio Widget Type*
66	state	1	Number	*Greybus Audio Widget State*
67	ncontrols	1	Number	Number of widget controls
68	ctl	XX	Structure	*Greybus Audio Control Structure*

Table 10.18: Audio Widget Structure

Widget Type	Value
Invalid	0x00
GB_AUDIO_WIDGET_TYPE_INPUT	0x01
GB_AUDIO_WIDGET_TYPE_OUTPUT	0x02
GB_AUDIO_WIDGET_TYPE_MUX	0x03
GB_AUDIO_WIDGET_TYPE_VIRT_MUX	0x04
GB_AUDIO_WIDGET_TYPE_VALUE_MUX	0x05
GB_AUDIO_WIDGET_TYPE_MIXER	0x06
GB_AUDIO_WIDGET_TYPE_MIXER_NAMED_CTL	0x07
GB_AUDIO_WIDGET_TYPE_PGA	0x08
GB_AUDIO_WIDGET_TYPE_OUT_DRV	0x09
GB_AUDIO_WIDGET_TYPE_ADC	0x0a
GB_AUDIO_WIDGET_TYPE_DAC	0x0b
GB_AUDIO_WIDGET_TYPE_MICBIAS	0x0c
GB_AUDIO_WIDGET_TYPE_MIC	0x0d
GB_AUDIO_WIDGET_TYPE_HP	0x0e
GB_AUDIO_WIDGET_TYPE_SPK	0x0f
GB_AUDIO_WIDGET_TYPE_LINE	0x10
GB_AUDIO_WIDGET_TYPE_SWITCH	0x11
GB_AUDIO_WIDGET_TYPE_VMID	0x12
GB_AUDIO_WIDGET_TYPE_PRE	0x13
GB_AUDIO_WIDGET_TYPE_POST	0x14
GB_AUDIO_WIDGET_TYPE_SUPPLY	0x15
GB_AUDIO_WIDGET_TYPE_REGULATOR_SUPPLY	0x16
GB_AUDIO_WIDGET_TYPE_CLOCK_SUPPLY	0x17
GB_AUDIO_WIDGET_TYPE_AIF_IN	0x18
GB_AUDIO_WIDGET_TYPE_AIF_OUT	0x19
GB_AUDIO_WIDGET_TYPE_SIGGEN	0x1a
GB_AUDIO_WIDGET_TYPE_DAI_IN	0x1b
GB_AUDIO_WIDGET_TYPE_DAI_OUT	0x1c
GB_AUDIO_WIDGET_TYPE_DAI_LINK	0x1d

Table 10.19: Audio Widget Type

Widget State	Value
Invalid	0x00
GB_AUDIO_WIDGET_STATE_DISABLED	0x01
GB_AUDIO_WIDGET_STATE_ENABLED	0x02

Table 10.20: Audio Widget State

Offset	Field	Size	Value	Description
0	source_id	1	Number	ID of source widget
1	destination_id	1	Number	ID of destination widget
2	control_id	1	Number	Control ID
3	index	1	Number	Index within the [enumerated] control

Table 10.21: Audio Route Structure

Offset	Field	Size	Value	Description
0	control_id	1	Number	Control ID
1	index	1	Number	Index

Table 10.22: Audio Get Control Request

Greybus Audio Route Structure

Table 10.21 describes the structure containing route information for Audio Modules.

10.1.8 Greybus Audio Get Control Operation

The Greybus Audio Get Control operation allows the requester to retrieve the current value of an audio control from an Audio Module.

Greybus Audio Get Control Request

Table 10.22 describes the Greybus Audio Get Control request. The request contains a one-byte control ID which uniquely identifies the audio control.

Greybus Audio Get Control Response

Table 10.23 describes the Greybus Audio Get Control response. The response payload contains a four-byte value specifying the current value for a control.

Greybus Audio Control Element Value Structure

Table 10.24 describes the structure containing control element identification and values for Audio Modules.

Greybus Audio Control Element Value Union

Table 10.25 describes the union containing control element values for Audio Modules.

Offset	Field	Size	Value	Description
0	value	63	Structure	*Greybus Audio Control Element Value Structure*

Table 10.23: Audio Get Control Response

Offset	Field	Size	Value	Description
0	timestamp	8	Number	Timestamp
8	value	8	Union	*Greybus Audio Control Element Value Union*

Table 10.24: Audio Control Element Value Structure

Offset	Field	Size	Value	Description
0	integer	8	Number	The 32-bit integer value
0	integer64	16	Number	The 64-bit integer value
0	enumerated	8	Number	Enumerated type item index

Table 10.25: Audio Control Element Value Union

10.1.9 Greybus Audio Set Control Operation

The Greybus Audio Set Control operation allows the requester to set the current value of an audio control on an Audio Module.

Greybus Audio Set Control Request

Table 10.26 describes the Greybus Audio Set Control request. The request contains a one-byte control ID which uniquely identifies the audio control and a 63-byte structure that specifies the new value.

Greybus Audio Set Control Response

The Greybus Audio Set Control response has no payload.

10.1.10 Greybus Audio Enable Widget Operation

The Greybus Audio Enable Widget operation allows the requester to enable a widget on an Audio Module.

Greybus Audio Enable Widget Request

Table 10.27 describes the Greybus Audio Enable Widget request. The request supplies the widget_id which uniquely identifies the widget.

Greybus Audio Enable Widget Response

The Greybus Audio Enable Widget response has no payload.

Offset	Field	Size	Value	Description
0	control_id	1	Number	Control ID
1	index	1	Number	Index
2	value	63	Structure	*Greybus Audio Control Element Value Structure*

Table 10.26: Audio Set Control Request

Offset	Field	Size	Value	Description
0	widget_id	1	Number	Widget Id

Table 10.27: Audio Enable Widget Request

Offset	Field	Size	Value	Description
0	widget_id	1	Number	Widget Id

Table 10.28: Audio Disable Widget Request

10.1.11 Greybus Audio Disable Widget Operation

The Greybus Audio Disable Widget operation allows the requester to disable a widget on an Audio Module.

Greybus Audio Disable Widget Request

Table 10.28 describes the Greybus Audio Disable Widget request. The request supplies the widget_id which uniquely identifies the widget.

Greybus Audio Disable Widget Response

The Greybus Audio Disable Widget response has no payload.

10.1.12 Greybus Audio Get PCM Operation

The Greybus Audio Get PCM operation allows the requester to retrieve the current audio PCM settings from an Audio Module.

Greybus Audio Get PCM Request

Table 10.29 describes the Greybus Audio Get PCM request. The request supplies the DAI CPort which uniquely identifies the DAI whose configuration is being queried.

Greybus Audio Get PCM Response

Table 10.30 describes the Greybus Audio Get PCM response. The response payload contains a four-byte value specifying the current PCM format, a four-byte value specifying the current sampling rate, a one-byte value specifying the number of audio channels, and a one-byte value specifying the number of significant bits of audio data in each channel.

Offset	Field	Size	Value	Description
0	data_cport	2	Number	Data CPort

Table 10.29: Audio Get PCM Request

Offset	Field	Size	Value	Description
0	format	4	Bit mask	*Greybus Audio Format Flags Bits*
4	rate	4	Bit mask	*Greybus Audio Rate Flags Bits*
8	channels	1	Number	Number of audio channels
9	sig_bits	1	Number	Number of significant bits of data

Table 10.30: Audio Get PCM Response

Offset	Field	Size	Value	Description
0	data_cport	2	Number	Data CPort
2	format	4	Bit mask	*Greybus Audio Format Flags Bits*
6	rate	4	Bit mask	*Greybus Audio Rate Flags Bits*
10	channels	1	Number	Number of audio channels
11	sig_bits	1	Number	Number of significant bits of data

Table 10.31: Audio Set PCM Request

10.1.13 Greybus Audio Set PCM Operation

The Greybus Audio Set PCM operation allows the requester to set the current audio PCM settings on an Audio Module.

Greybus Audio Set PCM Request

Table 10.31 describes the Greybus Audio Set PCM request. The request supplies the DAI CPort which uniquely identifies the DAI whose configuration is being set.

Greybus Audio Set PCM Response

The Greybus Audio Set PCM response has no payload.

10.1.14 Greybus Audio Set TX Data Size Operation

The Greybus Audio Set TX Data Size operation allows the requester to set the number of bytes of audio data contained in a *Greybus Audio Send Data Operation* going from the AP Module to an Audio Module.

Greybus Audio Set TX Data Size Request

Table 10.32 describes the Greybus Audio Set TX Data Size request. The request supplies the DAI CPort, which uniquely identifies the DAI, and the number of bytes of audio data that shall be contained in a *Greybus Audio Send Data Operation*. The size shall be an integer multiple of the number of bytes in a complete audio sample (i.e., number of bytes per channel times the number of channels).

Offset	Field	Size	Value	Description
0	data_cport	2	Number	Data CPort
2	size	2	Number	Number of audio data bytes

Table 10.32: Audio Set TX Data Size Request

Offset	Field	Size	Value	Description
0	data_cport	2	Number	Data CPort

Table 10.33: Audio Get TX Delay Request

Offset	Field	Size	Value	Description
0	delay	4	Number	Delay in microseconds

Table 10.34: Audio Get TX Delay Response

Greybus Audio Set TX Data Size Response

The Greybus Audio Set TX Data Size response has no payload.

10.1.15 Greybus Audio Get TX Delay Operation

The Greybus Audio Get TX Delay operation allows the requester to retrieve the amount of time the module requires from when the first *Greybus Audio Send Data Operation* is received until the first audio sample contained in that message is audible. The delay value is in microseconds.

Greybus Audio Get TX Delay Request

Table 10.33 describes the Greybus Audio Get TX Delay request. The request supplies the DAI CPort which uniquely identifies the DAI.

Greybus Audio Get TX Delay Response

Table 10.34 describes the Greybus Audio Get TX Delay response. The response payload contains a four-byte unsigned value specifying the amount of time the module requires from when the first *Greybus Audio Send Data Operation* is received until the first audio sample contained in that message is audible. The delay value is in microseconds.

10.1.16 Greybus Audio Activate TX Operation

The Greybus Audio Activate TX operation requests that the Audio Module prepare to receive audio data on the specified Audio Data Connection. The audio data shall be output using an audio output device (e.g., speaker).

Greybus Audio Activate TX Request

Table 10.35 describes the Greybus Audio Activate TX request. The request supplies the DAI CPort which uniquely identifies the DAI.

Offset	Field	Size	Value	Description
0	data_cport	2	Number	Data CPort

Table 10.35: Audio Activate TX Request

Offset	Field	Size	Value	Description
0	data_cport	2	Number	Data CPort

Table 10.36: Audio Deactivate TX Request

Offset	Field	Size	Value	Description
0	data_cport	2	Number	Data CPort
2	size	2	Number	Number of audio data bytes

Table 10.37: Audio Set RX Data Size Request

Greybus Audio Activate TX Response

The Greybus Audio Activate TX response has no payload.

10.1.17 Greybus Audio Deactivate TX Operation

The Greybus Audio Deactivate TX operation requests that the AP Module no longer accept audio data on the specified CPort. The AP Module may free any resources allocated by the corresponding *Greybus Audio Activate TX Operation*. Any audio data received on a deactivated Audio Data Connection shall be ignored.

Greybus Audio Deactivate TX Request

Table 10.36 describes the Greybus Audio Deactivate TX request. The request supplies the DAI CPort which uniquely identifies the DAI.

Greybus Audio Deactivate TX Response

The Greybus Audio Deactivate TX response has no payload.

10.1.18 Greybus Audio Set RX Data Size Operation

The Greybus Audio Set RX Data Size operation allows the requester to set the number of bytes of audio data contained in a *Greybus Audio Send Data Operation* going from an Audio Module to the AP Module.

Greybus Audio Set RX Data Size Request

Table 10.37 describes the Greybus Audio Set RX Data Size request. The request supplies the DAI CPort, which uniquely identifies the DAI, and the number of bytes of audio data that shall be contained in a *Greybus Audio Send Data Operation*. The size shall be an integer multiple of the number of bytes in a complete audio sample (i.e., number of bytes per channel times the number of channels).

Greybus Audio Set RX Data Size Response

The Greybus Audio Set RX Data Size response has no payload.

Offset	Field	Size	Value	Description
0	data_cport	2	Number	Data CPort

Table 10.38: Audio Get RX Delay Request

Offset	Field	Size	Value	Description
0	delay	4	Number	Delay in microseconds

Table 10.39: Audio Get RX Delay Response

10.1.19 Greybus Audio Get RX Delay Operation

The Greybus Audio Get RX Delay operation allows the requester to retrieve the amount of time the module requires from when the receive function is activated until the first *Greybus Audio Send Data Operation* is sent. The delay value is in microseconds.

Greybus Audio Get RX Delay Request

Table 10.38 describes the Greybus Audio Get RX Delay request. The request supplies the DAI CPort which uniquely identifies the DAI.

Greybus Audio Get RX Delay Response

Table 10.39 describes the Greybus Audio Get RX Delay response. The response payload contains a four-byte unsigned value specifying the amount of time the module requires from when the receive function is activated until the first *Greybus Audio Send Data Operation* is sent in the current configuration. The delay value is in microseconds.

10.1.20 Greybus Audio Activate RX Operation

The Greybus Audio Activate RX operation requests that the Audio Module begin capturing audio data and sending it to the AP Modules using the specified CPort.

Greybus Audio Activate RX Request

Table 10.40 describes the Greybus Audio Activate RX request. The request supplies the DAI CPort which uniquely identifies the DAI.

Greybus Audio Activate RX Response

The Greybus Audio Activate RX response has no payload.

Offset	Field	Size	Value	Description
0	data_cport	2	Number	Data CPort

Table 10.40: Audio Activate RX Request

Offset	Field	Size	Value	Description
0	data_cport	2	Number	Data CPort

Table 10.41: Audio Deactivate RX Request

Offset	Field	Size	Value	Description
0	widget_id	1	Number	Widget ID
1	type	1	Number	*Greybus Audio Widget Type*
2	event	1	Number	*Greybus Audio Jack Events*

Table 10.42: Audio Jack Event Request

10.1.21 Greybus Audio Deactivate RX Operation

The Greybus Audio Deactivate RX operation requests that the Audio Module stop capturing audio data and sending it to the AP Module. The AP Module may free any resources allocated by the corresponding *Greybus Audio Activate RX Operation*.

Greybus Audio Deactivate RX Request

Table 10.41 describes the Greybus Audio Deactivate RX request. The request supplies the DAI CPort which uniquely identifies the DAI.

Greybus Audio Deactivate RX Response

The Greybus Audio Deactivate RX response has no payload.

10.1.22 Greybus Audio Jack Event Operation

The Greybus Audio Jack Event operation allows the requester to notify the AP Module of audio jack events.

Greybus Audio Jack Event Request

Table 10.42 defines the Greybus Audio Jack Event Request. The request supplies a one-byte widget ID, a one-byte widget type, and the one-byte event being reported.

Greybus Audio Jack Events

Table 10.43 defines the Greybus Audio audio jack events and their values.

Symbol	Brief Description	Value
GB_AUDIO_JACK_EVENT_INSERTION	Device inserted into jack	0x01
GB_AUDIO_JACK_EVENT_REMOVAL	Device removed from jack	0x02

Table 10.43: Audio Events

Offset	Field	Size	Value	Description
0	widget_id	1	Number	Widget ID
1	button_id	1	Number	Button ID
2	event	1	Number	*Greybus Audio Button Events*

Table 10.44: Audio Button Event Request

Symbol	Brief Description	Value
GB_AUDIO_BUTTON_EVENT_PRESS	Button was pressed	0x01
GB_AUDIO_BUTTON_EVENT_RELEASE	Button was released	0x02

Table 10.45: Audio Events

Greybus Audio Jack Event Response

The Greybus Audio Jack Event response message has no payload.

10.1.23 Greybus Audio Button Event Operation

The Greybus Audio Button Event operation allows the requester to notify the AP Module of audio button events.

Greybus Audio Button Event Request

Table 10.44 defines the Greybus Audio Button Event Request. The request supplies a one-byte widget ID, a one-byte button ID, and the one-byte button event being reported.

Greybus Audio Button Events

Table 10.45 defines the Greybus Audio audio button events and their values.

Greybus Audio Button Event Response

The Greybus Audio Button Event response message has no payload.

10.1.24 Greybus Audio Streaming Event Operation

The Greybus Audio Streaming Event operation allows the requester to notify the AP Module of audio streaming events.

Greybus Audio Streaming Event Request

Table 10.46 defines the Greybus Audio Streaming Event Request. The request supplies the DAI CPort, which uniquely identifies the DAI, and the one-byte event being reported.

Offset	Field	Size	Value	Description
0	data_cport	2	Number	Data CPort
2	event	1	Number	*Greybus Audio Streaming Events*

Table 10.46: Audio Streaming Event Request

Symbol	Brief Description	Value
GB_AUDIO_STREAMING_EVENT_UNSPECIFIED	Catch-all for events not in this table	0x01
GB_AUDIO_STREAMING_EVENT_HALT	Streaming has halted	0x02
GB_AUDIO_STREAMING_EVENT_INTERNAL_ERROR	Internal error that should never happen	0x03
GB_AUDIO_STREAMING_EVENT_PROTOCOL_ERROR	Incorrect Operation order, etc.	0x04
GB_AUDIO_STREAMING_EVENT_FAILURE	Operation failed	0x05
GB_AUDIO_STREAMING_EVENT_UNDERRUN	No data to send	0x06
GB_AUDIO_STREAMING_EVENT_OVERRUN	Flooded by data	0x07
GB_AUDIO_STREAMING_EVENT_CLOCKING	Low-level clocking issue	0x08
GB_AUDIO_STREAMING_EVENT_DATA_LEN	Invalid message data length	0x09

Table 10.47: Audio Events

Greybus Audio Streaming Events

Table 10.47 defines the Greybus Audio audio streaming events and their values.

Greybus Audio Streaming Event Response

The Greybus Audio Streaming Event response message has no payload.

10.1.25 Greybus Audio Send Data Operation

The Greybus Audio Send Data Operation sends audio data over a Greybus Audio Data Connection. No response message shall be sent.

Greybus Audio Send Data Request

Table 10.48 Greybus Audio Send Data Request sends one or more complete audio samples. The size of the audio data is shall match the value specified in the most recent *Greybus Audio Set RX Data Size Operation*. It is a protocol error to send this message without first setting the data size.

Greybus Audio Send Data Response

There shall be no response message for the Greybus Audio send data request.

Offset	Field	Size	Value	Description
0	timestamp	8	Number	Time that audio sample is to be output
8	data	*size*	Data	Audio data

Table 10.48: Audio Protocol Send Data Request

10.2 Camera Protocol

10.2.1 System Architecture (Informative)

The Greybus Camera Device Class Protocol defines how Camera Modules communicate with AP Modules in a Greybus System.

MIPI has specified two interface protocols for camera integration relevant for Greybus Systems, CSI-2 and CSI-3. CSI-2 is a high-speed point-to-point unidirectional data transfer protocol. It defines an interface between a camera peripheral device and a host processor. CSI-2 usage is widespread in the mobile industry and is natively supported by most mobile Application Processors.

CSI-3 is a high-speed bidirectional communication protocol for camera systems. Based on UniPro, it specifies communication between camera sensors, image signal processors, bridge devices and host processors. The Greybus Camera Device Class Specification currently does not support CSI-3 devices within Modules.

The current Greybus Camera Device Class Protocol assumes that the AP Module and any Camera Modules in the system tunnel CSI-2 protocol data through the UniPro switch using the UniPro Bridge vendor proprietary protocol.

The Camera AP Bridge encapsulates the CSI-2 stream received from the camera into packets and sends them on the UniPro network. On the receiving side the Application Processor AP Bridge extracts the stream and outputs it over CSI-2 to the application processor.

The Greybus Camera Device Class Protocol describes transmission of image frames on the Greybus Camera Device Class Data Connections in terms of the CSI-2 interface on each side of the CSI-2 over UniPro tunnel. Control messages exchanged over UniPro outside of this are be described in terms of Greybus Camera Management Operations as for all other Greybus Device Class Protocols.

The specific protocol used to communicate between the Camera AP Bridge and the components internal to the Camera Module is considered to be implementation-specific and outside the scope of this document.

10.2.2 Connection

Camera Bundle

Camera Modules shall have at least one Greybus Interface that contains a Camera Bundle. The Camera Bundle, whose class is specified in Table 6.8, shall contain exactly two CPorts referred to as the Camera Management CPort and the Camera Data CPort. Protocol numbers assigned to these CPorts are specified in Table 6.10.

Camera Management Connection

A Camera Interface is configured and managed via its Camera Management Connection, which exchanges Operations defined by the Camera Management Protocol.

Camera Data Connection

Transmission of image data streams shall happen over a single CSI-2 port, through the Camera Data Connection, where CSI-2 packet transfer is implemented using the UniPro AP Bridge vendor proprietary CSI-2 encapsulation protocol.

The Camera Module may select the number of CSI-2 data lanes to setup between its CSI-2 transmitter and the AP Bridge CSI-2 receiver up to a maximum of four lanes.

10.2.3 Communications

Camera Management Protocol

The Camera Management Protocol is implemented by a set of Camera Management Operations, split in three categories:

- The Video Setup Operations, which handle capability enumeration and generally any retrieval of information from the Camera Interface, for the purpose of initializing the peer. Currently, the only defined Video Setup Operation is the *Greybus Camera Management Capabilities Operation*.
- The Video Streaming Operations, which control the video streams and their parameters such as image resolution and image format. Currently, the two defined video streaming Operations are the *Greybus Camera Management Configure Streams Operation* and *Greybus Camera Flush Streams Operation*.
- The Image Processing Operations, which control all the Camera Module image capture and processing algorithms and their parameters. Currently, the only defined image processing Operation is the *Greybus Camera Management Capture Streams Operation*.

Camera Modules shall implement all the Operations defined in this specification.

When explicitly allowed, Camera Modules may freely select implementation options but shall ensure that the options are compatible with each other as mandated by this specification, and shall report the selected options through capabilities.

Image Data Transmission

All Camera Modules shall support transmission of one video stream over CSI-2. Additionally, Camera Modules may support additional concurrent video streams, for instance, to transmit still images or auxiliary channels such as depth maps or resized images.

Camera Modules shall transmit all streams multiplexed over a single CSI-2 port and a single Virtual Channel using the Data Type Interleaving method defined by CSI-2. The Camera Module shall use Packet Level Interleaving as defined in section 9.13.1 of *[CSI-2]*.

Metadata Transmission

Metadata is defined as data other than image content that relates to a particular image frame. Metadata is used by Camera Modules to inform the image receiver about the characteristics of the transmitted frames, and the applied capture settings.

Metadata support is optional. However, when supported, it shall be implemented according to this specification.

The Greybus Camera Device Class Protocol defines two transport methods for metadata:

- using the *Greybus Camera Metadata Streams Operation* explicitly, through the Camera Management Connection.
- sending metadata along with image frames over the CSI-2 interface, through the Camera Data Connection.

Whenever possible, Camera Modules should use the CSI-2 transport to deliver metadata.

Camera Modules may implement neither, one or both of these transport methods. The supported methods shall be reported through the *Greybus Camera Management Capabilities Operation*

Camera Modules that support metadata transmission shall implement the CSI-2 frame number counter for all streams that can generate metadata.

CSI-2 Transport

When transmitting metadata over CSI-2, the Camera Module shall send the metadata using the same Virtual Channel number as the image frames and set the Data Type to User Defined 8-bit Data Type 8 (0x37).

Camera Modules should encode metadata using the properties and serialization format defined in the *Greybus Camera Properties* section of Greybus Camera Device Class specifications.

However, when this isnt possible or practical (for instance, when the Module hardware dictates the metadata format), Modules may choose to encode metadata using a custom method for metadata transmitted over CSI-2.

Metadata transmitted over CSI-2 using a custom encoding shall at minimum contain the ID of the associated request.

Metadata Operation

When transmitting metadata through the dedicated Operation, the Camera Module shall send a single *Greybus Camera Metadata Streams Operation Request* per image frame.

Metadata transmitted over Camera Management Connection using the *Greybus Camera Metadata Streams Operation Request* shall always be encoded as specified in the *Greybus Camera Properties* section of this specification.

10.2.4 Operational Model

Figure 10.1 describes the operational model of a Greybus Camera Bundle.

Upon a *Greybus Control Connected Operation*, that notifies the Camera Interface that a Connection to its Camera Management CPort has been successfully established, the Greybus Camera Device Class Protocol state machine is entered, in the UNCONFIGURED state.

The Camera Device Class state machine is exited when the Camera Management Connection is closed, either as notified by a *Greybus Control Disconnected Operation* referring to the Camera Management CPort, or as a consequence of forced removal.

The Greybus Camera Device Class state machine has 3 states: UNCONFIGURED, CONFIGURED, and STREAMING. Certain Operations are only valid in specific states, but the *Greybus Camera Management Capabilities Operation* may be used in any state, and shall always return the same set of camera capabilities.

The states that define the Camera Device Class state machine are:

- **UNCONFIGURED:** In this state the Camera Management Connection is operational. The state transitions to CONFIGURED state happens upon receipt of a *Greybus Camera Configure Streams Operation Request* if the following conditions are respected:
  - The Configure Streams Operation return GB_SUCCESS;
  - The Configure Streams Request does not contain any flag that explicitly require the Module to remain in UNCONFIGURED state;
  - The Module fully support the requested streams configuration;
- **CONFIGURED:** In this state the Bundle shall be ready to process *Greybus Camera Management Capture Streams Request* immediately as it receives them and then move to STREAMING state. Reception of a *Greybus Camera Configure Streams Operation Request* with a zero stream count returns the Bundle to the UNCONFIGURED state.
- **STREAMING:** In this state the Bundle transmits video frames in UniPro Messages encapsulating CSI-2 packets, sent over the Greybus Camera Device Class Data Connection. Greybus Capture Stream Requests can be queued, and once there are no active or queued Requests, the Bundle moves back to

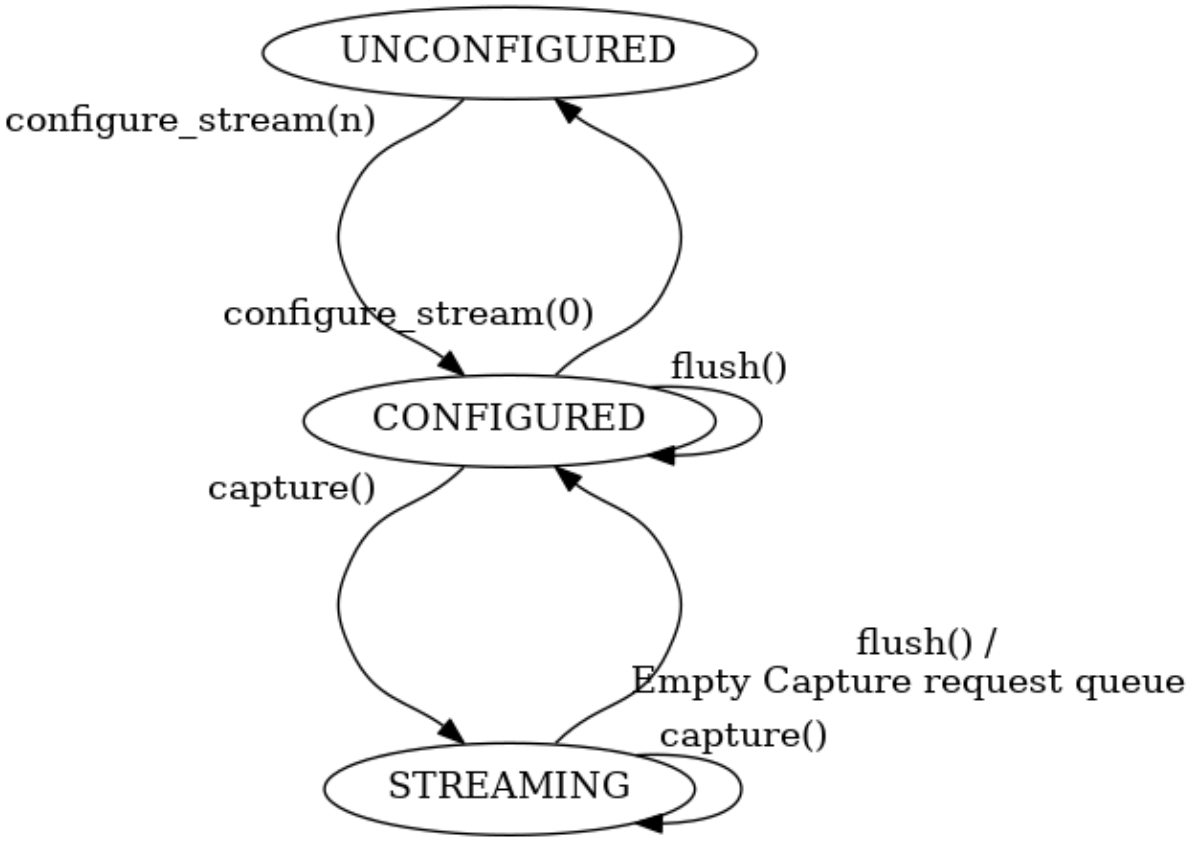


Fig. 10.1: Operational State Machine of a Greybus Camera Bundle

Camera Operation Type	Request Value	Response Value
CPort Shutdown	0x00	0x80
Reserved	0x01	0x81
Capabilities	0x02	0x82
Configure Streams	0x03	0x83
Capture	0x04	0x84
Flush	0x05	0x85
Metadata	0x06	N/A
(all other values reserved)	0x07..0x7f	0x87..0xff

Table 10.49: Camera Device Class operations

CONFIGURED state. Reception of a *Greybus Camera Flush Streams Operation Request* clears the queue of pending capture requests and also moves the Bundle to the CONFIGURED state.

10.2.5 Greybus Camera Management Protocol

Conceptually, the Operations in the Greybus Camera Management Protocol are:

`int cport_shutdown(u8 phase);`
See *Common Greybus Protocol CPort Shutdown Operation*.

`int capabilities(u8 *capabilities);`
Retrieve the list of camera capabilities.

`int configure_streams(u8 num_streams, u8 *flags, struct stream_config *streams);`
Prepares for or halts video streams.

`int capture(u32 request_id, u8 streams, u16 num_frames, const u8 *settings, u16 size);`
Enqueue a frame capture request.

`int flush(u32 *request_id);`
Removes all capture requests from the request queue.

`void metadata(u8 *metadata);`
Send image metadata to the AP.

All the above Operations shall be initiated by the AP Module, except for the *Greybus Camera Metadata Streams Operation* which is, instead, initiated by the Camera Module.

10.2.6 Greybus Camera Management Message Types

Table 10.49 describes the Greybus Camera Management Message Types and their values.

10.2.7 Greybus Camera Management CPort Shutdown Operation

The Greybus Camera Management CPort Shutdown Operation is the *Common Greybus Protocol CPort Shutdown Operation* for the Camera Management Protocol.

10.2.8 Greybus Camera Management Capabilities Operation

To allow support for various features and levels of complexity, the Greybus Camera Device Class defines optional features, which may be implemented by Camera Bundles.

Offset	Field	Size	Value	Description
0	capabilities	n	Data	Capabilities of Camera Module

Table 10.50: Camera Class Capabilities response

Using this Operation the sender can dynamically query the Camera Module for its capabilities.

Once the Camera Management Connection has been set up, the Camera Module shall respond to all Camera Management Capabilities Requests with the same set of capabilities. The Interface shall ensure identical capabilities are available as long as its Interface Lifecycle State remains ENUMERATED.

Greybus Camera Management Capabilities Request

The Greybus Camera Management Capabilities Request has no payload.

Greybus Camera Management Capabilities Response

The Greybus Camera Management Capabilities Response contains a variable-size capabilities block that shall conform to the format described in the *Greybus Camera Properties* section of this specification.

The Response payload is shown in Table 10.50.

10.2.9 Greybus Camera Management Configure Streams Operation

The Greybus Camera Management Configure Streams Operation is used to prepare the Camera Bundle for image transmission. When applied to a non-zero number of streams the Operation configures the Camera Module for capture with a list of stream parameters. A non-zero streams Request is only valid in the UNCONFIGURED state, the Camera Bundle shall reply with an empty payload and set the status to GB_OP_INVALID_STATE in all other states.

When instead applied to zero streams, the Operation removes the existing stream configuration, and moves back the Camera Bundle to the UNCONFIGURED state.

If the requested streams configuration is supported the Camera Bundle moves to the CONFIGURED state and shall be ready to process Capture Requests with as little delay as possible. In particular any time-consuming procedure which implements Module's specific power management shall be performed when moving to the CONFIGURED state. Camera Modules shall not be kept in the CONFIGURED state unnecessarily.

Streams shall be transmitted over CSI-2 using the reported Virtual Channels and Data Types.

All replies to Requests with the same set of parameters shall be identical.

Greybus Camera Configure Streams Operation Request

The Request specifies the number of streams to be configured. Up to four streams are supported. A Request with a number of streams higher than four shall be answered by an error Response with the status set to GB_OP_INVALID. A request with a zero number of streams remove the existing configuration and moves the Camera Bundle to the UNCONFIGURED state.

The flags field allows the AP Module to inform the Camera Bundle about special requirements applied to the Request. Accepted values for the Request flags field are listed in Table 10.52.

Offset	Field	Size	Value	Description
0	num_streams	1	Number	Number of streams. Between 0 and 4
1	flags	1	Number	Table 10.52
2	padding	2	0	Shall be set to 0
The following block appears num_streams times				
4+(i*8)	width	2	Number	Image width in pixels
6+(i*8)	height	2	Number	Image height in pixels
8+(i*8)	format	2	Number	Image Format
10+(i*8)	padding	2	0	Shall be set to 0

Table 10.51: Camera Class Configure Streams Request

Field (Bit)	Value	Description
0	TEST-ONLY	The requested configuration shall not be applied but Camera Module shall only verify it is supported or not.
1-7	Reserved	Shall be set to 0

Table 10.52: The flags bitmask in Camera Class Configure Stream Request

The TEST_ONLY bit of the Request flags field allows the AP to test a configuration without applying it. When the bit is set the Camera Module shall process the Request normally but stop from applying the configuration. The Module shall send the same Response as it would if the TEST_ONLY bit wasnt set and stay in the UNCONFIGURED state without modifying the device state.

The Request supplies a set of stream configurations with the desired image width, height and format for each stream, as show in Table 10.51. Both the width and height shall be multiples of 2. For each supplied stream configuration, the width, height and format fields shall be copied in the *Greybus Camera Configure Streams Operation Response* payload.

Greybus Camera Configure Streams Operation Response

The Camera Module reports its stream configuration in the Response message as shown in Table 10.53. The value of the num_streams field report the number of actually configured streams.

The flags field allows the Camera Bundle to provide additional information on the delivered Response. Accepted values for the Response flags field are listed in Table 10.54.

The ADJUSTED bit of the Response flags field is used to support negotiation of the stream configuration. The Camera Module may modify the requested configuration to match its capabilities. This includes lowering the number of requested streams, originally reported in the num_streams Request field, and modifying the width, height and format of each stream. The Module shall, in that case, reply with a configuration it can support, and set the ADJUSTED bit in the Response flags field. As a result the Camera Bundle shall stay in the UNCONFIGURED state without modifying the device state.

The data_rate field shall contain the total CSI-2 data rate expressed in Mbits per second, rounded up.

The Camera Module shall report in the Response, along with the (optionally adjusted) image format, width and height, the Virtual Channel number and Data Types for each stream, regardless of whether the response was adjusted or not

All Virtual Channel numbers shall be identical and between zero and three inclusive. All Data Types shall be different.

Offset	Field	Size	Value	Description
0	num_streams	1	Number	Number of streams. Between 0 and 4
1	flags	1	Number	Table 10.54
2	padding	2	0	Shall be set to 0
4	data_rate	4	Number	The CSI-2 data rate, expressed in Mbits per second (rounded up)
The following block appears num_streams times				
8+(i*16)	width	2	Number	Image width in pixels
10+(i*16)	height	2	Number	Image height in pixels
12+(i*16)	format	2	Number	Image Format
14+(i*16)	virtual_chan	1	Number	Virtual channel number
15+(i*16)	data_type[2]	2	Number	Data types for the stream
17+(i*16)	max_pkt_size	2	Number	The length in bytes of largets CSI Long Packet that transmits frame lines
19+(i*16)	padding	1	0	Shall be set to 0
20+(i*16)	max_size	4	Number	Maximum frame size in Bytes

Table 10.53: Camera Class Configure Streams Response

Field (Bit)	Value	Description
0	ADJUSTED	The requested configuration is not supported and has been adjusted
1-7	Reserved	Shall be set to 0

Table 10.54: The flags bitmask in Camera Class Configure Stream Response

Up to two data types can be used to identify different components of the same stream sent by a Camera Module. At least one data type shall be provided by the Camera Module, the second is optional and shall be set to the reserved 0x00 value if not used. The Data Types should be set to the CSI-2 Data Type value matching the streams formats if possible, and may be set to a User Defined 8-bit Data Type (0x30 to 0x37).

The Camera Module shall report in the max_pkt_size field the size in bytes of the largest CSI-2 Long Packet payload for the stream. CSI-2 Long packets are defined in section 9.1 of *[CSI-2]*.

For non-binary image formats Camera Modules shall transmit each line of the image individually in a single CSI-2 Long Packet. Image lines may have different sizes depending on the image format. The max_pkt_size is the size in bytes of the largest line of the image.

Binary image formats do not split the image in lines but encode it as a single block of bytes. Binary non-image formats transmit arbitrary non-image data in a single block of bytes. Camera Modules shall split the data in chunks in an implementation-defined way and send each chunk in a separate CSI-2 Long Packet. The max_pkt_size is then the size in bytes of the largest data chunk.

Binary and non-binary formats IDs are defined in the *Image Format Identifiers* section of this specifications.

10.2.10 Greybus Camera Management Capture Streams Operation

The Capture Streams Operation is used to submit a request for a new image frame transmission on the Camera Data Connection.

Upon receiving a valid Greybus Camera Management Capture Streams Request, the Camera Bundle shall return a Response immediately. The capture and transmission of the resulting frames via the Camera Data

Offset	Field	Size	Value	Description
0	request_id	4	number	An incrementing integer to uniquely identify the capture request
4	streams	1	bitmask	Bitmask of the streams included in the capture request
5	padding	1	0	Shall be set to 0
6	num_frames	2	number	Number of frames to capture (0 for infinite)
8	settings	n	data	Capture Request settings

Table 10.55: Camera Class Capture response

Connection occurs asynchronously to the processing of this Operation. These Requests shall be processed in the order they are received.

Camera Modules should minimize the delay between Requests by pre-processing pending Requests ahead of time as necessary.

When the first Request is queued, the Camera Module moves to the STREAMING state and starts transmitting frames as soon as possible. When the last Request completes the Bundle moves to the CONFIGURED state and stops transmitting frames immediately. Modules shall not transmit any UniPro Segment on the Camera Data Connection except as result of receiving a new Capture Request.

Greybus Camera Management Capture Streams Request

Each Camera Management Capture Stream Request contains an incrementing ID, a bitmask of the streams it affects, a number of frames to capture for all the streams in the bitmask and a list of settings to be applied to the transmitted image.

The AP shall set the request_id field in the Request payload to zero for the first Capture Streams Request it sends, and shall increment the value in this payload by one in each subsequent Request. If the value of the request_id field is not higher than the ID of the previous Request the Camera Bundle shall ignore the Request and set the reply status to GB_OP_INVALID.

Modules shall not use the value of the request_id field number for any purpose other than synchronizing the Capture Operation with the Flush and Metadata Operations. In particular, Camera Bundle shall accept Requests with IDs higher than the previous one by more than one.

The num_frames field contains the number of times the Request shall be repeated for all affected streams. Camera Modules shall capture and transmit one frame per stream for every repetition of the image capture request using the same capture settings. When the num_frames field is set to zero the image capture request shall be repeated indefinitely until the next Capture Operations Request, or a Flush Operation Request, is received.

The Capture Streams Request is only valid in the CONFIGURED and STREAMING states. The Camera Module shall set the Response status to GB_OP_INVALID_STATE in all other states.

The Capture Streams Request also contains a variable-size settings block that shall conform to the format described in the *Properties Section* of this specification. If no settings need to be applied for the Request the settings block shall have zero size.

Parameters for the Capture Stream Request are shown in Table 10.55

Greybus Camera Management Capture Streams Respose

The Camera Management Operation Capture Response message has no payload.

If the Capture Request streams bitmask field contains non-configured streams the Camera Module shall set the Response status to GB_OP_INVALID.

Offset	Field	Size	Value	Description
0	request_id	4	Number	The last Request that will be processed before the module stops transmitting frames

Table 10.56: Camera Class Flush response

10.2.11 Greybus Camera Flush Streams Operation

The Greybus Camera Management Flush Operation removes all Capture requests from the queue and stops frame transmission as soon as possible.

Delays are permitted to the extent they are necessary to flush hardware pipelines.

After finishing processing of that Request the module moves to the CONFIGURED state and shall not transmit any more frames.

The Request is only valid in the CONFIGURED and STREAMING states, the Camera Bundle shall reply with an empty payload and set the status to GB_OP_INVALID_STATE in all other states.

Greybus Camera Flush Streams Operation Request

The Camera Flush Request Message has no payload.

Greybus Camera Flush Streams Operation Respose

In order to allow synchronization, the Greybus Camera Management Flush Response reports the ID contained in the request_id field of the last processed *Greybus Camera Management Capture Streams Request*

When the Flush Operation is invoked while the Bundle is in the CONFIGURED state, the request_id field shall report the ID of the last frame transmitted over the Camera Data Connection. If no frames have been transmitted yet, the response_id field shall be set to zero.

Payload description for Flush Operation Response is reported in Table 10.56

10.2.12 Greybus Camera Metadata Streams Operation

The Greybus Camera Management Metadata Operation allows the Camera Module to transmit metadata associated with a frame though the Camera Management Connection.

The frame the delivered metadata is associated with is identified by the request_id field, the frame_number field and the stream_id field.

Greybus Camera Metadata Streams Operation Request

The Greybus Camera Management Metadata Request is sent by the Camera Module over the Camera Management Connection. It contains a variable-size metadata block that shall conform to the format described in the *Greybus Camera Properties* section of this specification.

If no metadata needs to be reported for a particular frame the metadata block shall have zero size.

The Greybus Camera Metadata Streams Operation Request is defined in Table 10.57

Offset	Field	Size	Value	Description
0	request_id	4	Number	The ID of the corresponding frame request
4	frame_number	2	Number	The CSI-2 frame number
6	stream_id	1	Number	The stream number
7	padding	1	0	Shall be set to zero
8	metadata	n	metadata	Metadata block

Table 10.57: Camera Class Metadata Request

10.2.13 Greybus Camera Properties

The Capabilities, Capture and Metadata operations modify or report the value of a set of Camera Module properties. Properties are defined as parameters that can report or modify the nature, state or operation of the Camera Module.

This section defines the structure of a property and a simple and efficient method to encode a set of property values in a binary data block that can be transmitted over Greybus.

Properties Definition

The Camera Class Protocol specifications defines properties through the following information.

- *name*

  A human readable string used to refer to the property in documentation.

- *key*

  An integer value that uniquely identifies the property.

- *data type*

  Type of the property value data that determines how the value is to be interpreted.

- *values*

  List, range or otherwise description of acceptable values for the property.

Properties defined in this specification are considered as standard Greybus Camera Device Class properties. Camera Module vendors are allowed to define additional properties to the extent allowed by the specification. If they chose to do so they shall define such additional properties using the mechanism described in this specification.

Property keys range from 0x0000 to 0xffff organized as follows:

- 0x0000 - 0x7fff: Standard Greybus Camera properties
- 0x8000 - 0x8fff: Vendor-specific properties
- 0x9000 - 0xffff: Reserved

A property stores a value using one of the following data types.

- int8: a signed 8-bit integer
- uint8: an unsigned 8-bit integer
- int32: a signed 32-bit integer
- uint32: an unsigned 32-bit integer
- int64: a signed 64-bit integer

Property Name	TAG	Type	Description
GB_CAM_SAMPLE_PROPERTY	0xXXXX	type[]	Description of property and intended use-cases

Table 10.58: Camera Class Property Example

Offset	Field	Size	Value	Description
0	size	2	number	Size of the payload, header excluded
2	nprops	2	number	Number of properties in the packet

Table 10.59: Camera Class Property Packet Header

- uint64: an unsigned 64-bit integer
- float: a single-precision (32-bit) IEEE 754 floating-point value, as defined in *[IEEE745]*
- double: a double-precision (64-bit) IEEE 754 floating-point value, as defined in *[IEEE745]*
- rational: a rational expressed as a 32-bit integer numerator and a 32-bit integer denominator. The denominator shall not be zero

Properties can also store an array of values of the same data type. In that case the property data type is postfixed with [] to denote the array nature of the data. For instance the data type of an array of 32-bit integers would be described as int32[].

When the property is directed to (or comes from) the Android Camera framework, only its name and TAG value are shown.

When a property, instead, is Greybus Camera specific, and not directed to the Android camera framework, a more detailed description and a range of accepted values (when applicable) is provided, as shown in figure 10.58.

Properties Value Encoding

Greybus Camera Device Class Operations need to transmit a set of property values.

A Property values set is an unordered list of property keys associated with values. To transport it over Greybus the set shall be serialized into an array of bytes called Properties Packets as follows.

Unless stated otherwise, all numerical fields shall be stored in little-endian format. Signed integers shall be encoded using a two's complement representation.

The memory of a Greybus Camera Device Class defined property is shown in Figure 10.2.

The packet starts with a fixed-size header that contains the payload size and the number of Properties it contains, as shown in Table 10.59.

The header is followed by a payload that stores Property value entries. Each entry contains the Property key, the Property value length and the Property value, as shown in table 10.60.

The packet shall not contain multiple entries with the same key. The order of payload entries is unspecified and shall not be relied upon when interpreting the content of the packet.

All value fields shall be padded to a multiple of 4 bytes. The size of the defined data types makes padding needed for int8 values only.

Values of array data type properties shall be encoded by storing the array elements sequentially without any space or padding between elements.

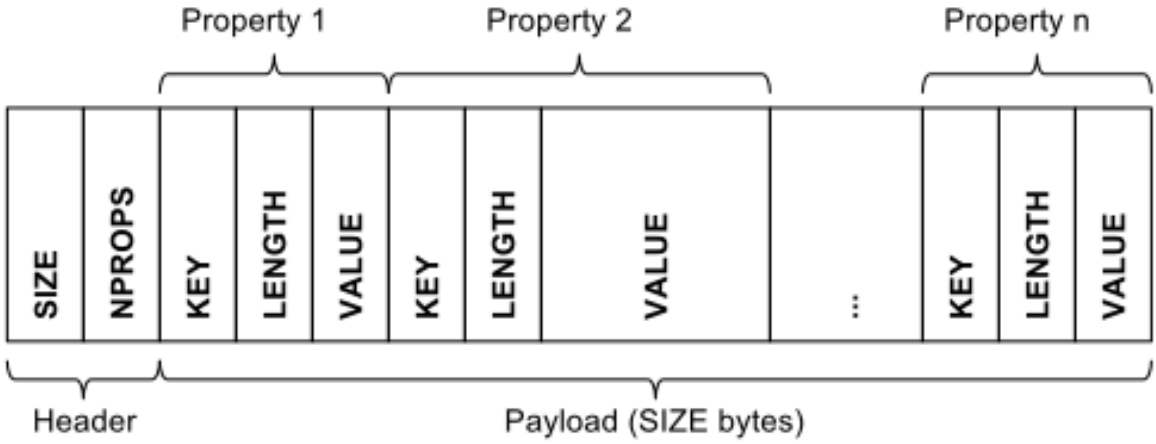


Fig. 10.2: Memory layout of a Greybus Camera Device Class Property Packet

Offset	Field	Size	Value	Description
0	key	2	number	Property key
2	length	2	number	Property length in bytes, padding excluded
4	value	n	property specific	Value of the property

Table 10.60: Camera Class Property Entry

Property Name	TAG	Property Name	TAG
COLOR_CORRECTION_AVAILABLE_ABERRATION_MODES	0x0004	SCALER_AVAILABLE_JPEG_SIZES	0x0d03
CONTROL_AE_AVAILABLE_ANTIBANDING_MODES	0x0112	SCALER_AVAILABLE_MAX_DIGITAL_ZOOM	0x0d04
CONTROL_AE_AVAILABLE_MODES	0x0113	SCALER_AVAILABLE_PROCESSED_MIN_DURATIONS	0x0d05
CONTROL_AE_AVAILABLE_TARGET_FPS_RANGES	0x0114	SCALER_AVAILABLE_PROCESSED_SIZES	0x0d06
CONTROL_AE_COMPENSATION_RANGE	0x0115	SCALER_AVAILABLE_RAW_MIN_DURATIONS	0x0d07
CONTROL_AE_COMPENSATION_STEP	0x0116	SCALER_AVAILABLE_RAW_SIZES	0x0d08
CONTROL_AF_AVAILABLE_MODES	0x0117	SCALER_AVAILABLE_INPUT_OUTPUT_FORMATS_MAP	0x0d09
CONTROL_AVAILABLE_EFFECTS	0x0118	SCALER_AVAILABLE_STREAM_CONFIGURATIONS	0x0d0a
CONTROL_AVAILABLE_SCENE_MODES	0x0119	SCALER_AVAILABLE_MIN_FRAME_DURATIONS	0x0d0b
CONTROL_AVAILABLE_VIDEO_STABILIZATION_MODES	0x011a	SCALER_AVAILABLE_STALL_DURATIONS	0x0d0c
CONTROL_AWB_AVAILABLE_MODES	0x011b	SCALER_CROPPING_TYPE	0x0d0d
CONTROL_MAX_REGIONS	0x011c	SENSOR_INFO_ACTIVE_ARRAY_SIZE	0x0f00
CONTROL_SCENE_MODE_OVERRIDES	0x011d	SENSOR_INFO_SENSITIVITY_RANGE	0x0f01
CONTROL_AVAILABLE_HIGH_SPEED_VIDEO_CONFIGURATIONS	0x0123	SENSOR_INFO_COLOR_FILTER_ARRANGEMENT	0x0f02
CONTROL_AE_LOCK_AVAILABLE	0x0124	SENSOR_INFO_EXPOSURE_TIME_RANGE	0x0f03
CONTROL_AWB_LOCK_AVAILABLE	0x0125	SENSOR_INFO_MAX_FRAME_DURATION	0x0f04
CONTROL_AVAILABLE_MODES	0x0126	SENSOR_INFO_PHYSICAL_SIZE	0x0f05
FLASH_INFO_AVAILABLE	0x0500	SENSOR_INFO_PIXEL_ARRAY_SIZE	0x0f06
HOT_PIXEL_AVAILABLE_HOT_PIXEL_MODES	0x0601	SENSOR_INFO_WHITE_LEVEL	0x0f07
JPEG_AVAILABLE_THUMBNAIL_SIZES	0x0707	SENSOR_INFO_TIMESTAMP_SOURCE	0x0f08
JPEG_MAX_SIZE	0x0708	SENSOR_INFO_LENS_SHADING_APPLIED	0x0f09
LENS_FACING	0x0805	SENSOR_INFO_PRE_CORRECTION_ACTIVE_ARRAY_SIZE	0x0f0a
LENS_POSE_ROTATION	0x0806	SENSOR_CALIBRATION_TRANSFORM1	0x0e05
LENS_POSE_TRANSLATION	0x0807	SENSOR_CALIBRATION_TRANSFORM2	0x0e06
LENS_INFO_AVAILABLE_APERTURES	0x0900	SENSOR_COLOR_TRANSFORM1	0x0e07
LENS_INFO_AVAILABLE_FILTER_DENSITIES	0x0901	SENSOR_COLOR_TRANSFORM2	0x0e08
LENS_INFO_AVAILABLE_FOCAL_LENGTHS	0x0902	SENSOR_FORWARD_MATRIX1	0x0e09
LENS_INFO_AVAILABLE_OPTICAL_STABILIZATION	0x0903	SENSOR_FORWARD_MATRIX2	0x0e0a
LENS_INFO_HYPERFOCAL_DISTANCE	0x0904	SENSOR_BLACK_LEVEL_PATTERN	0x0e0c
LENS_INFO_MINIMUM_FOCUS_DISTANCE	0x0905	SENSOR_MAX_ANALOG_SENSITIVITY	0x0e0d
LENS_INFO_SHADING_MAP_SIZE	0x0906	SENSOR_ORIENTATION	0x0e0e
LENS_INFO_FOCUS_DISTANCE_CALIBRATION	0x0907	SENSOR_PROFILE_HUE_SAT_MAP_DIMENSIONS	0x0e0f
LENS_INTRINSIC_CALIBRATION	0x080a	SENSOR_AVAILABLE_TEST_PATTERN_MODES	0x0e19
LENS_RADIAL_DISTORTION	0x080b	SHADING_AVAILABLE_MODES	0x1002
QUIRKS_METERING_CROP_REGION	0x0b00	STATISTICS_INFO_AVAILABLE_FACE_DETECT_MODES	0x1200
QUIRKS_TRIGGER_AF_WITH_AUTO	0x0b01	STATISTICS_INFO_MAX_FACE_COUNT	0x1202
QUIRKS_USE_ZSL_FORMAT	0x0b02	STATISTICS_INFO_AVAILABLE_HOT_PIXEL_MAP_MODES	0x1206
QUIRKS_USE_PARTIAL_RESULT	0x0b03	STATISTICS_INFO_AVAILABLE_LENS_SHADING_MAP_MODES	0x1207
REQUEST_MAX_NUM_OUTPUT_STREAMS	0x0c06	TONEMAP_MAX_CURVE_POINTS	0x1304
REQUEST_MAX_NUM_REPROCESS_STREAMS	0x0c07	TONEMAP_AVAILABLE_TONE_MAP_MODES	0x1305
REQUEST_PIPELINE_MAX_DEPTH	0x0c0a	LED_AVAILABLE_LEDS	0x1401
REQUEST_PARTIAL_RESULT_COUNT	0x0c0b	INFO_SUPPORTED_HARDWARE_LEVEL	0x1500
REQUEST_AVAILABLE_CAPABILITIES	0x0c0c	SYNC_MAX_LATENCY	0x1701
REQUEST_AVAILABLE_REQUEST_KEYS	0x0c0d	DEPTH_MAX_DEPTH_SAMPLES	0x1900
REQUEST_AVAILABLE_RESULT_KEYS	0x0c0e	DEPTH_AVAILABLE_DEPTH_STREAM_CONFIGURATIONS	0x1901
REQUEST_AVAILABLE_CHARACTERISTICS_KEYS	0x0c0f	DEPTH_AVAILABLE_DEPTH_MIN_FRAME_DURATIONS	0x1902
SCALER_AVAILABLE_FORMATS	0x0d01	DEPTH_AVAILABLE_DEPTH_STALL_DURATIONS	0x1903
SCALER_AVAILABLE_JPEG_MIN_DURATIONS	0x0d02	DEPTH_DEPTH_IS_EXCLUSIVE	0x1904

Fig. 10.3: Camera Device Class Capababilities IDs

Padding is only required at the end of the array to align its size to a multiple of 4 bytes.

Capabilities

Capabilities tags are reported by Camera Modules in order to describe their characteristics and their available features.

Capabilities tags defined in Table 10.3 are directed to the Android framework, for this reason their types, supported values and detailed description are documented by the Android system documentation.

Greybus Camera Device Class specific capabilities tags are defined in Table 10.4. Greybus Camera Device Class tags are used to describe Greybus Camera specific attributes and Camera Module shall include all of them in their reported Capabilities packets.

The accepted values for the reported GB_CAM_METADATA_FORMAT tag are listed in Table 10.61.

The accepted values for the reported GB_CAM_METADATA_TRANSPORT tag are listed in Table 10.62.

The GB_CAM_PRE_CROP_REGIONS specifies an array of uint32_t fields, whose values are listed in Table 10.63.

Property Name	TAG	Type	Description
GB_CAM_FEATURE_JPEG	0X7f00	bool	The Camera Module supports on-board JPEG encoding
GB_CAM_FEATURE_SCALER	0X7f01	bool	The Camera Module supports on-board image scaling
GB_CAM_METADATA_FORMAT	0x7f02	int8	Supported metadata format as defined in Table 10.61
GB_CAM_METADATA_TRANSPORT	0x7f03	int8	Supported metadata transport as defined in Table 10.62
GB_CAM_PER_FRAME_CONTROL	0x7f04	bool	The Camera Module support per-frame Control
GB_CAM_PRE_CROP_REGIONS	0x7f05	uint32[]	Field of view cropping, applied by Camera Module on its full pixel array size. Array members are shown in Table 10.63

Fig. 10.4: Camera Device Class Capabilities IDs

Property Name	Value	Description
METADATA_TRANSPORT_GB	0	The Camera Module sends metadata encoded as prescribed by this Specifications
METADATA_TRANSPORT_CUSTOM	1	The Camera Module sends metadata encoded in custom format

Table 10.61: Camera Device Class Accepted Metadata Format

Property Name	Value	Description
METADATA_TRANSPORT_NONE	0	The Camera Module does not send metadata
METADATA_TRANSPORT_CSI	1	The Camera Module sends metadata interleaved to image frames on the CSI-2 transport
METADATA_TRANSPORT_OP	2	The Camera Module sends metadata using the *Greybus Camera Metadata Streams Operation*

Table 10.62: Camera Device Class Accepted Metadata Transport Methods

Array Entry Index	Name	Description
0	Stream Format	Greybus wire image format, as defined in Table 10.66
1	Stream Width	Width, in pixels, of the video stream
2	Stream Height	Height, in pixels, of the video stream
3	Crop Top	Vertical offset, in pixels, of the top-left corner of the cropping rectangle
4	Crop Left	Horizontal offset, in pixels, of the top-left corner of the cropping rectangle
5	Crop Width	Width, in pixels, of the cropping rectangle
6	Crop Height	Height, in pixels, of the cropping rectangle

Table 10.63: Camera Device Class Pre Crop Region Array

Camera Modules can crop and/or scale the full sensor's field of view to achieve desired output resolutions. This property is used to describe, for each supported stream configuration, the associated cropping applied to the sensor's pixel array.

Camera Modules shall report, for each stream configuration listed in the SCALER_AVAILABLE_STREAM_CONFIGURATIONS property, the coordinates of the top-left corner of the associated cropping rectangle, expressed as displacement (in pixels) from the top-left corner of the sensor's active pixel array, and the cropping rectangle horizontal and vertical dimensions.

The data transported by GB_CAM_PRE_CROP_REGIONS property shall have an exact multiple of twenty-eight bytes as size, being composed by a number of tuples of seven elements, each of them four bytes long.

The number of seven element tuples reported in this property shall correspond to the number of elements reported in the SCALER_AVAILABLE_STREAM_CONFIGURATIONS property, one for each supported stream configuration. The elements shall be stored in the same order as the SCALER_AVAILABLE_STREAM_CONFIGURATIONS entries.

Capture Settings

Capture Setting tags are used to provide to the Camera Module the desired image processing settings it shall apply to the next captured frames. Camera Modules should minimize the delay required to apply the received settings as much as possible.

Capture Settings are generated by the Android framework, and sent on the wire along with each *Greybus Camera Management Capture Streams Request*. For this reason, their types, accepted values and detailed description are provided by the Android system documentation.

Metadata

Camera Modules should encode metadata using the properties and serialization format defined in this section.

However, when this isnt possible or practical (for instance when the module hardware dictates the metadata format), modules may chose to encode metadata using a custom method for metadata transmitted over CSI-2.

Property Name	TAG	Property Name	TAG
COLOR_CORRECTION_MODE	0x0000	JPEG_THUMBNAIL_SIZE	0x0706
COLOR_CORRECTION_TRANSFORM	0x0001	LENS_APERTURE	0x0800
COLOR_CORRECTION_GAINS	0x0002	LENS_FILTER_DENSITY	0x0801
COLOR_CORRECTION_ABERRATION_MODE	0x0003	LENS_FOCAL_LENGTH	0x0802
CONTROL_AE_ANTIBANDING_MODE	0x0100	LENS_FOCUS_DISTANCE	0x0803
CONTROL_AE_EXPOSURE_COMPENSATION	0x0101	LENS_OPTICAL_STABILIZATION_MODE	0x0804
CONTROL_AE_LOCK	0x0102	REQUEST_FRAME_COUNT	0x0c00
CONTROL_AE_MODE	0x0103	REQUEST_ID	0x0c01
CONTROL_AE_REGIONS	0x0104	REQUEST_INPUT_STREAMS	0x0c02
CONTROL_AE_TARGET_FPS_RANGE	0x0105	REQUEST_OUTPUT_STREAMS	0x0c04
CONTROL_AE_PRECAPTURE_TRIGGER	0x0106	REQUEST_TYPE	0x0c05
CONTROL_AF_MODE	0x0107	SCALER_CROP_REGION	0x0d00
CONTROL_AF_REGIONS	0x0108	SENSOR_EXPOSURE_TIME	0x0e00
CONTROL_AF_TRIGGER	0x0109	SENSOR_FRAME_DURATION	0x0e01
CONTROL_AWB_LOCK	0x010a	SENSOR_SENSITIVITY	0x0e02
CONTROL_AWB_MODE	0x010b	SENSOR_TEST_PATTERN_DATA	0x0e17
CONTROL_AWB_REGIONS	0x010c	SENSOR_TEST_PATTERN_MODE	0x0e18
CONTROL_CAPTURE_INTENT	0x010d	SHADING_MODE	0x1000
CONTROL_EFFECT_MODE	0x010e	STATISTICS_FACE_DETECT_MODE	0x1100
CONTROL_MODE	0x010f	STATISTICS_HOT_PIXEL_MAP_MODE	0x1103
CONTROL_SCENE_MODE	0x0110	STATISTICS_LENS_SHADING_MAP_MODE	0x1110
CONTROL_VIDEO_STABILIZATION_MODE	0x0111	TONEMAP_CURVE_BLUE	0x1300
FLASH_MODE	0x0402	TONEMAP_CURVE_GREEN	0x1301
HOT_PIXEL_MODE	0x0600	TONEMAP_CURVE_RED	0x1302
JPEG_GPS_COORDINATES	0x0700	TONEMAP_MODE	0x1303
JPEG_GPS_PROCESSING_METHOD	0x0701	TONEMAP_GAMMA	0x1306
JPEG_GPS_TIMESTAMP	0x0702	TONEMAP_PRESET_CURVE	0x1307
JPEG_ORIENTATION	0x0703	LED_TRANSMIT	0x1400
JPEG_QUALITY	0x0704	BLACK_LEVEL_LOCK	0x1600
JPEG_THUMBNAIL_QUALITY	0x0705		

Table 10.64: Camera Device Class Capture Settings IDs

Metadata transmitted over Greybus using the *Greybus Camera Metadata Streams Operation Request* shall always be encoded as specified in this section.

Metadata transmitted over CSI-2 using a custom encoding shall at minimum contain the ID of the associated request.

Table 10.65 define the IDs of metadata tags accepted by the Greybus Camera Device Class. Metadata tags are sent to the Android framework, for this reason their types, accepted values and detailed description are provided by the Android system documentation.

10.2.14 Greybus Camera Image Formats (Informative)

Introduction

Image formats specify how image data is structured to be sent over CSI-2.

A format defines the following properties.

- *The color encoding*

  Colors are encoded as three integer values called components. The components most frequently represent RGB or YUV values.

  In RGB encoding each pixel is described by Red, Green and Blue components. For sensors using a color filter array such as a Bayer filter, only one of the components is available for a given pixel.

  In YUV encoding each pixel is described by its Luma (Y), Blue Chroma (Cb or U) and Red Chroma (Cr or V). The red and blue chroma are collectively called chroma components or chroma and abbreviated UV.

- *The color depth*

  Also known as bit depth, the color depth is the number of bits used for each color component of a pixel.

  The Camera Class Protocol uses the same number of bits of all color components of a pixel. Typical values are 8, 10 and 12.

- *The components interleaving method*

  Components of a pixel may be transmitted together or separately. A format that transmits all components together is called a packed format. Figure 10.5 shows how the first three pixels of an image are transmitted in a packed RGB format.

Fig. 10.5: Three pixels encoded in packed RGB format

Property Name	TAG	Property Name	TAG
COLOR_CORRECTION_MODE	0x0000	LENS_POSE_TRANSLATION	0x0807
COLOR_CORRECTION_TRANSFORM	0x0001	LENS_INTRINSIC_CALIBRATION	0x080a
COLOR_CORRECTION_GAINS	0x0002	LENS_RADIAL_DISTORTION	0x080b
COLOR_CORRECTION_ABERRATION_MODE	0x0003	QUIRKS_PARTIAL_RESULT	0x0b04
CONTROL_AE_PRECAPTURE_ID	0x011e	REQUEST_ID	0x0c01
CONTROL_AE_ANTIBANDING_MODE	0x0100	REQUEST_OUTPUT_STREAMS	0x0c04
CONTROL_AE_EXPOSURE_COMPENSATION	0x0101	REQUEST_PIPELINE_DEPTH	0x0c09
CONTROL_AE_LOCK	0x0102	SCALER_CROP_REGION	0x0d00
CONTROL_AE_MODE	0x0103	SENSOR_EXPOSURE_TIME	0x0e00
CONTROL_AE_REGIONS	0x0104	SENSOR_FRAME_DURATION	0x0e01
CONTROL_AE_TARGET_FPS_RANGE	0x0105	SENSOR_SENSITIVITY	0x0e02
CONTROL_AE_PRECAPTURE_TRIGGER	0x0106	SENSOR_TIMESTAMP	0x0e10
CONTROL_AE_STATE	0x011f	SENSOR_NEUTRAL_COLOR_POINT	0x0e12
CONTROL_AF_MODE	0x0107	SENSOR_NOISE_PROFILE	0x0e13
CONTROL_AF_REGIONS	0x0108	SENSOR_PROFILE_HUE_SAT_MAP	0x0e14
CONTROL_AF_TRIGGER	0x0109	SENSOR_PROFILE_TONE_CURVE	0x0e15
CONTROL_AF_STATE	0x0120	SENSOR_TEST_PATTERN_DATA	0x0e17
CONTROL_AF_TRIGGER_ID	0x0121	SENSOR_TEST_PATTERN_MODE	0x0e18
CONTROL_AWB_LOCK	0x010a	SENSOR_ROLLING_SHUTTER_SKEW	0x0e1a
CONTROL_AWB_MODE	0x010b	SHADING_MODE	0x1000
CONTROL_AWB_REGIONS	0x010c	STATISTICS_FACE_DETECT_MODE	0x1100
CONTROL_CAPTURE_INTENT	0x010d	STATISTICS_FACE_LANDMARKS	0x1105
CONTROL_AWB_STATE	0x0122	STATISTICS_FACE_RECTANGLES	0x1106
CONTROL_EFFECT_MODE	0x010e	STATISTICS_FACE_SCORES	0x1107
CONTROL_MODE	0x010f	STATISTICS_LENS_SHADING_CORRECTION_MAP	0x110a
CONTROL_SCENE_MODE	0x0110	STATISTICS_LENS_SHADING_MAP	0x110b
CONTROL_VIDEO_STABILIZATION_MODE	0x0111	STATISTICS_PREDICTED_COLOR_GAINS	0x110c
FLASH_MODE	0x0402	STATISTICS_PREDICTED_COLOR_TRANSFORM	0x110d
FLASH_STATE	0x0405	STATISTICS_SCENE_FLICKER	0x110e
HOT_PIXEL_MODE	0x0600	STATISTICS_HOT_PIXEL_MAP_MODE	0x1103
JPEG_GPS_COORDINATES	0x0700	STATISTICS_HOT_PIXEL_MAP	0x110f
JPEG_GPS_PROCESSING_METHOD	0x0701	STATISTICS_LENS_SHADING_MAP_MODE	0x1110
JPEG_GPS_TIMESTAMP	0x0702	TONEMAP_CURVE_BLUE	0x1300
JPEG_ORIENTATION	0x0703	TONEMAP_CURVE_GREEN	0x1301
JPEG_QUALITY	0x0704	TONEMAP_CURVE_RED	0x1302
JPEG_THUMBNAIL_QUALITY	0x0705	TONEMAP_MODE	0x1303
JPEG_THUMBNAIL_SIZE	0x0706	TONEMAP_GAMMA	0x1306
LENS_APERTURE	0x0800	TONEMAP_PRESET_CURVE	0x1307
LENS_FILTER_DENSITY	0x0801	LED_TRANSMIT	0x1400
LENS_FOCAL_LENGTH	0x0802	BLACK_LEVEL_LOCK	0x1600
LENS_FOCUS_DISTANCE	0x0803	SYNC_FRAME_NUMBER	0x1700
LENS_OPTICAL_STABILIZATION_MODE	0x0804	REPROCESS_EFFECTIVE_EXPOSURE_FACTOR	0x1800
LENS_POSE_ROTATION	0x0806		

Table 10.65: Camera Device Class Metadata IDs

The same component arrangement is repeated for the remaining pixels of the image, line after line.

A format that transmit components separately is called a planar format. Figure 10.6 shows how an image may be transmitted in a planar YUV format.

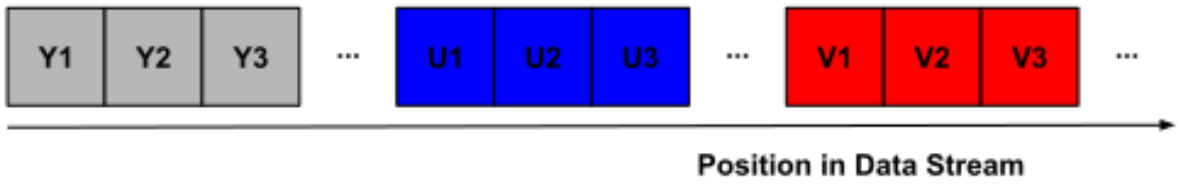


Fig. 10.6: Planar YUV image encoding

The ellipsis patterns (...) denote the rest of all luma, blue chroma and red chroma components respectively.

A format may also combine planar and packed components arrangements. Such a format is called semi-planar. In practice semi-planar formats are used with YUV encoding only and split components in a Y plane and a packed UV plane, as shown in Figure 10.7.

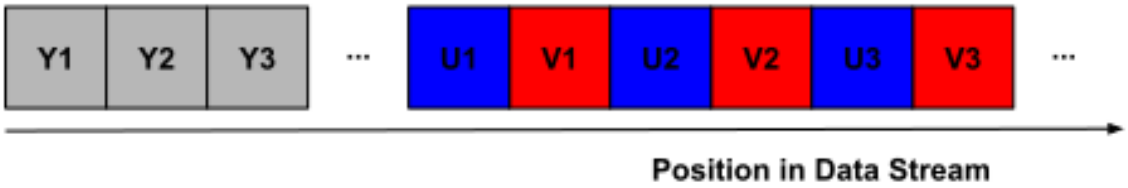


Fig. 10.7: Semi-planar YUV image encoding

In full planar YUV formats luma and chroma components are separated in three planes, one for each component.

In semi-planar YUV formats luma and chroma components are separated in two planes. The luma plane contains the luma components only, and the chroma plane contains the blue and red chroma components interleaved. Every semi-planar format comes in two chroma interleaving variants, in the UV or VU order.

- *The components ordering*

  Within a given interleaving method components may be arranged differently. For instance, a packed RGB format may transmit the three pixel components in the (R, G, B) or (B, G, R) order. Similarly, a planar YUV format may transfer the U plane before the V plane or the V plane before the U plane.

- *The components subsampling ratios*

  In YUV formats the chroma components may be sub-sampled horizontally and/or vertically to reduce bandwidth.

  The most common subsampling ratios are:

- 4:4:4 - No subsampling, every pixel has three color components
- 4:2:2 - Horizontal subsampling by 1/2
- 4:2:0 - Horizontal and vertical subsampling by 1/2

Figure 10.8 shows the relationship between pixels and luma and chroma components in a 8x2 pixels image.

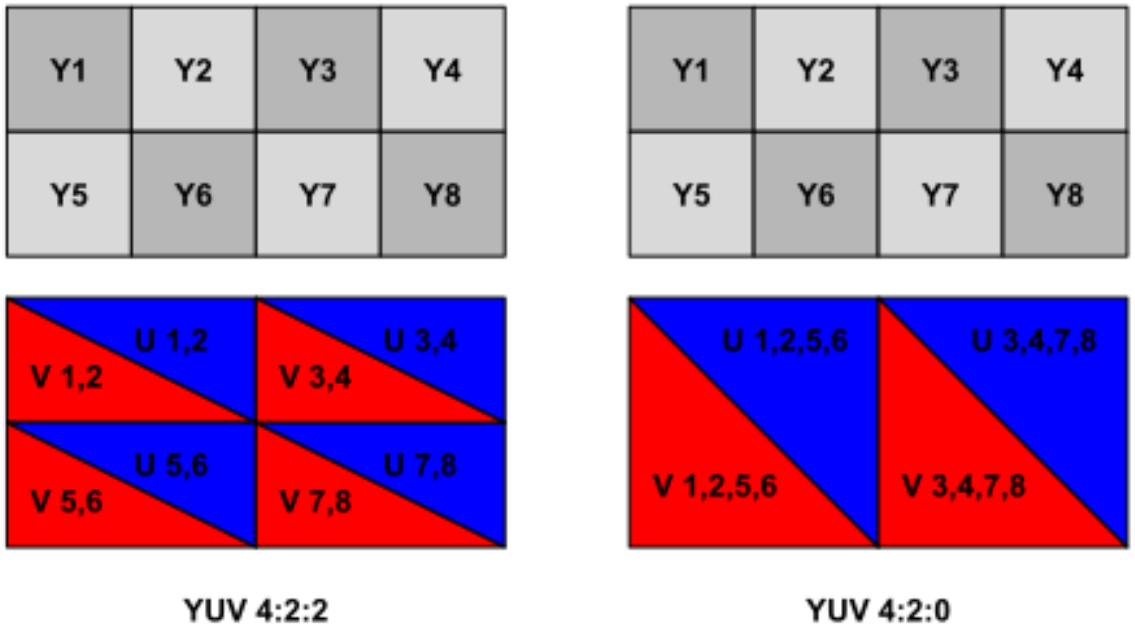


Fig. 10.8: YUV4:2:2 and YUV4:2:0 sampling examples

When subsampling chroma components the location of the components relatively to the pixels must be specified.

Data Transmission

Unless otherwise noted all image frames shall be transmitted in accordance with section 9 of *[CSI-2]*.

Camera Modules shall transmit all streams multiplexed over a single CSI-2 port and a single Virtual Channel, using the Data Type Interleaving method defined by CSI-2. The modules shall use Packet Level Interleaving as defined in section 9.13.1 of *[CSI-2]*.

Each format defined in this specification may add specific requirements.

In the following figures symbols shall be interpreted as follows.

- FS: Frame Start
- FE: Frame End
- PH: Packet Header
- PF: Packet Footer

Packed Formats

All packed formats are sent using a single CSI-2 Data Type

Packed YUV4:2:2 Image Format

This format transmits pixels encoded in YUV with 8 bits per component and a 4:2:2 subsampling. The image width shall be a multiple of two pixels.

Packed YUV 4:2:2 shall be transmitted as specified in section 11.2.4 of *[CSI-2]*.

Figure 10.9 illustrates how to transmit one line of the image.

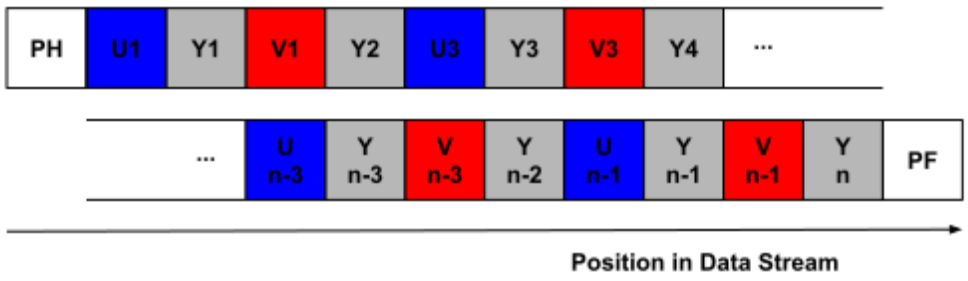


Fig. 10.9: Packed YUV4:2:2 image transmission format

Chroma components are spatially sampled at the same location as the luma components with a corresponding sample number.

Packed YUV4:2:0 Image Format

This format transmits pixels encoded in YUV with 8 bits per component and a 4:2:0 subsampling. The image width and height shall be multiples of two pixels.

Packed YUV 4:2:0 shall be transmitted as specified in sections 11.2.2 and 11.2.1 (legacy format) of *[CSI-2]*.

Figure 10.10 and 10.11 illustrate how to transmit image lines in YUV4:2:0 non-legacy and legacy format respectively.

In the non-legacy format even lines are twice as long as odd lines.

Chroma components x transmitted on odd line y and even line y+1 are spatially sampled in the middle of the four pixels at locations (x,y), (x+1,y), (x,y+1), (x+1,y+1).

Planar and Semi-Planar Formats

Planar and semi-planar formats separate pixel components in two or more planes.

Planes from one image frame shall be transmitted using line interleaving or plane sequential mode.

- In line interleaving mode, samples from a single line of a plane shall be transmitted in one or more consecutive CSI-2 packets. Lines shall then be interleaved as specified by each format. All samples from a line are thus transmitted contiguously relatively to samples from different planes of the same frame.
- In plane sequential mode, samples from a single plane shall be transmitted in consecutive CSI-2 packets. All samples from a plane are thus transmitted contiguously relatively to samples from different planes of the same frame.

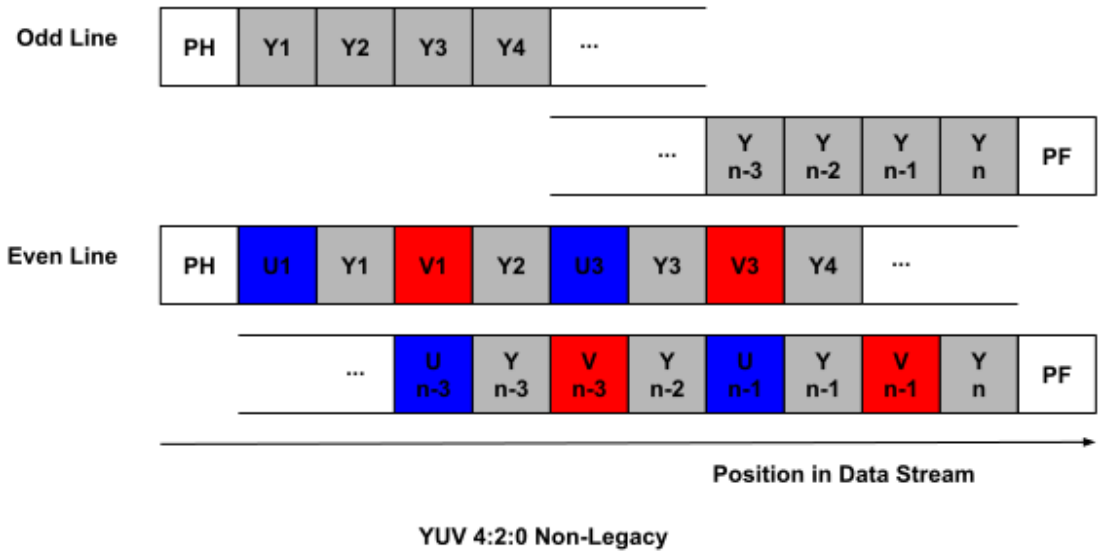


Fig. 10.10: Packed YUV4:2:0 Non-Legacy image transmission format

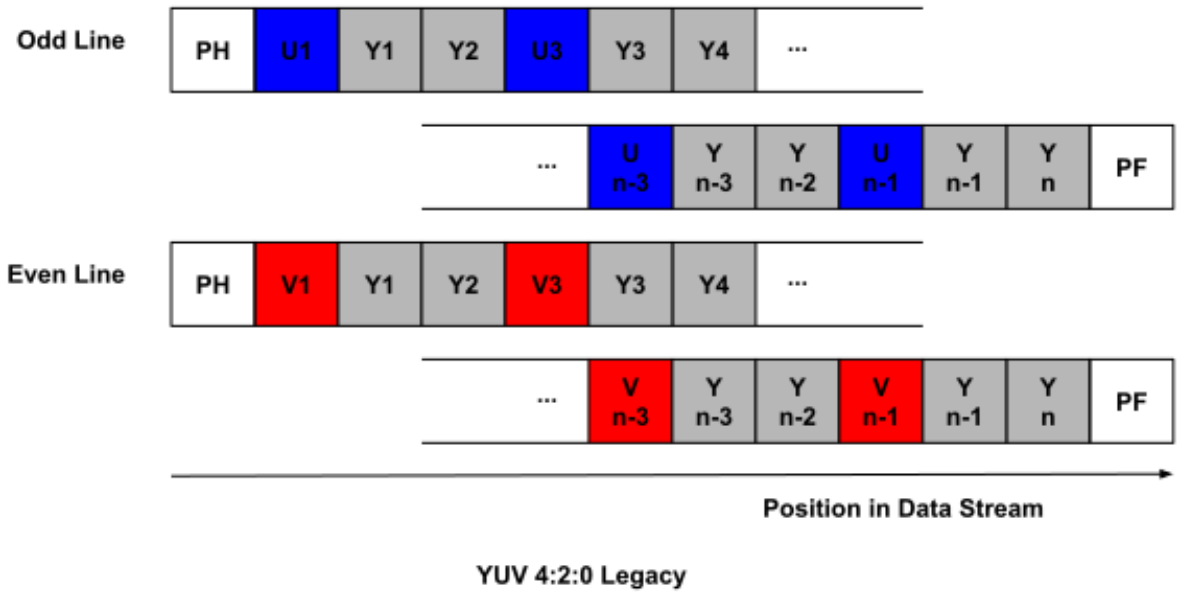


Fig. 10.11: Packed YUV4:2:0 Legacy image transmission format

In both modes packets from multiple streams may be interleaved freely.

Planar formats can come in two variants, one with all planes transmitted using a single Data Type, and one with planes transmitted using separate Data Types.

Semi-Planar YUV4:2:2 Image Format

These formats transmit pixels encoded in YUV with 8 bits per component and a 4:2:2 subsampling. The image width shall be a multiple of two pixels. The number of chroma line is equal to the number of luma lines.

The semi-planar YUV 4:2:2 formats are Greybus specific, they are not defined in *[CSI-2]*. They come in eight variants with all combinations of number of Data Types, U/V ordering and interleaving mode.

In line-interleaved mode a luma line is sent first followed by one chroma line. The chroma line contains samples related to the same pixels as the luma line. The same pattern repeats until the end of the frame. Figure 10.12 illustrates how to transmit one frame in line-interleaved mode with the UV chroma interleaving order.

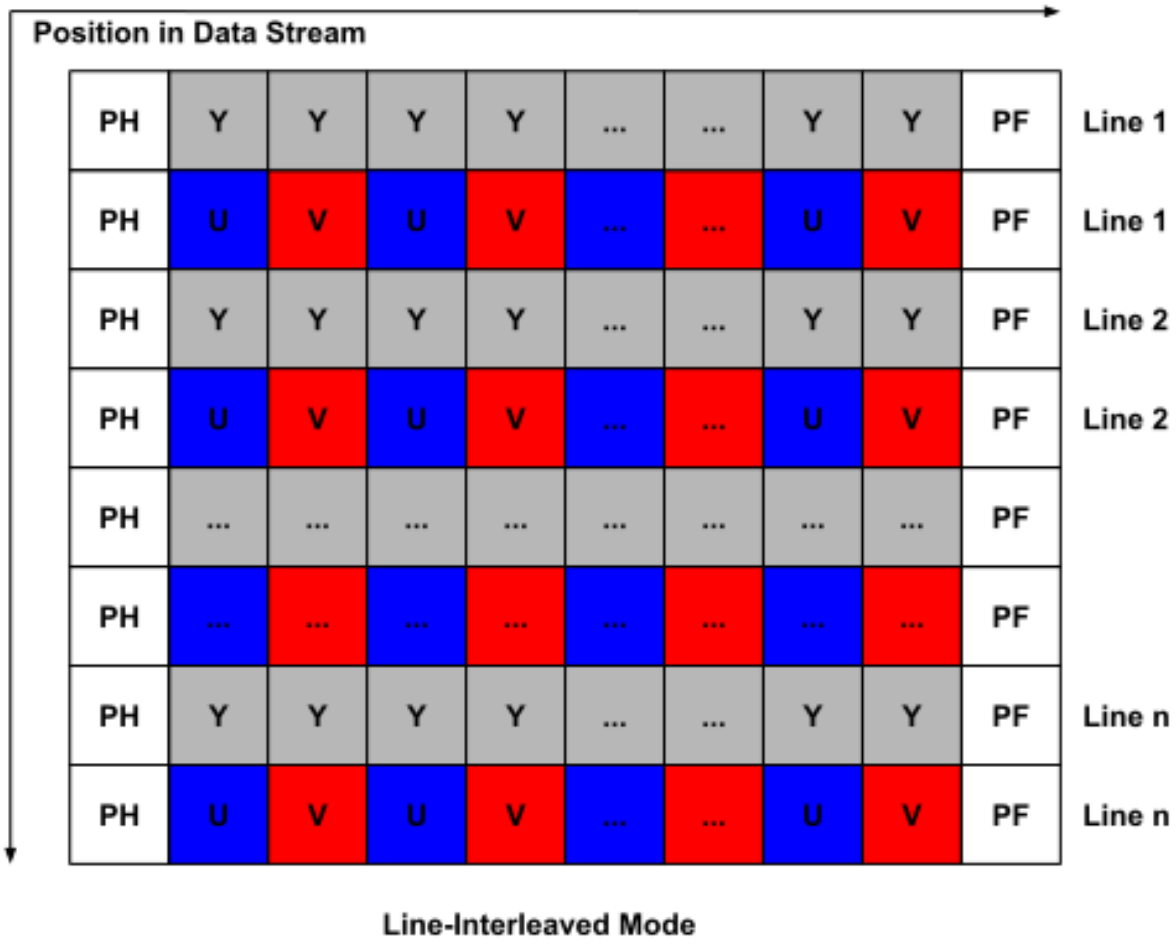


Fig. 10.12: Example of image transmission using line interleaving mode and YUV4:2:2 semi-planar sampling mode

In plane-interleaved mode all luma lines are sent first followed by all chroma lines. Figure 10.13 illustrates how to transmit one frame in plane sequential mode with the UV chroma interleaving order.

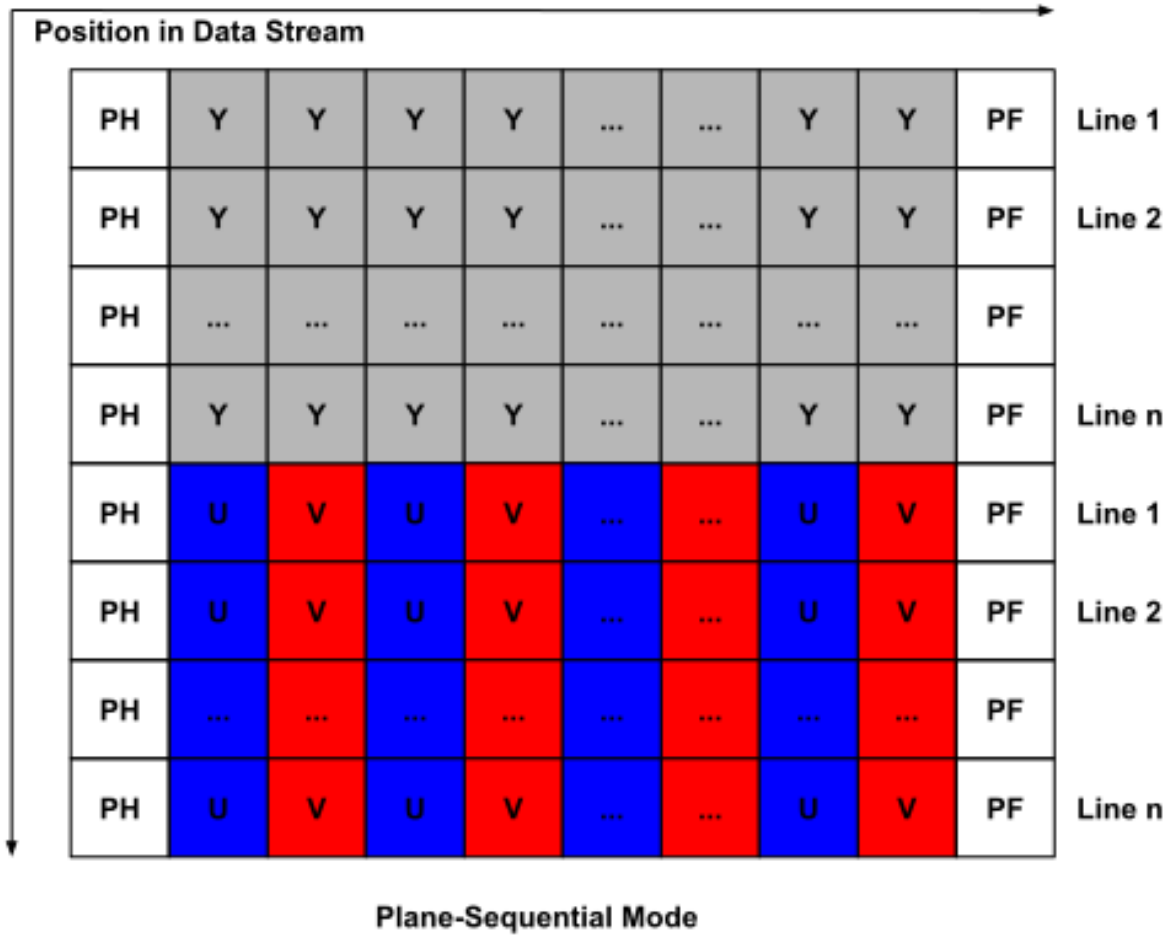


Fig. 10.13: Example of image transmission using plane interleaving mode and YUV4:2:2 semi-planar sampling mode

Chroma components are spatially sampled at the same location as the luma components with a corresponding sample number.

Semi-Planar YUV4:2:0 Image Format

These formats transmit pixels encoded in YUV with 8 bits per component and a 4:2:0 subsampling. The image width and height shall be multiples of two pixels. The number of chroma lines is half the number of luma lines. Each chroma line stores values related to two lines of pixels.

The semi-planar YUV 4:2:0 formats are Greybus specific, they are not defined in *[CSI-2]*. They come in eight variants with all combinations of number of Data Types, U/V ordering and interleaving mode.

In line-interleaved mode lines are sent in groups of two luma lines and one chroma line. The group starts with an odd luma line, followed by one chroma line, followed by an even luma line. The chroma line contains samples related to the same pixels as the two luma lines. The same pattern repeats until the end of the frame.

Figure 10.14 illustrates how to transmit one frame in line-interleaved mode with the UV chroma interleaving order.

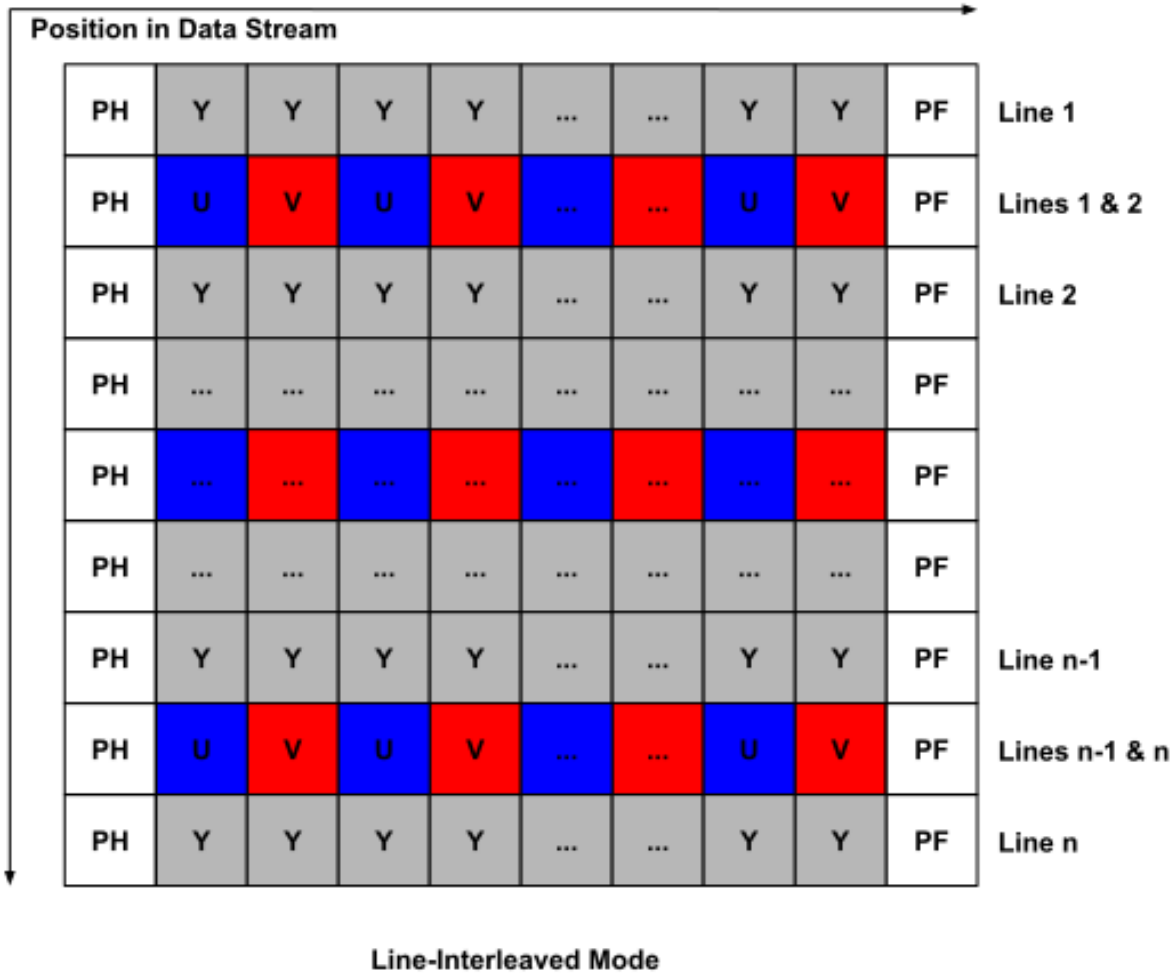


Fig. 10.14: Example of image transmission using line interleaving mode and YUV4:2:0 semi-planar sampling mode

In plane-interleaved mode all luma lines are sent first followed by all chroma lines. Figure 10.15 illustrates how to transmit one frame in plane sequential mode with the UV chroma interleaving order.

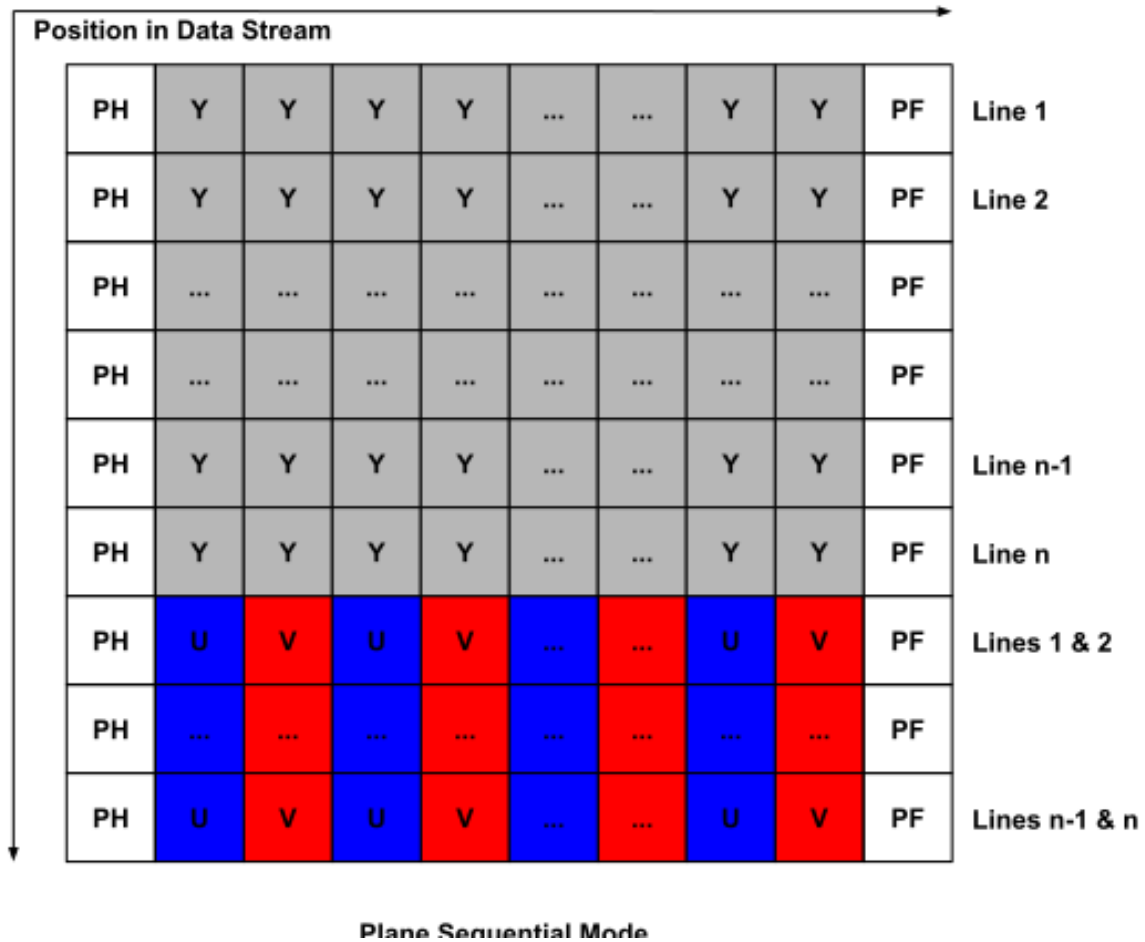


Fig. 10.15: Example of image transmission using plane interleaving mode and YUV4:2:0 semi-planar sampling mode

Chroma components x transmitted on odd line y and even line y+1 are spatially sampled in the middle of the four pixels at locations (x,y), (x+1,y), (x,y+1), (x+1,y+1).

Planar YUV4:2:2 Image Format

These formats transmit pixels encoded in YUV with 8 bits per component and a 4:2:2 subsampling. The image width shall be a multiple of two pixels. The number of chroma line is equal to the number of luma lines.

The planar YUV 4:2:2 formats are Greybus specific, they are not defined in *[CSI-2]*. They come in two variants for U/V ordering.

Only plane-interleaved is supported. All luma lines are sent first, followed by all blue or red chroma lines, followed by all remaining (red or blue) chroma lines. Figure 10.16 illustrates how to transmit one frame in plane sequential mode with the UV chroma order.

Chroma components are spatially sampled at the same location as the luma components with a corresponding sample number.

Planar YUV4:2:0 Image Format

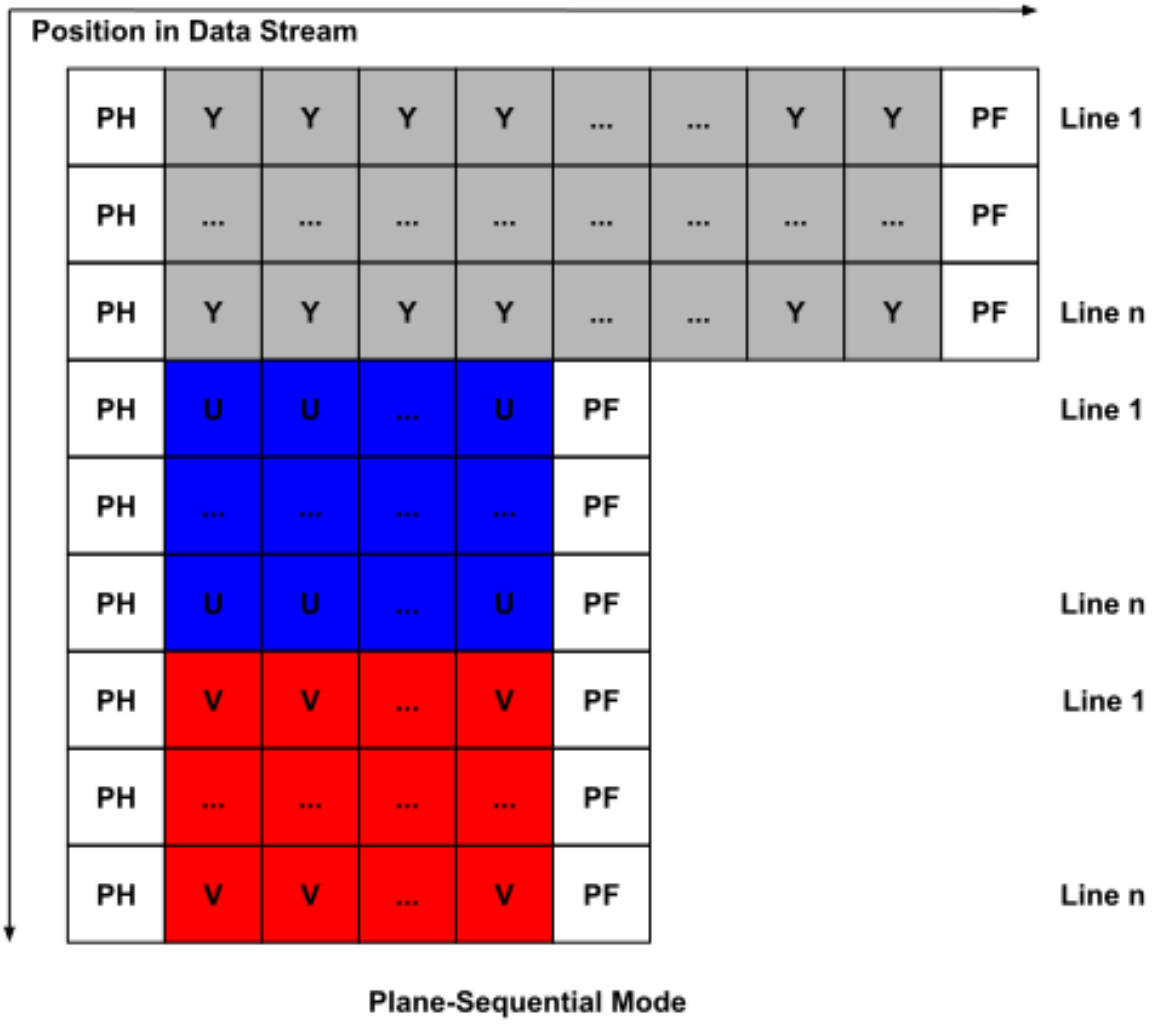


Fig. 10.16: Example of image transmission using plane interleaving mode and YUV4:2:2 planar sampling mode

These formats transmit pixels encoded in YUV with 8 bits per component and a 4:2:0 subsampling. The image width and height shall be multiples of two pixels. The number of chroma lines is half the number of luma lines. Each chroma line stores values related to two lines of pixels.

The planar YUV 4:2:0 formats are Greybus specific, they are not defined in *[CSI-2]*. They come in two variants for U/V ordering.

Only plane-interleaved is supported. All luma lines are sent first, followed by all blue or red chroma lines, followed by all remaining (red or blue) chroma lines.

Figure 10.17 illustrates how to transmit one frame in plane sequential mode with the UV chroma order.

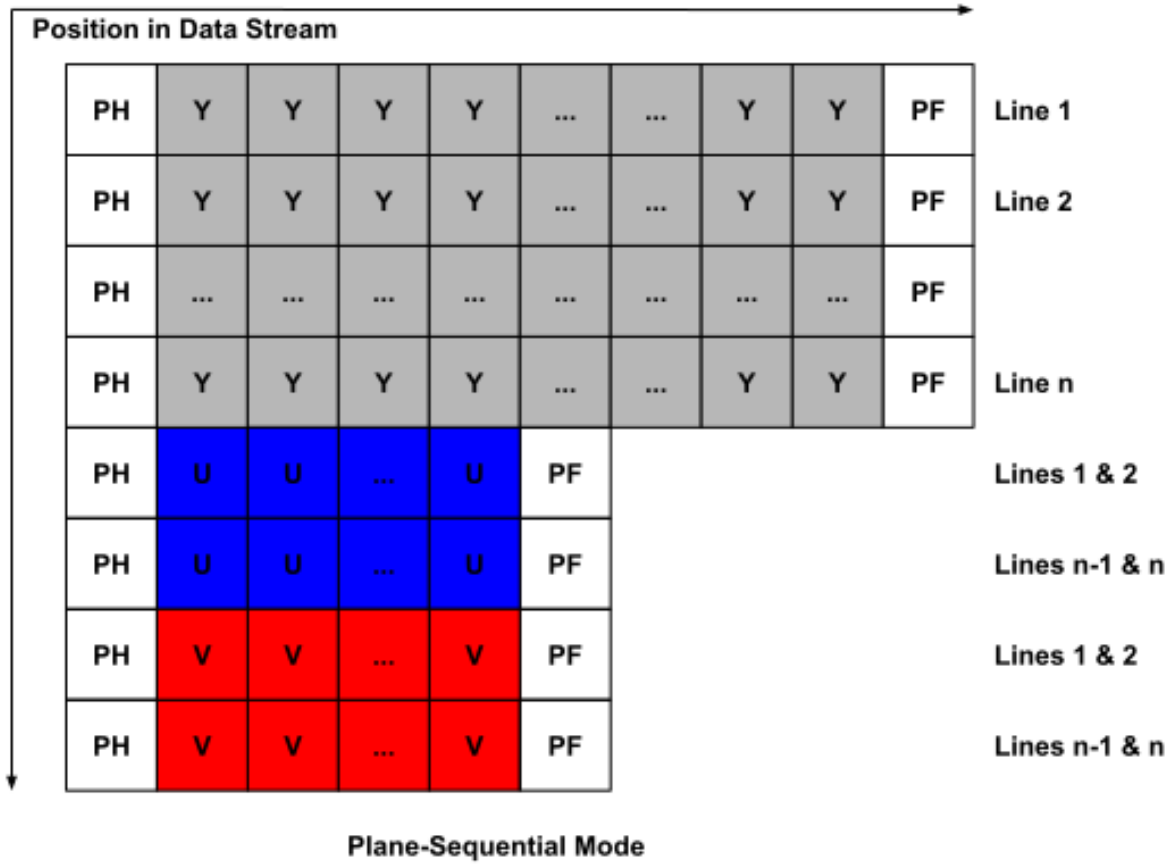


Fig. 10.17: Example of image transmission using plane interleaving mode and YUV4:2:2 planar sampling mode

Chroma components x transmitted on odd line y and even line y+1 are spatially sampled in the middle of the four pixels at locations (x,y), (x+1,y), (x,y+1), (x+1,y+1).

10.2.15 Image Format Identifiers

Image formats are identified by a numeric ID, as reported in table 10.66.

Format	ID	Packing	DT	UV
Reserved shall not be used	0x00			
YUV Formats				
UYVY422_PACKED	0x01	Packed	1	
UYVY420_PACKED	0x02	Packed	1	
UYYVYY420_PACKED	0x03	Packed	1	
YUV422_SEMIPLANAR_LINE_1DT	0x04	Semi Planar	1	UV
YVU422_SEMIPLANAR_LINE_1DT	0x05	Semi Planar	1	VU
YUV422_SEMIPLANAR_LINE_2DT	0x06	Semi Planar	2	UV
YVU422_SEMIPLANAR_LINE_2DT	0x07	Semi Planar	2	VU
YUV422_SEMIPLANAR_PLANE_1DT	0x08	Semi Planar	1	UV
YVU422_SEMIPLANAR_PLANE_1DT	0x09	Semi Planar	1	VU
YUV422_SEMIPLANAR_PLANE_2DT	0x0A	Semi Planar	2	UV
YVU422_SEMIPLANAR_PLANE_2DT	0x0B	Semi Planar	2	VU
YUV422_PLANAR_PLANE_1DT	0x0C	Planar	1	UV
YVU422_PLANAR_PLANE_1DT	0x0D	Planar	1	VU
YUV420_SEMIPLANAR_LINE_1DT	0x0E	Semi Planar	1	UV
YVU420_SEMIPLANAR_LINE_1DT	0x0F	Semi Planar	1	VU
YUV420_SEMIPLANAR_LINE_2DT	0x10	Semi Planar	2	UV
YVU420_SEMIPLANAR_LINE_2DT	0x11	Semi Planar	2	VU
YUV420_SEMIPLANAR_PLANE_1DT	0x12	Semi Planar	1	UV
YVU420_SEMIPLANAR_PLANE_1DT	0x13	Semi Planar	1	VU
YUV420_SEMIPLANAR_PLANE_2DT	0x14	Semi Planar	2	UV
YVU420_SEMIPLANAR_PLANE_2DT	0x15	Semi Planar	2	VU
YUV420_PLANAR_PLANE_1DT	0x16	Planar	1	UV
YVU420_PLANAR_PLANE_1DT	0x17	Planar	1	VU
Binary Formats				
JPEG	0x40			
Metadata	0x41			
Raw Formats				
RAW1 (FIXME)	0x80			

Table 10.66: Camera Device Class Image Format Identifiers

Component Authentication Operation Type	Request Value	Response Value
CPort Shutdown	0x00	0x80
Get Endpoint UID	0x01	0x81
Get IMS Certificate	0x02	0x82
Authenticate	0x03	0x83
(all other values reserved)	0x04..0x7e	0x84..0xfe
Invalid	0x7f	0xff

Table 10.67: Component Authentication Operation Types

10.3 Component Authentication Protocol

The Greybus Component Authentication Protocol may be used by the AP to determine the authenticity of an Interface.

Two forms of authentication are currently defined:

- Ecosystem Authentication, which uses an Ecosystem Authentication Certificate (EAC) and a secret key derived from the Internal Master Secret (IMS) associated with an Interface.
- Identity Authentication, which uses an Identity Authentication Certificate (IAC) and a secret key derived from the IMS associated with an Interface.

Conceptually, the Operations in the Component Authentication Protocol are:

`int cport_shutdown(u8 phase);`
See *Common Greybus Protocol CPort Shutdown Operation*.

`int get_endpoint_uid(u8 endpoint_uid[8]);`
This Operation may be initiated only by the AP to obtain the endpoint unique ID of an Interface. The Response to this Operation contains the endpoint unique ID value.

`int get_ims_certificate(u32 cert_class, u32 cert_id, u8 *result_code, u8 *certificate);`
This Operation may be initiated only by the AP to obtain a certificate from an Interface. The Response to this Operation contains the certificate data.

`int authenticate(u8 auth_type, u8 endpoint_uid[8], u8 challenge[32], u8 *result_code, u8 auth_response[64`
This Operation may be initiated only by the AP to present a challenge to an Interface. The Response to this Operation contains cryptographic response to the challenge and a signature for the cryptographic response.

10.3.1 Component Authentication Operations

The AP Module may use the Component Authentication Protocol to evaluate the authenticity of an Interface. The Request and Response messages for each Component Authentication Operation are defined below.

Table 10.67 describes the Greybus Component Authentication Operation Types and their values.

10.3.2 Greybus Component Authentication CPort Shutdown Operation

The Greybus Component Authentication CPort Shutdown Operation is the *Common Greybus Protocol CPort Shutdown Operation* for the Component Authentication Protocol.

Offset	Field	Size	Value	Description
0	endpoint_uid	8	Byte array	Endpoint Unique ID

Table 10.68: Component Authentication Get Endpoint UID Response

10.3.3 Greybus Component Authentication Get Endpoint UID Operation

The Greybus Component Authentication Get Endpoint UID Operation may be used by the AP to obtain the Endpoint Unique ID (EPUID) associated with an Interface.

The EPUID is a constant eight-byte value guaranteed to be unique across all UniPro endpoints (e.g., Interfaces) in any system components supporting the Greybus Component Authentication Protocol. The EPUID bytes are sent in little-endian format–least significant byte first. The EPUID is derived from a globally unique value known as the IMS, which shall be available to each Interface that supports this Protocol.

The EPUID serves as a key for determining the names of cryptographic certificates used in this Protocol.

Greybus Component Authentication Get Endpoint UID Request

The Greybus Component Authentication Get Endpoint UID Request has no payload.

Greybus Component Authentication Get Endpoint UID Response

The Greybus Component Authentication Get Endpoint UID Response contains an eight-byte field, endpoint_uid.

The endpoint_uid field in the Response payload shall contain the little endian format Endpoint Unique ID value for the Interface.

10.3.4 Greybus Component Get IMS Certificate Operation

The Greybus Component Authentication Get IMS Certificate Operation may be used by the AP to retrieve one of the cryptographic certificates held by an Interface for use in Component Authentication.

Greybus Component Authentication Get IMS Certificate Request

The Greybus Component Authentication Get IMS Certificate Request contains a four-byte field, cert_class and a four-byte field, cert_id. The cert_class field specifies which of the potentially multiple certificates held by an Interface is selected for this Operation, and shall be set to one of the valid values in Table 10.69. The cert_id is the ID of the certificate. It is reserved for future use, and implementations adhering to this version of the protocol shall set its value to zero.

The Greybus Component Authentication Get IMS Certificate Request is sent by the AP to an Interface in order to obtain the data content of a cryptographic certificate of appropriate class.

Greybus Component Authentication Get IMS Certificate Response

The Greybus Component Authentication Get IMS Certificate Response contains a one-byte field, result_code, and an arbitrary-size data block, cert_data, that is the requested certificate. The size of the certificate shall not exceed 1600 bytes.

The result_code field shall identify one of the conditions defined in Table 10.72.

Certificate Class	Description	Value
CERT_IMS_INVALID	Invalid	0x00000000
CERT_IMS_EAPC	Ecosystem Authentication Certificate, Primary Key	0x00000001
CERT_IMS_EASC	Ecosystem Authentication Certificate, Secondary Key	0x00000002
CERT_IMS_EARC	Ecosystem Authentication Certificate, RSA Key	0x00000003
CERT_IMS_IAPC	Identity Authentication Certificate, Primary Key	0x00000004
CERT_IMS_IASC	Identity Authentication Certificate, Secondary Key	0x00000005
CERT_IMS_IARC	Identity Authentication Certificate, RSA Key	0x00000006
	(All other values are reserved)	0x00000007..0xffffffff

Table 10.69: Component Authentication Certificate Classes

Offset	Field	Size	Value	Description
0	cert_class	4	Number	Class of the desired certificate as present in the Table 10.69
4	cert_id	4	Number	ID of the desired certificate

Table 10.70: Component Authentication Get IMS Certificate Size Request

- If the result_code is not CERT_FOUND, the value of cert_data is undefined and shall be ignored.
- If the result_code is CERT_FOUND, the cert_data field shall contain the certificate. AP shall determine the size of the certificate by the size of the Response payload minus the size of the all other fields in the Response payload.

10.3.5 Greybus Component Authentication Authenticate Operation

The Greybus Component Authentication Authenticate Operation may be used by the AP to send a Component Authentication challenge to an Interface and retrieve a Component Authentication response from it.

To authenticate an Interface, the AP shall prepare a Greybus Component Authentication Authenticate Request and send it to the Interface. The receiving Interface shall compute a auth_response, perform a digital signature calculation covering the auth_response, and send both auth_response and signature back to the AP in a Greybus Component Authentication Authenticate Response.

To complete an authentication decision, the AP shall validate the digital signature in the Response using a validation key obtained from an appropriate certificate.

The receiving Interface shall complete its digital signature calculation and return a Response to the AP within an implementation-defined time interval. If the AP does not receive a Response within that time, the AP shall recognize a timeout. The AP may treat timeout as an error, or may repeat the Authenticate Operation.

Offset	Field	Size	Value	Description
0	result_code	1	Number	Result code
1	cert_data	variable data	Byte array	Content of the desired certificate

Table 10.71: Component Authentication Get IMS Certificate Size Response

Result Code	Description	Value
CERT_FOUND	Certificate was located as requested	0x00
CERT_CLASS_INVALID	The specified cert_class is not valid	0x01
CERT_CORRUPT	The storage for certificates is corrupted	0x02
CERT_NOT_FOUND	No certificate of the specified class was found	0x03
	(All other values are reserved)	0x04..0xff

Table 10.72: Component Authentication Certificate Result Codes

Offset	Field	Size	Value	Description
0	auth_type	4	Number	Type of authentication for response
4	endpoint_uid	8	Data	Endpoint Unique ID of target Interface
12	challenge	32	Data	Cryptographic challenge value

Table 10.73: Component Authentication Authenticate Request

Greybus Component Authentication Authenticate Request

The Greybus Component Authentication Authenticate Request contains a four-byte field, auth_type, an eight-byte field, endpoint_uid, and a 32-byte field, challenge. The auth_type field shall be set to one of the valid values in Table 10.74. The endpoint_uid field shall be set to he endpoint_uid of the Interface, which shall have been previously determined by a *Greybus Component Authentication Get Endpoint UID Operation*. For auth_type of AUTH_IMS_PRI, AUTH_IMS_SEC, and AUTH_IMS_RSA, the challenge field shall be set to a 32-byte cryptographically random challenge value.

Several types of authentication are supported, as defined in Table 10.74.

The authentication type in the Request determines the cryptographic algorithm and which class(es) of certificates may be used to validate the Response, as described in table 10.75.

Greybus Component Authentication Authenticate Response

The Greybus Component Authentication Authenticate Response contains a one-byte field, result_code, a 64-byte field, auth_response, and an arbitrary-size data block, auth_response_sig. The size of auth_response_sig shall not exceed 320 bytes.

The result_code field shall identify one of the conditions defined in Table 10.77. If the result_code is not CR_SUCCESS, the values of auth_response and auth_response_sig are undefined and shall be ignored.

The remainder of this section describes processing for auth_type values of AUTH_IMS_PRI, AUTH_IMS_SEC, and AUTH_IMS_RSA.

Upon receiving a Component Authentication Authenticate Request, the Interface shall perform several validation checks (the order of which is unspecified) and calculate a signature. The Interface shall check that:

Type	Description	Value
AUTH_INVALID	Invalid	0x00000000
AUTH_IMS_PRI	Authenticate using the IMS-derived Endpoint Primary Signing Key (EPSK)	0x00000001
AUTH_IMS_SEC	Authenticate using the IMS-derived Endpoint Secondary Signing Key (ESSK)	0x00000002
AUTH_IMS_RSA	Authenticate using the IMS-derived Endpoint RSA Private Key (ERRK)	0x00000003
	(All other values are reserved)	0x00000004..0xffffffff

Table 10.74: Component Authentication Protocol Authentication Types

Auth. Type	Algorithm	Certificate Classes for Authentication
AUTH_IMS_PRI	ed448 *[ED448]*	CERT_IMS_EAPC, CERT_IMS_IASC
AUTH_IMS_SEC	ed25519 *[ED25519]*	CERT_IMS_EASC, CERT_IMS_IASC
AUTH_IMS_RSA	RSA 2048 *[RSA]*	CERT_IMS_EARC, CERT_IMS_IARC

Table 10.75: Component Authentication Types and Certificates

Offset	Field	Size	Value	Description
0	result_code	1	Number	Result code
1	auth_response	64	Byte array	auth_response from module
65	auth_response_sig	variable data	Byte array	Digital signature of auth_response

Table 10.76: Component Authentication Authenticate Response

- The auth_type specifies an authentication type that it is prepared to perform, and shall return a Response with a result_code of CR_BAD_TYPE if not.
- Its own endpoint unique ID matches the endpoint_uid field in the Request, and shall return a Response with a result_code of CR_WRONG_EP if not.
- It has access to the signing key needed to perform the signature calculation, and shall return a Response with a result_code of CR_NO_KEY if not.

Following the validation steps, the Interface shall perform a digital signature calculation using the designated key. If an error occurs performing this calculation, the Interface shall return a Response with a result_code of CR_SIG_FAIL.

The Interface shall calculate the digital signature by preparing a 64-byte response buffer in which the first 32 bytes are a copy of the first 32 bytes of the challenge parameter in the Request, the next 24 bytes are a cryptographically random nonce value calculated by the Interface, the next 8 bytes are the endpoint_uid of the Interface. The Interface shall calculate the digital signature of the 64-byte response buffer using the SHA-256 hash algorithm *[FIPS180]* and the digital signature algorithm identified in Table 10.75.

Having calculated the digital signature, the Interface shall send a Response in which the result_code is CR_SUCCESS, the auth_response is a copy of the response buffer, and the auth_response_sig contains digital signature output.

Upon receipt of a Greybus Component Authentication Authenticate Response, if the result_code is not CR_SUCCESS, the AP shall treat the authentication Operation as having failed. If result_code is CR_SUCCESS, the AP shall perform several validation checks (the order of which is unspecified) The AP shall check that:

- The first 32 bytes of the auth_response field are equal to the the challenge it sent.
- Bytes 56-63 of the auth_response field are equal to the endpoint_uid of the request.

Result	Description	Value
CR_SUCCESS	Authentication response and signature generated successfully	0x00
CR_BAD_TYPE	The specified auth_type is invalid	0x01
CR_WRONG_EP	The supplied endpoint_uid does not match the target Interface	0x02
CR_NO_KEY	The Interface cannot access the required signing key	0x03
CR_SIG_FAIL	The requested signature could not be calculated	0x04
	(All other values are reserved)	0x05..0xff

Table 10.77: Component Authentication Challenge/Response Result Codes

- The size of auth_response_sig, determined by the size of the Response payload minus the size of the all other fields in the Response payload, is non-zero and no greater than 320 bytes.

Having performed the validation checks, the AP shall then locate a certificate containing the validation key for the signature (for example, one obtained from a Greybus Component Authentication Get IMS Certificate Operation, which may occur at any time before the validation calculation, either before or after the Greybus Component Authentication Authenticate Operation). Appropriate certificate(s) may also have been obtained by out of band mechanisms, or found in local storage managed by the AP, depending on system architecture. If the certificate cannot be located or obtained, then the validation fails.

The AP shall then validate that the common name (CN) in the certificate appropriately incorporates the hexadecimal representation of the endpoint_uid value for the Interface and that it otherwise matches the certificate naming conventions (for example, to perform identity authentication, the certificate must also incorporate the hexadecimal representations of the Ara VID and Ara PID attributes of the Interface in an appropriate format). If the certificate name does not meet requirements, then the validation fails.

Finally, the AP shall use the public key from that certificate to attempt to validate that the signature in the signature field is a valid signature of the auth_response field.

If any errors occur in the validation checks, or the signature validation calculation fails, the authentication has failed; otherwise, it has succeeded.

Note that a single Response can be validated with respect to multiple different certificates, depending on goal of the authentication (e.g., ecosystem authentication, identity authentication). The different certificates will contain the same (public) validation key but will be distinguished by the Common Name in the certificate.

10.4 Firmware Download Protocol

The Greybus Firmware Download Protocol can be used by an Interface to communicate with the AP and receive firmware packages over UniPro.

If an Interface requires to download a firmware package, it shall first request the AP to find a firmware package for the Interface using a *Greybus Firmware Download Find Firmware Operation*. This may be followed by one or more *Greybus Firmware Download Protocol Fetch Firmware Operations* to receive the firmware package block by block. Finally the Interface shall request the AP to release the firmware package using a *Greybus Firmware Download Release Firmware Operation*.

Conceptually, the Operations in the Greybus Firmware Download Protocol are:

`int cport_shutdown(u8 phase);`
: See *Common Greybus Protocol CPort Shutdown Operation*.

`int find_firmware(u8 firmware_tag[10], u8 format[10], u8 *firmware_id, u32 *size);`
: This Operation can be initiated only by an Interface to request the AP to find a firmware package for the Interface.

`int fetch_firmware(u8 firmware_id, u32 offset, u32 size, void *data);`
: This Operation can be initiated only by an Interface to fetch a block of data from the AP in the firmware package previously requested from the AP.

`int release_firmware(u8 firmware_id);`
: If the Interface has requested the AP to find a firmware package using a *Greybus Firmware Download Find Firmware Operation* earlier, it shall use this Operation to request the AP to release that firmware package.

Firmware Operation Type	Request Value	Response Value
CPort Shutdown	0x00	0x80
Find Firmware	0x01	0x81
Fetch Firmware	0x02	0x82
Release Firmware	0x03	0x83
(all other values reserved)	0x04..0x7e	0x84..0xfe
Invalid	0x7f	0xff

Table 10.78: Firmware Download Protocol Operation Types

10.4.1 Greybus Firmware Download Operations

All Firmware Download Protocol Operations are initiated using a Greybus Firmware Download Protocol Request message, which results in a matching Response message. The Request and Response messages for each Operation are defined below.

Table 10.78 defines the Greybus Firmware Download Protocol Operation types and their values. Both the Request type and the Response type values are shown below.

10.4.2 Greybus Firmware Download CPort Shutdown Operation

The Greybus Firmware Download CPort Shutdown Operation is the *Common Greybus Protocol CPort Shutdown Operation* for the Firmware Download Protocol.

10.4.3 Greybus Firmware Download Find Firmware Operation

The Greybus Firmware Download Find Firmware Operation Request can be sent only by an Interface to request the AP to find a firmware package for the Interface.

The Interface provides a firmware_tag and its format to the AP as part of the request, which may be used by the AP in an implementation-defined way to find the firmware package for the Interface.

In response, the AP locates a matching firmware package and returns to the Interface the size of the firmware package and a unique firmware_id associated with the firmware package.

The same firmware_id shall be sent by the Interface as part of the Fetch Firmware or the Release Firmware Requests sent later.

This may be followed by one or more *Greybus Firmware Download Fetch Firmware Operation Requests* from the Interface to the AP, in order to receive the firmware package block by block.

Once the firmware is successfully requested by the Interface using a *Greybus Firmware Download Find Firmware Operation*, the AP shall support all valid *Greybus Firmware Download Fetch Firmware Operation Requests* until the Interface initiates a *Greybus Firmware Download Release Firmware Operation* or the AP times out waiting for a request from the Interface.

An Interface may request the AP to find one or more firmware packages using separate *Greybus Firmware Download Find Firmware Operations* and fetch them in parallel by using the firmware_id received from the AP earlier in the Find Firmware Response.

The AP may impose implementation-defined timeouts for:

- The time interval between the Find Firmware Response and the first Fetch Firmware Request.
- The time interval between a Fetch Firmware Response and the next Fetch Firmware Request.

Offset	Field	Size	Value	Description
0	firmware_tag	10	*[US-ASCII]*	A null-terminated character string used to identify the firmware package.
10	format	10	*[US-ASCII]*	A null-terminated character string used to identify the format of firmware package.

Table 10.79: Firmware Download Find Firmware Request

Offset	Field	Size	Value	Description
0	firmware_id	1	Number	Unique firmware package identifier.
1	size	4	Number	Size of the firmware package in bytes.

Table 10.80: Firmware Download Find Firmware Response

- The time interval between a Fetch Firmware Response and the Release Firmware Request.
- The time interval between the Find Firmware Response and the Release Firmware Request.

If any of the above timeouts occur, the AP shall respond with GB_OP_TIMEOUT in the status byte of the Response header, to the next Request from the Interface that uses the same firmware_id for the which the AP has timed out.

Greybus Firmware Download Find Firmware Request

Table 10.79 defines the Greybus Firmware Download Find Firmware Request payload. The Request contains a 10-byte firmware_tag and a 10-byte format of the firmware package requested for download. This may be used by the AP in an implementation-defined way to find the requested firmware package.

Greybus Firmware Download Find Firmware Response

Table 10.80 defines the Greybus Firmware Download Find Firmware Response payload. The Response contains a one-byte firmware_id and a four-byte size of the firmware package in bytes.

The firmware_id is unique and the same firmware_id shall not be used by the AP in another *Greybus Firmware Download Find Firmware Operation* Request, until the Interface has initiated the *Greybus Firmware Download Release Firmware Operation* with the same firmware_id.

If the AP fails to find a firmware package for the Interface, it shall return GB_OP_INVALID in the status byte of the Response header.

10.4.4 Greybus Firmware Download Fetch Firmware Operation

The Greybus Firmware Download Fetch Firmware Operation Request can be sent only by an Interface to request the AP to provide a block of data, from the firmware package the Interface has previously requested from the AP.

The Interface sends to the AP the firmware_id of the firmware package, received as part of the Find Firmware Response earlier, the offset within the firmware package, and the size in bytes of the block of data to fetch from the offset.

Unless the AP finds the Request to be invalid or if the AP hasn't timed out waiting for a Fetch Firmware Request, it shall respond with exactly the number of bytes requested by the Interface, from the firmware package associated with the firmware_id.

The AP may consider a Request as invalid if:

Offset	Field	Size	Value	Description
0	firmware_id	1	Number	Unique firmware package identifier.
1	offset	4	Number	Offset into the firmware package.
5	size	4	Number	Size of block of data in bytes.

Table 10.81: Firmware Download Fetch Firmware Request

Offset	Field	Size	Value	Description
0	data	*size*	Data	Block of data within the firmware package.

Table 10.82: Firmware Download Fetch Firmware Response

- The AP couldn't associate the firmware_id sent by the Interface to an already requested firmware package.
- The Interface tries to read past the end of the firmware package.
- Size field in the Request is set to 0.

The Interface may send one or more Fetch Firmware Requests to receive the firmware package. The access to the firmware package isn't required to be sequential and the Interface may download the firmware package in any order. The Interface may download a section of the firmware package multiple times.

Greybus Firmware Download Fetch Firmware Request

Table 10.81 defines the Greybus Firmware Download Fetch Firmware Request payload. The Request contains a one-byte firmware_id associated with the firmware package, a four-byte offset within the firmware package, and a four-byte size of the block of data requested in bytes.

The requested size must be less than or equal to the firmware size received with the Find Firmware Response, minus the requested offset into the firmware package.

The Interface is responsible for tracking its offset into the firmware package as needed.

Greybus Firmware Download Fetch Firmware Response

Table 10.82 defines the Greybus Firmware Download Fetch Firmware Response payload. The Response contains the block of data requested by the Interface.

The AP may return GB_OP_INVALID in the status byte of the Response header, if the AP finds the Request sent by an Interface as invalid, as described in the *Greybus Firmware Download Fetch Firmware Operation* section.

Upon receiving a Response with status equal to GB_OP_INVALID, the Interface may resend this Request after verifying its parameters.

The AP may return GB_OP_TIMEOUT in the status byte of the Response header, if the AP has timed out waiting for the Fetch Firmware Request.

If this occurs, the firmware_id is no longer valid. Upon receiving a Response with status equal to GB_OP_TIMEOUT, the Interface shall not send additional Fetch Firmware Requests with the same firmware_id, unless a subsequent *Greybus Firmware Download Find Firmware Operation* Response includes that firmware_id. The Interface may initiate another *Greybus Firmware Download Find Firmware Operation* with the same firmware_tag in order to attempt to subsequently recover from the timeout and retrieve the same firmware package.

Offset	Field	Size	Value	Description
0	firmware_id	1	Number	Unique firmware package identifier.

Table 10.83: Firmware Download Release Firmware Request

10.4.5 Greybus Firmware Download Release Firmware Operation

The Greybus Firmware Download Release Firmware Operation Request can be sent only by an Interface to request the AP to release a firmware package it has requested earlier.

The Interface sends to the AP the firmware_id associated with the firmware package, provided earlier by the AP in the response to the *Greybus Firmware Download Find Firmware Operation*.

Greybus Firmware Download Release Firmware Request

Table 10.83 defines the Greybus Firmware Download Release Firmware Request payload. The Request contains a one-byte firmware_id associated with the firmware package to be released.

Greybus Firmware Download Release Firmware Response

The Greybus Firmware Download Release Firmware Response has no payload.

If the AP couldn't associate the firmware_id sent by the Interface to a firmware package, then the AP shall return GB_OP_INVALID in the status byte of the Response header.

If the AP has timed out waiting for the Release Firmware Request, it shall return GB_OP_TIMEOUT in the status byte of the Response header.

On any such errors, the Interface shall do nothing as the firmware package shall already have been released by the AP.

10.5 Firmware Management Protocol

The Firmware Management Protocol can be used by the Application Processor (AP) to communicate with an Interface to:

- Load and Validate an *Interface Firmware* package for an Interface.
- Prepare the Interface to enter the *MODE_SWITCHING Interface Lifecycle State*.
- Update *Interface Backend Firmware* packages on an Interface.

The *Interface Firmware* that requires the capability to enter the *MODE_SWITCHING Interface Lifecycle State*, may provide a CPort that implements the Firmware Management Protocol.

In order to use the Firmware Management Protocol for an Interface, the Interface *Manifest* received by the AP from the Interface over the *Control Protocol* shall contain a *Bundle Descriptor* with the Class Type Firmware-Management. This Bundle shall contain one *CPort Descriptor* with the Protocol Type Firmware-Management.

The Firmware Management Protocol shall not be used by the AP, if its *CPort Descriptor* isn't part of the *Bundle Descriptor* with the Class Type Firmware-Management.

The Firmware-Management Bundle may contain another *CPort Descriptor* with the Protocol Type SPI, if the Interface contains a local SPI flash and the Interface Firmware running on the Interface is designed to

allow the AP to manage updates to the SPI flash. The AP shall communicate over this SPI CPort using the *SPI Protocol*.

The Firmware-Management Bundle may contain another *CPort Descriptor* with the Protocol Type Firmware-Download. The Interface Firmware may use this CPort to receive firmware packages from the AP using the *Firmware Download Protocol*.

The Firmware-Management Bundle may contain another *CPort Descriptor* with the Protocol Type Component Authentication Protocol (CAP). The AP may use this CPort to Authenticate the Interface.

Todo

Add Component Authentication Protocol (CAP) to Greybus Specifications.

The rest of this section defines the Firmware Management Protocol.

Conceptually, the Operations of the Greybus Firmware Management Protocol are:

`int cport_shutdown(u8 phase);`
See *Common Greybus Protocol CPort Shutdown Operation*.

Note: Below Operations are specific to the *Interface Firmware* for an Interface.

`int interface_firmware_version(u8 firmware_tag[10], u16 *major, u16 *minor);`
This Operation can be initiated only by the AP to get the firmware_tag and the version of the Interface Firmware currently running on an Interface.

`int interface_firmware_load_and_validate(u8 request_id, u8 load_method, u8 firmware_tag[10]);`
This Operation can be initiated only by the AP to instruct an Interface to load and validate an Interface Firmware package.

`int interface_firmware_loaded(u8 request_id, u8 status, u16 major, u16 minor);`
If the AP has requested an Interface to load an Interface Firmware using the *Greybus Firmware Management Interface Firmware Load and Validate Operation* earlier, then the Interface shall use this Operation to inform the AP once the requested Interface Firmware package is loaded and validated by the Interface.

Note: Below Operations are specific to the *Interface Backend Firmware* for an Interface.

`int interface_backend_firmware_version(u16 *major, u16 *minor, u8 *status);`
This Operation can be initiated only by the AP to get the current version of the Interface Backend Firmware packages available locally with an Interface.

`int interface_backend_firmware_update(u8 request_id);`
This Operation can be initiated only by the AP to request an Interface to update the Interface Backend Firmware packages.

`int interface_backend_firmware_updated(u8 request_id, u8 status);`
If the AP has requested an Interface to update an Interface Backend Firmware using the *Greybus Firmware Management Interface Backend Firmware Update Operation* earlier, then the Interface shall use this Operation to inform the AP once the update to the Interface Backend Firmware has finished.

Firmware Management Operation Type	Request Value	Response Value
CPort Shutdown	0x00	0x80
Interface Firmware Version	0x01	0x81
Interface Firmware Load and Validate	0x02	0x82
Interface Firmware Loaded	0x03	0x83
Interface Backend Firmware Version	0x04	0x84
Interface Backend Firmware Update	0x05	0x85
Interface Backend Firmware Updated	0x06	0x86
(all other values reserved)	0x07..0x7e	0x87..0xfe
Invalid	0x7f	0xff

Table 10.84: Firmware Management Protocol Operation Types

Offset	Field	Size	Value	Description
0	firmware_tag	10	*[US-ASCII]*	A null-terminated character string used to identify the Interface Firmware.
10	major	2	Number	Major version number of the currently running Interface Firmware.
12	minor	2	Number	Minor version number of the currently running Interface Firmware.

Table 10.85: Firmware Management Interface Firmware Version Response

10.5.1 Greybus Firmware Management Protocol Operations

All Firmware Management Protocol Operations are initiated using a Greybus Firmware Management Protocol Request message, which results in a matching Response message. The Request and Response messages for each Operation are defined below.

Table 10.84 defines the Greybus Firmware Management Protocol Operation types and their values. Both the Request type and the Response type values are shown below.

10.5.2 Greybus Firmware Management CPort Shutdown Operation

The Greybus Firmware Management CPort Shutdown Operation is the *Common Greybus Protocol CPort Shutdown Operation* for the Firmware Management Protocol.

10.5.3 Greybus Firmware Management Interface Firmware Version Operation

The Greybus Firmware Management Interface Firmware Version Operation Request can be sent only by the AP to an Interface. The Interface shall respond with the firmware_tag, and the version of the Interface Firmware currently running on the Interface.

Greybus Firmware Management Interface Firmware Version Request

The Greybus Firmware Management Interface Firmware Version Request has no payload.

Greybus Firmware Management Interface Firmware Version Response

Table 10.85 defines the Greybus Firmware Management Interface Firmware Version Response payload. The Response contains a 10-byte firmware_tag, and two 2-byte version numbers, major and minor. The firmware_tag may be used by the AP in an implementation-defined way to identify the currently running Interface Firmware.

10.5.4 Greybus Firmware Management Interface Firmware Load and Validate Operation

The Greybus Firmware Management Interface Firmware Load and Validate Operation Request can be sent only by the AP to an Interface.

On receiving this Request, the Interface shall respond immediately and start loading the requested Interface Firmware package using the specified load_method and then validate it using implementation-defined means. Once the Interface has loaded and validated the Interface Firmware package or if the Interface failed to load or validate the Interface Firmware package, it shall initiate a *Greybus Firmware Management Interface Firmware Loaded Operation*.

The Interface shall load at most one Interface Firmware package at a time. A Request to load a new Interface Firmware package may replace the Interface Firmware package loaded earlier.

The process of validating an Interface Firmware package is implementation-defined.

The AP sends a unique request_id to the Interface and the Interface shall use the same request_id while sending the *Greybus Firmware Management Interface Firmware Loaded Operation* Request.

The AP may wait for an implementation-defined time interval, for the Interface to initiate a *Greybus Firmware Management Interface Firmware Loaded Operation*. If the AP times out waiting for it, the AP may re-initiate this Operation with a new request_id.

If an Interface receives another Interface Firmware Load and Validate Request with a different request_id, before it has initiated a *Greybus Firmware Management Interface Firmware Loaded Operation* for the earlier Load and Validate Firmware Request, then the Interface shall abort the previous Load and Validate Firmware Request and start servicing the new Request.

The AP may initiate this Operation any number of times.

If the AP is using the *Firmware Download Protocol* to prepare an Interface to enter the *MODE_SWITCHING Interface Lifecycle State*, then the AP shall initiate the *Greybus Control Mode Switch Operation* only after it has received a successful *Greybus Firmware Management Interface Firmware Loaded Operation* Request from the Interface.

Greybus Firmware Management Interface Firmware Load and Validate Request

The Greybus Firmware Management Interface Firmware Load and Validate Request contains a one-byte request_id, a one-byte load_method, which identifies the method to be used to load the Interface Firmware, and a 10-byte firmware_tag of the Interface Firmware that is requested to be loaded. The firmware_tag may be used by the Interface in an implementation-defined way to identify the requested Interface Firmware package.

The request_id is unique and the same request_id shall not be used by the AP in another *Greybus Firmware Management Interface Firmware Load and Validate Operation* Request until the Interface has initiated a *Greybus Firmware Management Interface Firmware Loaded Operation* with the same request_id.

If the load_method specified in the Request is set to FIRMWARE_LOAD_METHOD_UNIPRO, then the Interface shall receive the Interface Firmware package using the *Firmware Download Protocol* and send the same firmware_tag value received from the AP to the *Greybus Firmware Download Find Firmware Operation* Request.

If load_method specified in the Request from the AP is set to FIRMWARE_LOAD_METHOD_INTERNAL, then the Interface shall load the Interface Firmware package available locally with the Interface, in an implementation-defined way.

Offset	Field	Size	Value	Description
0	request_id	1	Number	Unique Request Identifier.
1	load_method	1	Number	Possible values of load_method are specified in table 10.87.
2	firmware_tag	10	*[US-ASCII]*	A null-terminated character string used to identify the Interface Firmware.

Table 10.86: Firmware Management Interface Firmware Load and Validate Request

Interface Firmware Load Method	Brief Description	Value
FIRMWARE_LOAD_METHOD_INVALID	Invalid	0x00
FIRMWARE_LOAD_METHOD_UNIPRO	Load Interface Firmware package over UniPro.	0x01
FIRMWARE_LOAD_METHOD_INTERNAL	Load Interface Firmware package internally available to the Interface.	0x02
	(Reserved Range)	0x03..0xFF

Table 10.87: Firmware Management Interface Firmware Load Method

Greybus Firmware Management Interface Firmware Load and Validate Response

The Greybus Firmware Management Interface Firmware Load and Validate Response has no payload.

10.5.5 Greybus Firmware Management Interface Firmware Loaded Operation

The Greybus Firmware Management Interface Firmware Loaded Operation Request can be sent only by an Interface to indicate to the AP that an earlier *Interface Firmware Load and Validate Operation Request* from the AP has finished.

On receiving this Request, the AP may check the status byte from the Request and compare the version of the loaded Interface Firmware with the Interface Firmware packages available with the AP. The AP may subsequently choose to initiate another *Greybus Firmware Management Interface Firmware Load and Validate Operation*, to load a new Interface Firmware package.

If the AP is using the *Firmware Download Protocol* to prepare an Interface to enter the *MODE_SWITCHING Interface Lifecycle State*, then the AP shall initiate the *Greybus Control Mode Switch Operation* only after it has received a successful *Greybus Firmware Management Interface Firmware Loaded Operation* Request from the Interface.

Greybus Firmware Management Interface Firmware Loaded Request

The Greybus Firmware Management Interface Firmware Loaded Request contains a one-byte request_id, a one-byte status of the loaded Interface Firmware package, a two-byte major version, a two-byte minor version.

The value of the request_id field shall be set to the value of the request_id field sent by the AP in the *Greybus Firmware Management Interface Firmware Load and Validate Operation* Request, in response to which the Interface is sending this Request.

If the AP has initiated another *Greybus Firmware Management Interface Firmware Load and Validate Operation* before receiving a *Greybus Firmware Management Interface Firmware Loaded Operation* Response from the Interface for the previous *Greybus Firmware Management Interface Firmware Load and Validate Operation* Request, then the AP shall ignore the Interface Firmware Loaded Request with the request_id matching the request_id of the first *Greybus Firmware Management Interface Firmware Load and Validate Operation* Request.

Offset	Field	Size	Value	Description
0	request_id	1	Number	Unique Request Identifier.
1	status	1	Number	Status of the Interface Firmware loading and validation is defined by the table 10.89 and is set by the Interface in an implementation-defined way.
2	major	2	Number	Major version number of the loaded Interface Firmware package.
4	minor	2	Number	Minor version number of the loaded Interface Firmware package.

Table 10.88: Firmware Management Interface Firmware Loaded Response

Interface Firmware Status	Brief Description	Status Value
FW_STATUS_LOAD_FAILED	Failed to Load the Interface Firmware package.	0x00
FW_STATUS_UNVALIDATED	Loaded Interface Firmware Package is not signed.	0x01
FW_STATUS_VALIDATED	Loaded Interface Firmware Package is signed and is validated by the Interface.	0x02
FW_STATUS_VALIDATION_FAILED	Loaded Interface Firmware Package is signed and the Interface failed to validate it.	0x03
	(Reserved Range)	0x04..0xFF

Table 10.89: Firmware Management Interface Firmware Loaded Status

Greybus Firmware Management Interface Firmware Loaded Response

The Greybus Firmware Management Interface Firmware Loaded Response has no payload.

10.5.6 Greybus Firmware Management Interface Backend Firmware Version Operation

The Greybus Firmware Management Interface Backend Firmware Version Operation Request can be sent only by the AP to an Interface, to request the version of the Interface Backend Firmware Packages available locally with the Interface. The same version shall apply to all the Backend Firmware Packages.

Greybus Firmware Management Interface Backend Firmware Version Request

The Greybus Firmware Management Interface Backend Firmware Version Request has no payload.

Greybus Firmware Management Interface Backend Firmware Version Response

Table 10.90 defines the Greybus Firmware Management Interface Backend Firmware Version Response payload. The Response contains two 2-byte numbers, major and minor, and a 1-byte status.

The major and minor numbers shall be ignored by the AP if the status contains value other than FW_STATUS_SUCCESS.

If the Interface doesn't require any Interface Backend Firmware package for its functioning, then the Interface shall set the status to FW_STATUS_NOT_SUPPORTED.

If the Interface doesn't have all Interface Backend Firmware package available with it, then it shall set the status to FW_STATUS_NOT_AVAILABLE.

Otherwise, the Interface shall set both major and minor fields in its Response with the major and minor version of its Interface Backend Firmware packages.

The Interface may require some time before providing the version of the Interface Backend Firmware packages. This may happen, for example, if the Interface needs to boot the Backend Device Processors before getting the version of the available Interface Backend Firmware. On such an event, the Interface shall set the status to FW_STATUS_RETRY.

On receiving FW_STATUS_RETRY from the Interface, the AP may re-initiate this Operation after an implementation-defined time interval. The AP may keep sending this Request until the time it receives the Interface Backend Firmware version, or the Request fails and returns some other error value.

Offset	Field	Size	Value	Description
0	major	2	Number	Major version number of the Interface Backend Firmware packages.
2	minor	2	Number	Minor version number of the Interface Backend Firmware packages.
4	status	1	Number	Status of the Interface Backend Firmware version operation is defined by the table 10.91.

Table 10.90: Firmware Management Interface Backend Firmware Version Response

Update Status	Brief Description	Value
FW_STATUS_INVALID	Invalid Status.	0x00
FW_STATUS_SUCCESS	Firmware version successfully retrieved.	0x01
FW_STATUS_NOT_AVAILABLE	Firmware not available.	0x02
FW_STATUS_NOT_SUPPORTED	No Backend Firmware is required for functioning of Interface.	0x03
FW_STATUS_RETRY	Not ready to respond currently, retry.	0x04
FW_STATUS_FAIL_INTERNAL	Failed due to internal errors.	0x05
	(Reserved Range)	0x06..0xFF

Table 10.91: Firmware Interface Backend Firmware Version Status

10.5.7 Greybus Firmware Management Interface Backend Firmware Update Operation

The Greybus Firmware Management Interface Backend Firmware Update Operation Request can be sent only by the AP to request an Interface, to update the Interface Backend Firmware packages.

The Interface shall update all the Interface Backend Firmware packages.

If the Interface can not service the Interface Backend Firmware Update Request or if the Interface doesn't require any Interface Backend Firmware for its functioning, then it shall send GB_OP_INVALID in the status field of the Response header.

Otherwise, the Interface shall immediately respond to this Request and start downloading the Interface Backend Firmware packages from the AP, in any order it finds suitable.

If the Interface is designed to use the *Firmware Download Protocol* for downloading firmware packages, then it shall contain a *CPort Descriptor* with the Protocol Type Firmware-Download in its *Bundle Descriptor* whose Class Type is Firmware-Management, in the Interface *Manifest* sent to the AP.

The rest of this section uses the *Firmware Download Protocol* as the Interface Backend Firmware download method. The Interface may choose another implementation-defined method for receiving the Interface Backend Firmware packages.

Once the specific Interface Backend Firmware package is updated on the Interface, the Interface shall initiate a *Greybus Firmware Management Interface Backend Firmware Updated Operation*.

The AP sends a unique request_id to the Interface and the Interface shall use the same request_id while sending the *Greybus Firmware Management Interface Backend Firmware Updated Operation* Request.

The same request_id shall not be used by the AP in another *Greybus Firmware Management Interface Backend Firmware Update Operation* Request until the Interface has initiated a *Greybus Firmware Management Interface Backend Firmware Updated Operation* with the same request_id.

The AP may wait for an implementation-defined time interval, for the Interface to initiate a *Greybus Firmware Management Interface Backend Firmware Updated Operation*. In case the AP times out waiting for it, the AP may re-initiate this Operation with a different request_id.

If the Interface receives another Interface Backend Firmware Update Request before it has initiated a *Greybus Firmware Management Interface Backend Firmware Updated Operation* for the earlier Interface Backend Firmware Update Request, the Interface shall abort the previous Interface Backend Firmware Update Request and start servicing the new Request.

Offset	Field	Size	Value	Description
0	request_id	1	Number	Unique Request Identifier.

Table 10.92: Firmware Management Interface Backend Firmware Update Request

The Module can download Interface Backend Firmware packages in parallel on receiving this request.

Greybus Firmware Management Interface Backend Firmware Update Request

Table 10.92 defines the Greybus Firmware Management Interface Backend Firmware Update Request payload. The Request contains a one-byte request_id.

The request_id is unique and the same request_id shall not be used by the AP in another *Greybus Firmware Management Interface Backend Firmware Update Operation* Request until the Interface has initiated a *Greybus Firmware Management Interface Backend Firmware Updated Operation* with the same request_id.

Greybus Firmware Management Interface Backend Firmware Update Response

The Greybus Firmware Management Interface Backend Firmware Update Response has no payload.

10.5.8 Greybus Firmware Management Interface Backend Firmware Updated Operation

The Greybus Firmware Management Interface Backend Firmware Updated Operation Request can be send only by an Interface to inform the AP that the Interface Backend Firmware update to a specific Interface Backend Firmware package has finished. This shall be sent by the Interface after it has downloaded the requested Interface Backend Firmware package using the *Firmware Download Protocol* and updated it internally in an implementation-defined way.

The Interface shall also initiate this Operation if it has failed to update the requested Interface Backend Firmware package. It shall specify the reason of the failure in the status field of the Request.

The AP may initiate another *Greybus Firmware Management Interface Backend Firmware Update Operation* now.

Greybus Firmware Management Interface Backend Firmware Updated Request

Table 10.93 defines the Greybus Firmware Management Interface Backend Firmware Updated Request payload. The Request contains a one-byte request_id, and a one-byte status of the Firmware update.

The value of the request_id field shall be set to the value of the request_id field sent by the AP in the *Greybus Firmware Management Interface Backend Firmware Update Operation* Request, in response to which the Interface is sending this Request.

If the AP initiates another *Greybus Firmware Management Interface Backend Firmware Update Operation* before receiving a *Greybus Firmware Management Interface Backend Firmware Updated Operation* Request from the Interface for the previous *Greybus Firmware Management Interface Backend Firmware Update Operation* Request, then the AP shall ignore the Interface Backend Firmware Updated Request with the request_id matching the request_id of the first *Greybus Firmware Management Interface Backend Firmware Update Operation* Request.

Offset	Field	Size	Value	Description
0	request_id	1	Number	Unique Request Identifier.
1	status	1	Number	Status of the Interface Backend Firmware update is defined by the table 10.94 and is set by the Interface in an implementation-defined way.

Table 10.93: Firmware Management Interface Backend Firmware Updated Request

Update Status	Brief Description	Value
FW_STATUS_INVALID	Invalid Status.	0x00
FW_STATUS_SUCCESS	Interface Backend Firmware package successfully updated.	0x01
FW_STATUS_FAIL_FIND	Failed to find Interface Backend Firmware package.	0x02
FW_STATUS_FAIL_FETCH	Failed to fetch Interface Backend Firmware package.	0x03
FW_STATUS_FAIL_WRITE	Failed to write downloaded Interface Backend Firmware package.	0x04
FW_STATUS_FAIL_INTERNAL	Failed due to internal errors.	0x05
FW_STATUS_RETRY	Not ready to respond currently, retry.	0x06
FW_STATUS_NOT_SUPPORTED	No Backend Firmware is required for functioning of Interface.	0x07
	(Reserved Range)	0x08..0xFF

Table 10.94: Firmware Interface Backend Firmware Update Status

Greybus Firmware Management Interface Backend Firmware Updated Response

The Greybus Firmware Interface Backend Firmware Updated Response has no payload.

10.6 HID Protocol

This section defines the operations used on a connection implementing the Greybus Human Interface Device (HID) Protocol. The HID class is used primarily for devices that take input from humans or may give output to humans. Typical examples of HID class devices include:

- Keyboards and pointing devices
- Front panel controls, like: knobs, buttons, switches, etc.
- Steering wheels, rudder pedals found on gaming devices.
- Buttons, touchscreen found on phones.
- Bar-code readers, thermometers, or voltmeters.

The Greybus HID Protocol uses *descriptors* and *reports* to interact with a HID device. A HID Descriptor defines all capabilities of a HID device. Before exchanging data with a HID device, the AP Module can configure a HID device based on these capabilities by sending Feature Reports. Data exchange between the AP Module and a HID device are implemented by sending Input or Output Reports.

This document focuses on how the HID protocol is implemented over Greybus. The HID Protocol (as implemented over USB) is well defined by *[HID01]*.

10.6.1 Greybus HID Descriptors

The following section identifies the key data structures (referred to as HID Descriptors) that need to be exchanged between the host and the device during initialization.

HID Descriptor

The HID Descriptor is the top-level mandatory descriptor that every Greybus based HID device must have. The purpose of the HID Descriptor is to define all capabilities of the HID device with the host. These attributes describe the version of the HID Protocol the HID device is compliant with, the length of HID Descriptors, and other capabilities of the device. Please refer to Table 10.96 for further details.

HID Report Descriptor

A HID Report Descriptor describes the data generated by the HID device, and how to interpret that data. Details of the HID Report Descriptor are outside of the scope of this document and are defined in *[HID01]*.

10.6.2 HID Report Protocol

The Report is the fundamental block exchanged between the host and the device. Reports are well defined by *[HID01]*, and the the same will be followed here.

HID Input Report

The input reports are generated on the device and are sent from device to host. This can be requested synchronously or asynchronously.

In the asynchronous case, when the device has active data it wishes to report to the host, it will generate an data request towards the host. When the host receives the receive request, it is responsible for reading the data from the receive request.

In the synchronous case, the host can generate a get-report request to HID device, in response to which the device must respond with data.

HID Output Report

The output report is generated on the host and is sent from host to device over the Greybus transport. When the host has active data it wishes to report to the device, it must generate a set-report request.

HID Feature Report

The feature report is a bidirectional report and can be exchanged between the host and the device. They are normally used by the host to program the device into different configurations.

For the host to get/set a feature-report on the device, it must use the get-report and set-report requests described later.

10.6.3 Greybus HID Operations

Greybus HID Protocol allows an AP to manage a HID device present on a module. The Protocol consists of few basic operations, whose request and response message formats are defined here.

Conceptually, the operations in the greybus HID Protocol are:

`int cport_shutdown(u8 phase);`
See *Common Greybus Protocol CPort Shutdown Operation*.

HID Operation Type	Request Value	Response Value
CPort Shutdown	0x00	0x80
Reserved	0x01	0x81
Get Descriptor	0x02	0x82
Get Report Descriptor	0x03	0x83
Power On	0x04	0x84
Power Off	0x05	0x85
Get Report	0x06	0x86
Set Report	0x07	0x87
IRQ Event	0x08	0x88
(all other values reserved)	0x09..0x7e	0x89..0xfe
Invalid	0x7f	0xff

Table 10.95: HID Operation Types

`int get_descriptor(struct gb_hid_desc_response *desc);`
Returns *HID Descriptor*, that specifies details of the HID device.

`int get_report_descriptor(u8 *report_desc);`
Returns a HID Report Descriptor, defined by *[HID01]*.

`int power_on(void);`
Power-on the HID device.

`int power_off(void);`
Power-off the HID device.

`int get_report(u8 *report);`
Gets input or feature report from device to host synchronously.

`int set_report(u8 *report);`
Sets output or feature report from host to device synchronously.

`int irq_event(u8 *report);`
Input report sent from device to host asynchronously.

10.6.4 Greybus HID Message Types

Table 10.95 describes the Greybus HID operation types and their values. A message type consists of an operation type combined with a flag (0x80) indicating whether the operation is a request or a response.

10.6.5 Greybus HID CPort Shutdown Operation

The Greybus HID CPort Shutdown Operation is the *Common Greybus Protocol CPort Shutdown Operation* for the HID Protocol.

10.6.6 Greybus HID Get Descriptor Operation

The Greybus HID Get Descriptor operation is issued on the host and the HID device must respond with an *HID Descriptor*.

Offset	Field	Size	Value	Description
0	length	1	Number	Length of this descriptor
1	report_desc_length	2	Number	Length of the report descriptor
3	hid_version	2	Number	Version of the HID Protocol, as defined by *[HID01]*
5	product_id	2	Number	Product ID of the device
7	vendor_id	2	Number	Vendor ID of the device
9	country_code	1	Number	Country code of the localized hardware; see *[HID01]*

Table 10.96: Greybus HID Descriptor

Greybus HID Get Descriptor Request

The Greybus HID Get Descriptor request is sent from host to device and it has no payload.

Greybus HID Get Descriptor Response

The Greybus HID Get Descriptor response is sent from device to host and is described in Table 10.96.

10.6.7 Greybus HID Get Report Descriptor Operation

The Greybus HID Get Report Descriptor operation is issued on host and the HID device must respond with an report descriptor as defined by *[HID01]*.

Greybus HID Get Report Descriptor Request

The Greybus HID Get Report Descriptor request is sent from host to device and the request has no payload.

Greybus HID Get Report Descriptor Response

The Greybus HID Get Report Descriptor response is sent from device to host and it consists of a HID Report Descriptor defined by *[HID01]*.

10.6.8 Greybus HID Power ON Operation

The Greybus HID power-on operation is sent from host to device to power on the HID device.

Greybus HID Power ON Request

The Greybus HID power-on operation request has no payload.

Greybus HID Power ON Response

The Greybus HID power-on response has no payload.

10.6.9 Greybus HID Power OFF Operation

The Greybus HID power-off operation is sent from host to device to power off the HID device.

Offset	Field	Size	Value	Description
0	report_type	1	Number	*Greybus HID Report Type*
1	report_id	1	Number	Report ID defined by *[HID01]*

Table 10.97: HID Get Report Request

HID Report Type	Value
Input Report	0x0000
Output Report	0x0001
Feature Report	0x0002

Table 10.98: HID ReportType

Greybus HID Power OFF Request

The Greybus HID power-off operation request has no payload.

Greybus HID Power OFF Response

The Greybus HID power-off response has no payload.

10.6.10 Greybus HID Get Report Operation

The Greybus HID get report operation allows the host to fetch feature or input report synchronously from a HID device.

The get-report command is a mandatory request (if the device contains Input or Feature reports) that the host can issue to the device at any time after initialization to get a singular report from the device. The device is responsible for responding with the last known input or feature for the report id. If the value has not been set for that report yet, the device must return 0 for the length of report item.

Get-report is often used by applications on startup to retrieve the current state of the device rather than waiting for the device to generate the next Input/Feature Report.

Greybus HID Get Report Request

The Greybus HID get report request contain 1-byte report-type and report-id as defined by Table 10.97.

Greybus HID Report Type

Table 10.98 describes the defined HID report type values defined for Greybus HID devices.

Greybus HID Get Report Response

The Greybus HID Get Report response returns report as defined by *[HID01]*.

Offset	Field	Size	Value	Description
0	report_type	1	Number	*Greybus HID Report Type*
1	report_id	1	Number	Report ID defined by *[HID01]*
2	report	...	Data	Report defined by *[HID01]*

Table 10.99: HID Set Report Request

10.6.11 Greybus HID Set Report Operation

The Greybus HID set report operation allows the host to send feature or output report synchronously to a HID device.

The set-report command is a specific request that the host may issue to the device at any time after initialization to set a singular report on the device. The device is responsible for accepting the value provided in the operation and updating its state.

Greybus HID Set Report Request

The Greybus HID set report request contain report-type, report-id and report (as defined by *[HID01]*, and as defined in Table 10.99).

Greybus HID Set Report Response

The Greybus HID Set Report response has no payload.

10.6.12 Greybus HID IRQ Event Operation

The Greybus IRQ Event operation allows the AP to receive input-report asynchronously, when HID device has some data available to send to the AP.

Greybus HID IRQ Event Request

When the HID device has active data it wishes to report to the host, it will generate an data request towards the host. When the host receives the receive request, it is responsible for reading the data from the receive request.

The format of the input-report is defined by *[HID01]*.

Greybus HID IRQ Event Response

The Greybus IRQ Event response has no payload.

10.7 Lights Protocol

This section defines operations used on a connection implementing the Greybus Lights Protocol. This Protocol allows an AP Module to control Lights devices present on a Module. The Protocol consists of some basic operations that are defined here.

The operations in the Greybus Lights Protocol are:

`int cport_shutdown(u8 phase);`
See *Common Greybus Protocol CPort Shutdown Operation*.

`int get_lights(u8 *lights_count);`
Return the number of lights devices supported. lights_id used in the following operations are sequential increments from 0 to lights_count less one.

`int get_light_config(u8 light_id, u8 *channel_count, u8 *name[32]);`
Request the number of channels controlled by a light controller and its name, providing a valid identifier for that light. channel_id used in the following operations are sequential increments from 0 to channel_count less one.

`int get_channel_config(u8 light_id, u8 channel_id, struct gb_channel_config *config);`
Request a set of configuration parameters related to a channel in a light controller. The return structure elements shall map the fields of *Greybus Lights Get Channel Config Response*.

`int get_channel_flash_config(u8 light_id, u8 channel_id, struct gb_channel_flash_config *flash_config);`
Request a set of flash configuration parameters related to a channel in a light controller. The return structure elements shall map the fields of *Greybus Lights Get Channel Flash Config Response*

`int set_blink(u8 light_id, u8 channel_id, u16 time_on_ms, u16 time_off_ms);`
Set hardware blink if supported by the device, the time values are specified in milliseconds. Setting time values to 0 shall disable blink.

`int set_brightness(u8 light_id, u8 channel_id, u8 brightness);`
Set the level of brightness with the specified value.

`int set_color(u8 light_id, u8 channel_id, u32 color);`
Set color code with the specified value.

`int set_fade(u8 light_id, u8 channel_id, u32 fade_in, u32 fade_out);`
Set fade in and out level with the specified values.

`int set_flash_intensity(u8 light_id, u8 channel_id, u32 intensity_uA);`
Set flash current intensity in micro Amperes with the specified value.

`int set_flash_strobe(u8 light_id, u8 channel_id, u8 state);`
Set flash strobe state with the specified value, value 0 means strobe off other value means strobe on.

`int set_flash_timeout(u8 light_id, u8 channel_id, u32 timeout_us);`
Set flash timeout value in micro seconds with the specified value.

`int get_flash_fault(u8 light_id, u8 channel_id, *u32 fault);`
Get flash fault status from controller.

10.7.1 Greybus Lights Message Types

Table 10.100 describes the Greybus Lights operation types and their values. A message type consists of an operation type combined with a flag (0x80) indicating whether the operation is a request or a response.

10.7.2 Greybus Lights CPort Shutdown Operation

The Greybus Lights CPort Shutdown Operation is the *Common Greybus Protocol CPort Shutdown Operation* for the Lights Protocol.

Lights Operation Type	Request Value	Response Value
CPort Shutdown	0x00	0x80
Reserved	0x01	0x81
Get Lights	0x02	0x82
Get Light Config	0x03	0x83
Get Channel Config	0x04	0x84
Get Channel Flash Config	0x05	0x85
Set Brightness	0x06	0x86
Set Blink	0x07	0x87
Set Color	0x08	0x88
Set Fade	0x09	0x89
Event	0x0a	N/A
Set Flash Intensity	0x0b	0x8b
Set Flash Strobe	0x0c	0x8c
Set Flash Timeout	0x0d	0x8d
Get Flash Fault	0x0e	0x8e
(all other values reserved)	0x0f..0x7e	0x8f..0xfe
Invalid	0x7f	0xff

Table 10.100: Lights Operation Types

Offset	Field	Size	Value	Description
0	lights_count	1	Number	Number of Lights

Table 10.101: Lights Get Lights Response

10.7.3 Greybus Lights Get Lights Operation

The Greybus Lights Get Lights operation allows the requester to determine the actual number of Lights Controllers existing in the Module. If this operation fail, no further operations related to Greybus Lights shall occur.

Greybus Lights Get Lights Request

The Greybus Lights Get Lights request message has no payload.

Greybus Lights Get Lights Response

Table 10.101 describes the Greybus Lights Get Lights response. The response payload contains a one-byte value defining the number of lights controllers in the Module. If the value returned is 0 no further operations related to Greybus Lights shall follow. Lights Controllers shall be numbered sequentially starting at zero and ending in lights_count less one.

10.7.4 Greybus Lights Get Light Config Operation

The Greybus Lights Get Light Config operation allows the requester to collect a set of configuration parameters from a specific light controller. If this operation fail, all Module lights controllers configuration that already had occurred should be teared down and no further operations related to Greybus Lights shall follow.

Offset	Field	Size	Value	Description
0	light_id	1	Number	Light identification Number

Table 10.102: Lights Get Light Config Request

Offset	Field	Size	Value	Description
0	channel_count	1	Number	Number of Channels
1	name	32	UTF-8	Light Controller name

Table 10.103: Lights Get Light Config Response

Greybus Lights Get Light Config Request

Table 10.102 describes the Greybus Lights Get Light Config request. The request supplies only the light_id which is a unique identifier between 0 and lights_count less one.

Greybus Lights Get Light Config Response

Table 10.103 describes the Greybus Lights Get Light Config response. The response payload contains a one-byte value defining the number of existing channels in the Controller and thirty two byte representing the name of the Controller.

10.7.5 Greybus Lights Get Channel Config Operation

The Greybus Lights Get Channel Config operation allows the requester to collect a set of configuration parameters from a specific Channel of a Light Controller. If this operation fail, all Module lights Controllers configuration that already had occurred should be teared down and no further operations related to Greybus Lights shall follow.

Greybus Lights Get Channel Config Request

Table 10.104 describes the Greybus Lights Get Channel Config request. The request supplies the light_id and channel_id which are unique identifiers between 0 and lights_count or channel_count less one, respectively

Greybus Lights Get Channel Config Response

Table 10.105 describes the Greybus Lights Get Channel Config response. The response payload contains a set of parameters representing the configuration of the channel.

Offset	Field	Size	Value	Description
0	light_id	1	Number	Light identification Number
1	channel_id	1	Number	Channel identification Number

Table 10.104: Lights Get Channel Config Request

Offset	Field	Size	Value	Description
0	max_brightness	1	Number	Maximum Supported Value for Brightness
1	flags	4	Bit Mask	*Greybus Lights Channel Flags Bits*
5	color	4	Number	Color code value
9	color_name	32	UTF-8	Color name
41	mode	4	Bit Mask	*Greybus Lights Channel Mode Bits*
45	mode_name	32	UTF-8	Mode name

Table 10.105: Lights Get Channel Config Response

Symbol	Brief Description	Mask Value
GB_LIGHT_CHANNEL_MULTICOLOR	Channel Support more than one color	0x00000001
GB_LIGHT_CHANNEL_FADER	Channel Support Hardware Fader	0x00000002
GB_LIGHT_CHANNEL_BLINK	Channel Support Hardware Blink	0x00000004
	(All other values reserved)	0x00000008..0xffffffff

Table 10.106: Lights Channel Flag Bits

Greybus Lights Channel Flags Bits

Table 10.106 describes general flags associated to a Channel. Only the listed values are valid.

Greybus Lights Channel Mode Bits

Table 10.107 describes possible modes associated to a Channel. Only the listed values are valid.

10.7.6 Greybus Lights Get Channel Flash Config Operation

The Greybus Lights Get Channel Flash Config operation allows the requester to collect a set of configuration parameters related to flash type modes from a specific Channel of a Light Controller. If this operation fail, all Module lights Controllers configuration that already had occurred should be teared down and no further operations related to Greybus Lights shall follow.

Light Mode	Brief Description	Mask Value
GB_CHANNEL_MODE_NONE	Channel do not represent any specific mode	0x00000000
GB_CHANNEL_MODE_BATTERY	Channel can represent the battery mode	0x00000001
GB_CHANNEL_MODE_POWER	Channel can represent the power mode	0x00000002
GB_CHANNEL_MODE_WIRELESS	Channel can represent wifi activity mode	0x00000004
GB_CHANNEL_MODE_BLUETOOTH	Channel can represent bluetooth activity mode	0x00000008
GB_CHANNEL_MODE_KEYBOARD	Channel can represent light related to the keyboard	0x00000010
GB_CHANNEL_MODE_BUTTONS	Channel can represent light related to buttons	0x00000020
GB_CHANNEL_MODE_NOTIFICATION	Channel can represent general notification light	0x00000040
GB_CHANNEL_MODE_ATTENTION	Channel can represent general attention light	0x00000080
GB_CHANNEL_MODE_FLASH	Channel can be used as a flash light device	0x00000100
GB_CHANNEL_MODE_TORCH	Channel can be used as a flash torch device	0x00000200
GB_CHANNEL_MODE_INDICATOR	Channel can be used as a flash indicator device	0x00000400
	(Reserved Range)	0x00000800..0x00080000
GB_CHANNEL_MODE_VENDOR	Channel can be used as vendor specific mode	0x00100000..0x08000000
	(Reserved Range)	0x10000000..0x80000000

Table 10.107: Lights Channel Mode Bit Masks

Offset	Field	Size	Value	Description
0	light_id	1	Number	Light identification Number
1	channel_id	1	Number	Channel identification Number

Table 10.108: Lights Get Channel Flash Config Request

Offset	Field	Size	Value	Description
0	intensity_min_uA	4	Number	Minimum Value for Current Intensity in microampere
4	intensity_max_uA	4	Number	Maximum Value for Current Intensity in microampere
8	intensity_step_uA	4	Number	Step Value for Current Intensity in microampere
12	timeout_min_us	4	Number	Minimum Value for Strobe Flash timeout in microseconds
16	timeout_max_us	4	Number	Maximum Value for Strobe Flash timeout in microseconds
20	timeout_step_us	4	Number	Step Value for Strobe Flash timeout in microseconds

Table 10.109: Lights Get Channel Flash Config Response

Greybus Lights Get Channel Flash Config Request

Table 10.108 describes the Greybus Lights Get Channel Config request. The request supplies the light_id and channel_id which are unique identifiers between 0 and lights_count or channel_count less one, respectively

Greybus Lights Get Channel Flash Config Response

Table 10.109 describes the Greybus Lights Get Channel Flash Config response. The response payload contains a set of flash type parameters representing the configuration of the channel.

10.7.7 Greybus Lights Set Brightness Operation

The Greybus Lights Set Brightness operation allows the requester to set brightness level of a specific Channel to the specified value.

Greybus Lights Set Brightness Request

The Greybus Lights Set Brightness request payload contains three 1-byte values that represents light_id, channel_id and the level of brightness to be set by the light device channel being controlled, in which 0 represent the lower level (off) and 255 represent the highest possible brightness level as defined in table 10.110.

Greybus Lights Set Brightness Response

The Greybus Lights Set Brightness response message has no payload.

Offset	Field	Size	Value	Description
0	light_id	1	Number	Light identification Number
1	channel_id	1	Number	Channel identification Number
2	brightness	1	Number	Channel brightness level to set

Table 10.110: Lights Set Brightness Request

Offset	Field	Size	Value	Description
0	light_id	1	Number	Light identification Number
1	channel_id	1	Number	Channel identification Number
2	time_on_ms	2	Number	Time on in milliseconds
4	time_off_ms	2	Number	Time off in milliseconds

Table 10.111: Lights Set Blink Request

Offset	Field	Size	Value	Description
0	light_id	1	Number	Light identification Number
1	channel_id	1	Number	Channel identification Number
2	color	4	Number	Channel color code

Table 10.112: Lights Set Color Request

10.7.8 Greybus Lights Set Blink Operation

The Greybus Lights Set Blink operation allows the requester to enable the blink mode of a specific Channel. Setting time_on and time_off to 0 or setting brightness level to a fixed value shall disable blink.

Greybus Lights Set Blink Request

The Greybus Lights Set Blink request payload contains a two 1-byte values that represent the light_id and channel_id, more two 2-byte values that represents the duration in milliseconds of the on and off period during the blink to be set by the light device channel being controlled, as defined in table 10.111.

Greybus Lights Set Blink Response

The Greybus Lights Set Blink response message has no payload.

10.7.9 Greybus Lights Set Color Operation

The Greybus Lights Set Color operation allows the requester to set a value for a color space of a specific Channel to the specified value.

Greybus Lights Set Color Request

The Greybus Lights Set Color request payload contains two 1-byte values that represents light_id, channel_id and one 4-byte value which represents a color code in any color space for the light device channel, as defined in table 10.112.

Greybus Lights Set Color Response

The Greybus Lights Set Color response message has no payload.

Offset	Field	Size	Value	Description
0	light_id	1	Number	Light identification Number
1	channel_id	1	Number	Channel identification Number
2	fade_in	2	Number	Fade in level
4	fade_out	2	Number	Fade out level

Table 10.113: Lights Set Fade Request

Offset	Field	Size	Value	Description
0	light_id	1	Number	Light identification Number
1	event	1	Bit Mask	*Greybus Lights Event Bit Masks*

Table 10.114: Lights Event Request

10.7.10 Greybus Lights Set Fade Operation

The Greybus Lights Set Fade operation allows the requester to enable and set the parameters for fade effect of a specific Channel.

Greybus Lights Set Fade Request

The Greybus Lights Set Fade request payload contains a two 1-byte values that represent the light_id and channel_id, more two 2-byte values that represents a level of the fade in and out effect during brightness transitions by the light device channel being controlled, as defined in table 10.113.

Greybus Lights Set Fade Response

The Greybus Lights Set Fade response message has no payload.

10.7.11 Greybus Lights Event Operation

The Greybus Lights Event operation signals to the recipient that a change in the device setup have occurred.

This event shall be discarded by the recipient until a valid light controller configuration is known.

This operation is unidirectional and does not have a correspondent response.

Greybus Lights Event Request

Table 10.114 defines the Greybus Lights Event request. The request payload supplies two 1-byte fields that represent the light_id and event bit mask.

Greybus Lights Event Bit Masks

Table 10.115 defines the bit masks which specify the set of events that occurred in the sending controller.

Symbol	Brief Description	Mask Value
GB_LIGHTS_LIGHT_CONFIG	Configuration Changed	0x01
	(All other values reserved)	0x02..0x80

Table 10.115: Lights Protocol Event Bit Mask

Offset	Field	Size	Value	Description
0	light_id	1	Number	Light identification Number
1	channel_id	1	Number	Channel identification Number
2	intensity_uA	4	Number	Current Intensity in microamperes

Table 10.116: Lights Set Flash Intensity Request

10.7.12 Greybus Lights Set Flash Intensity Operation

The Greybus Lights Set Flash Intensity operation allows the requester to set current Intensity level in microamperes of a Channel to the specified value.

Greybus Lights Set Flash Intensity Request

The Greybus Lights Set Flash Intensity request payload contains two 1-byte values that represent the light_id and channel_id, and 4-byte value that represents the current intensity in microamperes. The value shall be set between the minimum and maximum values got from flash configuration operation. 10.116.

Greybus Lights Set Flash Intensity Response

The Greybus Lights Set Flash Intensity response message has no payload.

10.7.13 Greybus Lights Set Flash Strobe Operation

The Greybus Lights Set Flash Strobe operation allows the requester to enable or disable the strobe associated with a Channel.

Greybus Lights Set Flash Strobe Request

The Greybus Lights Set Flash Strobe request payload contains three 1-byte values that represents light_id, channel_id and the strobe state to be set. If state is 0 means disable, 1 means enable. Any other value shall be considered invalid. 10.117.

Offset	Field	Size	Value	Description
0	light_id	1	Number	Light identification Number
1	channel_id	1	Number	Channel identification Number
2	state	1	Number	Strobe state to be set

Table 10.117: Lights Set Flash Strobe Request

Offset	Field	Size	Value	Description
0	light_id	1	Number	Light identification Number
1	channel_id	1	Number	Channel identification Number
2	timeout_us	4	Number	Timeout Value in microseconds

Table 10.118: Lights Set Flash Timeout Request

Offset	Field	Size	Value	Description
0	light_id	1	Number	Light identification Number
1	channel_id	1	Number	Channel identification Number

Table 10.119: Lights Get Flash Fault Request

Greybus Lights Set Flash Strobe Response

The Greybus Lights Set Flash Strobe response message has no payload.

10.7.14 Greybus Lights Set Flash Timeout Operation

The Greybus Lights Set Flash Timeout operation allows the requester to set flash timeout in microseconds of a Channel to the specified value.

Greybus Lights Set Flash Timeout Request

The Greybus Lights Set Flash Timeout request payload contains two 1-byte values that represent the light_id and channel_id, and 4-byte value that represents the flash timeout in microseconds. The value shall be set between the minimum and maximum values got from flash configuration operation. 10.118.

Greybus Lights Set Flash Timeout Response

The Greybus Lights Set Flash Timeout response message has no payload.

10.7.15 Greybus Lights Get Flash Fault Operation

The Greybus Lights Get Flash Fault operation allows the requester to get a detailed information of the status and fault reasons of the flash type controller.

Greybus Lights Get Flash Fault Request

The Greybus Lights Get Flash Fault request payload contains two 1-byte values that represent the light_id and channel_id. 10.119.

Greybus Lights Get Flash Fault Response

The Greybus Lights Get Flash Fault response message payload contains a 4-byte bit mask with the current fault status of the flash controller, as defined in table 10.120

Offset	Field	Size	Value	Description
0	fault	4	Bit Mask	*Greybus Lights Flash Fault Bit Masks*

Table 10.120: Lights Get Flash Fault Response

Symbol	Brief Description	Mask Value
GB_LIGHTS_FLASH_FAULT_OVER_VOLTAGE	Over Voltage	0x00000001
GB_LIGHTS_FLASH_FAULT_TIMEOUT	Timeout	0x00000002
GB_LIGHTS_FLASH_FAULT_OVER_TEMPERATURE	Over Temperature	0x00000004
GB_LIGHTS_FLASH_FAULT_SHORT_CIRCUIT	Short Circuit	0x00000008
GB_LIGHTS_FLASH_FAULT_OVER_CURRENT	Over Current	0x00000010
GB_LIGHTS_FLASH_FAULT_INDICATOR	Indicator Fault	0x00000020
GB_LIGHTS_FLASH_FAULT_UNDER_VOLTAGE	Under Voltage	0x00000040
GB_LIGHTS_FLASH_FAULT_INPUT_VOLTAGE	Input Voltage	0x00000080
GB_LIGHTS_FLASH_FAULT_LED_OVER_TEMPERATURE	LED Over Temperature	0x00000100
	(All other values reserved)	0x00000200..0x80000000

Table 10.121: Lights Protocol Flash Fault Bit Mask

Greybus Lights Flash Fault Bit Masks

Table 10.121 defines the bit masks which specify the fault status of the flash controller.

10.8 Log Protocol

This section defines the Operations used on a Connection implementing the Greybus Log Protocol. This Protocol allows an Interface to send human-readable debug log messages to the AP. These messages are typically meant to be displayed by the AP's system logger (e.g. `dmesg`).

The Operations in the Greybus Log Protocol are:

`int cport_shutdown(u8 phase);`
See *Common Greybus Protocol CPort Shutdown Operation*.

`int send_log(u16 len, char *log);`
Log message from an Interface to the AP asynchronously

10.8.1 Greybus Log Message Types

Table 10.122 describes the Greybus Log Operation types and their values. A message type consists of an Operation type combined with a flag (0x80) indicating whether the Operation is a Request or a Response.

Log Operation Type	Request Value	Response Value
CPort Shutdown	0x00	0x80
Reserved	0x01	0x81
Send Log	0x02	0x82
(all other values reserved)	0x03..0x7e	0x83..0xfe
Invalid	0x7f	0xff

Table 10.122: Log Operation Types

Offset	Field	Size	Value	Description
0	length	2	Number	Length in bytes of the log message
2	log	X	UTF-8	Content of the log message

Table 10.123: Log Protocol Send Log Request

10.8.2 Greybus Log CPort Shutdown Operation

The Greybus Log CPort Shutdown Operation is the *Common Greybus Protocol CPort Shutdown Operation* for the Log Protocol.

10.8.3 Greybus Log Send Log Operation

The Greybus Log Send Log Operation sends a log message from the Interface to the AP in an asynchronous way. A log message is described by a null-terminated sequence of UTF-8 characters and its associated length.

Greybus Log Send Log Request

Table 10.123 defines the Greybus Log Send Log Request. The Request supplies the size of the log message that is sent by the Interface and the log message itself.

Greybus Log Send Log Response

The Greybus Log Send Log Response message has no payload.

10.9 Loopback Protocol

This section defines the operations used on a connection implementing the Greybus loopback Protocol. This Protocol is used for testing a Greybus device and the connection to the device, by sending and receiving data in a "loop".

The operations in the Greybus loopback Protocol are:

`int cport_shutdown(u8 phase);`
: See *Common Greybus Protocol CPort Shutdown Operation*.

`int ping(void);`
: Sends a "ping" message to the device, from the host, that needs to be acknowledged by the device. By measuring how long this message takes to succeed, an idea of the speed of the connection can be made.

`int transfer(u32 len, char *send, char *receive);`
: Sends a stream of bytes to the device and receives them back from the device.

`int sink(u32 len, char *send);`
: Sends a stream of bytes to the device that needs to be acknowledged by the device. No data are sent back from the device.

Loopback Operation Type	Request Value	Response Value
CPort Shutdown	0x00	0x80
Reserved	0x01	0x81
Ping	0x02	0x82
Transfer	0x03	0x83
Sink	0x04	0x84
(all other values reserved)	0x05..0x7e	0x85..0xfe
Invalid	0x7f	0xff

Table 10.124: Loopback Operation Types

10.9.1 Greybus Loopback Message Types

Table 10.124 describes the Greybus loopback operation types and their values. A message type consists of an operation type combined with a flag (0x80) indicating whether the operation is a request or a response.

10.9.2 Greybus Loopback CPort Shutdown Operation

The Greybus Loopback CPort Shutdown Operation is the *Common Greybus Protocol CPort Shutdown Operation* for the Loopback Protocol.

10.9.3 Greybus Loopback Ping Operation

The Greybus ping operation is a simple message that has no response. It is used to time how long a single message takes to be sent and acknowledged from the receiver.

Greybus Loopback Ping Request

The Greybus ping request message has no payload.

Greybus Loopback Ping Response

The Greybus ping response message has no payload.

10.9.4 Greybus Loopback Transfer Operation

The Greybus Loopback transfer operation sends data and then the same data is returned. This is used to determine the time required to transfer different size messages. To facilitate analysis, the messages used for both the Loopback Transfer Operation request and response message have identical formats.

Greybus Loopback Transfer Request

Table 10.125 defines the Greybus Loopback Transfer request. The request supplies size of the data that is sent to the device, and the data itself.

Offset	Field	Size	Value	Description
0	len	4	Number	length in bytes of the data field
4	reserved0	4	Number	Not used - same size as response
8	reserved1	4	Number	Not used - same size as response
12	data	X	Data	array of data bytes

Table 10.125: Loopback Protocol Transfer Request

Offset	Field	Size	Value	Description
0	len	4	Number	length in bytes of the data field
4	reserved0	4	Number	reserved for use by the implementation
8	reserved1	4	Number	reserved for use by the implementation
12	data	X	Data	array of data bytes

Table 10.126: Loopback Protocol Transfer Response

Greybus Loopback Transfer Response

Table 10.126 defines the Greybus Loopback Transfer response. The response contains the same data that was sent in the request.

10.9.5 Greybus Loopback Sink Operation

The Greybus Loopback sink operation sends data to the device. No data is returned back.

Greybus Loopback Sink Request

The Greybus sink request message is identical to the Greybus transfer request message.

Greybus Loopback Sink Response

The Greybus sink response message has no payload.

10.10 Power Supply Protocol

This section defines the operations used on a connection implementing the Greybus Power Supply Protocol. This Protocol allows to manage a power supply controller present on a Module. The Protocol consists of few basic operations, whose request and response message formats are defined here.

Conceptually, the operations in the Greybus Power Supply Protocol are:

`int cport_shutdown(u8 phase);`
See *Common Greybus Protocol CPort Shutdown Operation*.

`int get_power_supplies(u8 *psy_count);`
Returns a value indicating the number of devices that this power supply adapter controls.

`int get_description(u8 psy_id, struct gb_power_supply_description *description);`
Returns set of values related to a specific power supply controller defined by psy_id in the power supply

Power Supply Operation Type	Request Value	Response Value
CPort Shutdown	0x00	0x80
Reserved	0x01	0x81
Get Power Supplies	0x02	0x82
Get Description	0x03	0x83
Get Property Descriptors	0x04	0x84
Get Property	0x05	0x85
Set Property	0x06	0x86
Event	0x07	N/A
(all other values reserved)	0x08..0x7e	0x88..0xfe
Invalid	0x7f	0xff

Table 10.127: Power Supply Operation Types

adapter. The return structure elements shall map the fields of *Greybus Power Supply Get Description Response*

`int get_property_descriptors(u8 psy_id, u8 *properties_count, struct gb_power_supply_property_desc *props)`
Returns the number of property descriptors and set of descriptors related to a specific power supply defined by psy_id in the power supply adapter. The property descriptor shall map to the fields of *Greybus Power Supply Property Descriptor*. The number of properties can be zero.

`int get_property(u8 psy_id, u8 property, u32 *prop_val);`
Returns the current value of a property in a specific psy_id in the power supply adapter.

`int set_property(u8 psy_id, u8 property, u32 prop_val);`
It sets the value of a given property in a specified psy_id, if the property is not described in is descriptor as writable, this operation shall be discarded.

`int event(u8 *type);`
Input event sent from the device to host asynchronously.

10.10.1 Greybus Power Supply Message Types

Table 10.127 describes the Greybus power supply operation types and their values. A message type consists of an operation type combined with a flag (0x80) indicating whether the operation is a request or a response.

10.10.2 Greybus Power Supply CPort Shutdown Operation

The Greybus Power Supply CPort Shutdown Operation is the *Common Greybus Protocol CPort Shutdown Operation* for the Power Supply Protocol.

10.10.3 Greybus Power Supply Get Power Supplies Operation

The Greybus power supply get power supplies operation allows requester to determine the number of power supply devices controlled by the power supply adapter. Power Supply Controllers shall be numbered sequentially starting at zero and ending at psy_count less one.

Greybus Power Supply Get Power Supplies Request

The Greybus power supply get power supplies request message has no payload.

Offset	Field	Size	Value	Description
0	psy_count	1	Number	Number of Power Supplies controlled

Table 10.128: Power Supply Get Power Supplies Response

Offset	Field	Size	Value	Description
0	psy_id	1	Number	Power Supply identification Number

Table 10.129: Power Supply Get Description Request

Greybus Power Supply Get Power Supplies Response

The Greybus power supply get power supplies response contains a 1-byte value that represents the number of power supply being controlled as defined in Table 10.128.

10.10.4 Greybus Power Supply Get Description Operation

The Greybus power supply get description operation allows requester to determine a set of configuration parameters from a specific power supply controller.

Greybus Power Supply Get Description Request

Table 10.129 describes the Greybus Power Supply Get Description request. The request supplies only the psy_id which is an unique identifier between 0 and power supplies_count less one.

Greybus Power Supply Get Description Response

Table 10.130 describes the Greybus Power Supply Get Description response. The response payload contains a set of parameters representing the configuration of a power supply.

Greybus Power Supply Type

Table 10.131 describes the defined power supply types defined for Greybus power supply adapters.

10.10.5 Greybus Power Supply Get Property Descriptors Operation

The Greybus power supply get property descriptors operation allows requester to determine the set of properties supported by the power supply controller and if the property support the :ref:Set Property Operation.

Offset	Field	Size	Value	Description
0	manufacturer	32	UTF-8	Manufacturer name
32	model	32	UTF-8	Model name
64	serial_number	32	UTF-8	Serial Number
96	type	2	Number	*Greybus Power Supply Type*
98	properties_count	1	Number	Number of properties

Table 10.130: Power Supply Get Description Response

Power Supply Type	Value	Description
GB_POWER_SUPPLY_UNKNOWN_TYPE	0x0000	Unknown Type
GB_POWER_SUPPLY_BATTERY_TYPE	0x0001	Battery Type
GB_POWER_SUPPLY_UPS_TYPE	0x0002	Uninterruptible Power Supply Type
GB_POWER_SUPPLY_MAINS_TYPE	0x0003	AC Power Supply Type
GB_POWER_SUPPLY_USB_TYPE	0x0004	USB Standard Downstream Port
GB_POWER_SUPPLY_USB_DCP_TYPE	0x0005	USB Dedicated Charging Port
GB_POWER_SUPPLY_USB_CDP_TYPE	0x0006	USB Charging Downstream Port
GB_POWER_SUPPLY_USB_ACA_TYPE	0x0007	USB Accessory Charger Adapters
GB_POWER_SUPPLY_USB_HVDCP_TYPE	0x0008	USB High Voltage DCP
GB_POWER_SUPPLY_USB_TYPE_C_TYPE	0x0009	USB Type C Port
GB_POWER_SUPPLY_USB_PD_TYPE	0x000A	USB Power Delivery Port
GB_POWER_SUPPLY_USB_PD_DRP_TYPE	0x000B	USB Power Delivery Dual Role Port
GB_POWER_SUPPLY_WIRELESS_TYPE	0x000C	Wireless Power Supply Type

Table 10.131: Power Supply Type

Offset	Field	Size	Value	Description
0	psy_id	1	Number	Power Supply identification Number

Table 10.132: Power Supply Get Property Descriptor Request

Greybus Power Supply Get Property Descriptors Request

Table 10.132 describes the Greybus Power Supply Get Property Descriptors request. The request supplies only the psy_id which is an unique identifier between 0 and power supplies_count less one.

Greybus Power Supply Get Property Descriptors Response

Table 10.133 describes the Greybus Power Supply Get Property Descriptors response. The response payload contains the number and the properties descriptors in this response.

Greybus Power Supply Property Descriptor

Table 10.134 describes a property descriptor which contains the descriptor type and writable indication.

Greybus Power Supply Property Type

Table 10.135 describes the defined power supply properties for the Greybus power supply adapters. All voltages, currents, charges, energies, time and temperatures in micro-volt(μV), micro-ampere(μA), micro-ampere-hour(μAh), micro-watt-hour(μWh), seconds and tenths of degrees Celsius unless otherwise stated.

Offset	Field	Size	Value	Description
0	properties_count	1	Number	Number of properties descriptors
1	props[N]	(2*N)	Structure	N Property Descriptors *Greybus Power Supply Property Descriptor*

Table 10.133: Power Supply Get Property Descriptors Response

Offset	Field	Size	Value	Description
0	property	1	Number	*Greybus Power Supply Property Type*
1	is_writable	1	Number	Writable Property

Table 10.134: Power Supply Property Descriptor

In Table 10.135 the mandatory column indicates power supply properties that shall be supported by the Greybus Power Supply protocol. If a mandatory property is missing or advertises invalid mode the Greybus Power Supply Protocol connection shall be terminated.

Following notation is used to show these mandatory properties and their corresponding modes for each Greybus Power Supply type.

- [Mode][Type], where:
  - Mode can be: R (read) / W (write)
  - Type, based on values in *Greybus Power Supply Type*, can be:
    - B (Battery - GB_POWER_SUPPLY_BATTERY_TYPE)
    - C (Charger - From GB_POWER_SUPPLY_USB_TYPE to GB_POWER_SUPPLY_WIRELESS_TYPE types from the *Greybus Power Supply Type*)
    - O (Others - all other types)

Greybus Power Supply Property Status

Table 10.136 describes the defined power supply status values available for Greybus power supply adapters.

Greybus Power Supply Property Charge

Table 10.137 describes the defined power supply charge types available for Greybus power supply adapters.

Greybus Power Supply Property Health

Table 10.138 describes the defined power supply health values available for Greybus power supply adapters.

Greybus Power Supply Property Technology

Table 10.139 describes the defined power supply technologies available for Greybus power supply adapters.

Greybus Power Supply Property Capacity

Table 10.140 describes the defined power supply capacity levels available for battery adapters.

Greybus Power Supply Property Scope

Table 10.141 describes the defined power supply scopes available for Greybus power supply adapters.

Power Supply Property	Value	Mandatory	Description Battery
GB_POWER_SUPPLY_PROP_STATUS	0x00	RB-RC-RO	*Greybus Power Supply Property Status*
GB_POWER_SUPPLY_PROP_CHARGE_TYPE	0x01	RC	*Greybus Power Supply Property Charge*
GB_POWER_SUPPLY_PROP_HEALTH	0x02	RB-RC-RO	*Greybus Power Supply Property Health*
GB_POWER_SUPPLY_PROP_PRESENT	0x03	RB-RC-RO	Presence indicator (1 is present, 0 is not present).
GB_POWER_SUPPLY_PROP_ONLINE	0x04	RC-RO	Online indicator (1 is online, 0 is not online)
GB_POWER_SUPPLY_PROP_AUTHENTIC	0x05		Authentic indicator (1 is authentic, 0 is not authentic)
GB_POWER_SUPPLY_PROP_TECHNOLOGY	0x06	RB	*Greybus Power Supply Property Technology*
GB_POWER_SUPPLY_PROP_CYCLE_COUNT	0x07		A complete charge cycle counter
GB_POWER_SUPPLY_PROP_VOLTAGE_MAX	0x08	RC,WC	Value from measure and retain maximum Voltage
GB_POWER_SUPPLY_PROP_VOLTAGE_MIN	0x09		Value from measure and retain minimum Voltage
GB_POWER_SUPPLY_PROP_VOLTAGE_MAX_DESIGN	0x0A		Maximum value for Voltage by design
GB_POWER_SUPPLY_PROP_VOLTAGE_MIN_DESIGN	0x0B		Minimum value for Voltage by design
GB_POWER_SUPPLY_PROP_VOLTAGE_NOW	0x0C	RB	Instantaneous Voltage value
GB_POWER_SUPPLY_PROP_VOLTAGE_AVG	0x0D		Average Voltage value
GB_POWER_SUPPLY_PROP_VOLTAGE_OCV	0x0E		Open Circuit Voltage
GB_POWER_SUPPLY_PROP_VOLTAGE_BOOT	0x0F		Voltage during boot
GB_POWER_SUPPLY_PROP_CURRENT_MAX	0x10	RC,WC	Maximum Current Value
GB_POWER_SUPPLY_PROP_CURRENT_NOW	0x11	RB	Instantaneous Current Value
GB_POWER_SUPPLY_PROP_CURRENT_AVG	0x12		Average Current value
GB_POWER_SUPPLY_PROP_CURRENT_BOOT	0x13		Current measured at boot
GB_POWER_SUPPLY_PROP_POWER_NOW	0x14		Instantaneous Power consumption
GB_POWER_SUPPLY_PROP_POWER_AVG	0x15		Average Power consumption
GB_POWER_SUPPLY_PROP_CHARGE_FULL_DESIGN	0x16	RB	Threshold for full charge by design
GB_POWER_SUPPLY_PROP_CHARGE_EMPTY_DESIGN	0x17	RB	Threshold for empty charge value by design
GB_POWER_SUPPLY_PROP_CHARGE_FULL	0x18	RB	Value from measure and retain maximum charge
GB_POWER_SUPPLY_PROP_CHARGE_EMPTY	0x19		Value from measure and retain minimum charge
GB_POWER_SUPPLY_PROP_CHARGE_NOW	0x1A	RB	Instantaneous charge value
GB_POWER_SUPPLY_PROP_CHARGE_AVG	0x1B		Average charge value
GB_POWER_SUPPLY_PROP_CHARGE_COUNTER	0x1C		Charge counter
GB_POWER_SUPPLY_PROP_CONSTANT_CHARGE_CURRENT	0x1D		Charge Current programmed by charger
GB_POWER_SUPPLY_PROP_CONSTANT_CHARGE_CURRENT_MAX	0x1E		Maximum charge current supported
GB_POWER_SUPPLY_PROP_CONSTANT_CHARGE_VOLTAGE	0x1F	RC,WC	Charge Voltage programmed by charger
GB_POWER_SUPPLY_PROP_CONSTANT_CHARGE_VOLTAGE_MAX	0x20		Maximum charge voltage supported
GB_POWER_SUPPLY_PROP_CHARGE_CONTROL_LIMIT	0x21		Current charge control limit
GB_POWER_SUPPLY_PROP_CHARGE_CONTROL_LIMIT_MAX	0x22		Maximum charge control limit
GB_POWER_SUPPLY_PROP_INPUT_CURRENT_LIMIT	0x23	RC,WC	Input current limit programmed by charger
GB_POWER_SUPPLY_PROP_ENERGY_FULL_DESIGN	0x24		Threshold for full energy by design
GB_POWER_SUPPLY_PROP_ENERGY_EMPTY_DESIGN	0x25		Threshold for empty energy by design
GB_POWER_SUPPLY_PROP_ENERGY_FULL	0x26		Value from measure and retain maximum energy
GB_POWER_SUPPLY_PROP_ENERGY_EMPTY	0x27		Value from measure and retain minimum energy
GB_POWER_SUPPLY_PROP_ENERGY_NOW	0x28		Instantaneous energy value
GB_POWER_SUPPLY_PROP_ENERGY_AVG	0x29		Average energy value
GB_POWER_SUPPLY_PROP_CAPACITY	0x2A	RB	Capacity in percents
GB_POWER_SUPPLY_PROP_CAPACITY_ALERT_MIN	0x2B		Minimum capacity alert value in percents
GB_POWER_SUPPLY_PROP_CAPACITY_ALERT_MAX	0x2C		Maximum capacity alert value in percents
GB_POWER_SUPPLY_PROP_CAPACITY_LEVEL	0x2D	RB	*Greybus Power Supply Property Capacity*
GB_POWER_SUPPLY_PROP_TEMP	0x2E	RB-RC	Temperature
GB_POWER_SUPPLY_PROP_TEMP_MAX	0x2F	RB	Maximum operable temperature
GB_POWER_SUPPLY_PROP_TEMP_MIN	0x30	RB	Minimum operable temperature
GB_POWER_SUPPLY_PROP_TEMP_ALERT_MIN	0x31		Minimum temperature alert
GB_POWER_SUPPLY_PROP_TEMP_ALERT_MAX	0x32		Maximum temperature alert
GB_POWER_SUPPLY_PROP_TEMP_AMBIENT	0x33		Ambient temperature
GB_POWER_SUPPLY_PROP_TEMP_AMBIENT_ALERT_MIN	0x34		Minimum ambient temperature alert
GB_POWER_SUPPLY_PROP_TEMP_AMBIENT_ALERT_MAX	0x35		Maximum ambient temperature alert
GB_POWER_SUPPLY_PROP_TIME_TO_EMPTY_NOW	0x36		Instantaneous seconds left to be considered empty
GB_POWER_SUPPLY_PROP_TIME_TO_EMPTY_AVG	0x37		Average seconds left to be considered empty
GB_POWER_SUPPLY_PROP_TIME_TO_FULL_NOW	0x38		Instantaneous seconds left to be considered full
GB_POWER_SUPPLY_PROP_TIME_TO_FULL_AVG	0x39		Average seconds left to be considered full
GB_POWER_SUPPLY_PROP_TYPE	0x3A	RB-RC-RO	*Greybus Power Supply Type*
GB_POWER_SUPPLY_PROP_SCOPE	0x3B		*Greybus Power Supply Property Scope*
GB_POWER_SUPPLY_PROP_CHARGE_TERM_CURRENT	0x3C	RC,WC	Charge Termination current
GB_POWER_SUPPLY_PROP_CALIBRATE	0x3D		Calibration status
GB_POWER_SUPPLY_PROP_USB_HC	0x3E		High Current USB
GB_POWER_SUPPLY_PROP_USB_OTG	0x3F	RC,WC	OTG boost property
GB_POWER_SUPPLY_PROP_CHARGING_ENABLED	0x40	RC,WC	Control charging status

Table 10.135: Power Supply Property Type

Power Supply Status	Value
GB_POWER_SUPPLY_STATUS_UNKNOWN	0x0000
GB_POWER_SUPPLY_STATUS_CHARGING	0x0001
GB_POWER_SUPPLY_STATUS_DISCHARGING	0x0002
GB_POWER_SUPPLY_STATUS_NOT_CHARGING	0x0003
GB_POWER_SUPPLY_STATUS_FULL	0x0004

Table 10.136: Power Supply Property Status

Power Supply Charge	Value
GB_POWER_SUPPLY_CHARGE_TYPE_NONE	0x0001
GB_POWER_SUPPLY_CHARGE_TYPE_TRICKLE	0x0002
GB_POWER_SUPPLY_CHARGE_TYPE_FAST	0x0003

Table 10.137: Power Supply Property Charge

Power Supply Health	Value
GB_POWER_SUPPLY_HEALTH_UNKNOWN	0x0000
GB_POWER_SUPPLY_HEALTH_GOOD	0x0001
GB_POWER_SUPPLY_HEALTH_OVERHEAT	0x0002
GB_POWER_SUPPLY_HEALTH_DEAD	0x0003
GB_POWER_SUPPLY_HEALTH_OVERVOLTAGE	0x0004
GB_POWER_SUPPLY_HEALTH_UNSPEC_FAILURE	0x0005
GB_POWER_SUPPLY_HEALTH_COLD	0x0006
GB_POWER_SUPPLY_HEALTH_WATCHDOG_TIMER_EXPIRE	0x0007
GB_POWER_SUPPLY_HEALTH_SAFETY_TIMER_EXPIRE	0x0008

Table 10.138: Power Supply Property Health

Power Supply Technology	Value
GB_POWER_SUPPLY_TECH_UNKNOWN	0x0000
GB_POWER_SUPPLY_TECH_NiMH	0x0001
GB_POWER_SUPPLY_TECH_LION	0x0002
GB_POWER_SUPPLY_TECH_LIPO	0x0003
GB_POWER_SUPPLY_TECH_LiFe	0x0004
GB_POWER_SUPPLY_TECH_NiCd	0x0005
GB_POWER_SUPPLY_TECH_LiMn	0x0006

Table 10.139: Power Supply Property Technology

Power Supply Capacity	Value
GB_POWER_SUPPLY_CAPACITY_LEVEL_UNKNOWN	0x0000
GB_POWER_SUPPLY_CAPACITY_LEVEL_CRITICAL	0x0001
GB_POWER_SUPPLY_CAPACITY_LEVEL_LOW	0x0002
GB_POWER_SUPPLY_CAPACITY_LEVEL_NORMAL	0x0003
GB_POWER_SUPPLY_CAPACITY_LEVEL_HIGH	0x0004
GB_POWER_SUPPLY_CAPACITY_LEVEL_FULL	0x0005

Table 10.140: Power Supply Property Capacity

Power Supply Scope	Value
GB_POWER_SUPPLY_SCOPE_UNKNOWN	0x0000
GB_POWER_SUPPLY_SCOPE_SYSTEM	0x0001
GB_POWER_SUPPLY_SCOPE_DEVICE	0x0002

Table 10.141: Power Supply Property Scope

Offset	Field	Size	Value	Description
0	psy_id	1	Number	Power Supply identification Number
1	property	1	Number	*Greybus Power Supply Property Type*

Table 10.142: Power Supply Get Property Request

10.10.6 Greybus Power Supply Get Property Operation

The Greybus power supply get property operation allows requester to determine the current value of a property supported by the power supply controller.

Greybus Power Supply Get Property Request

Table 10.142 describes the Greybus Power Supply Get Property request. The request supplies only the psy_id which is an unique identifier between 0 and psy_count less one and the property to fetch the value.

Greybus Power Supply Get Property Response

Table 10.143 describes the Greybus Power Supply Get Property response. The response returns the current value of the property issued in the request.

10.10.7 Greybus Power Supply Set Property Operation

The Greybus power supply set property operation allows requester to change the current value of a property supported by the power supply controller. This operation shall fail if the property is not set as writable.

Greybus Power Supply Set Property Request

Table 10.144 describes the Greybus Power Supply Set Property request. The request supplies the psy_id which is an unique identifier between 0 and power supplies_count less one, the property to alter and the new value.

Greybus Power Supply Set Property Response

The Greybus power supply Set Property response message has no payload.

Offset	Field	Size	Value	Description
0	prop_val	4	Number	Property value

Table 10.143: Power Supply Get Property Response

Offset	Field	Size	Value	Description
0	psy_id	1	Number	Power Supply identification Number
1	property	1	Number	*Greybus Power Supply Property Type*
2	prop_val	4	Number	Property value

Table 10.144: Power Supply Set Property Request

Offset	Field	Size	Value	Description
0	psy_id	1	Number	Power Supply identification Number
1	event	1	Bit Mask	*Greybus Power Supply Event Bit Masks*

Table 10.145: Power Supply Event Request

Greybus Power Supply Event Request

Table 10.145 defines the Greybus Power Supply Event request. The request payload supplies two 1-byte fields that represent the psy_id and event bit mask.

Greybus Power Supply Event Bit Masks

Table 10.146 defines the bit masks which specify the set of events that occurred in the sending controller.

10.11 Raw Protocol

This section defines the operations used on a connection implementing the Greybus Raw Protocol. This Protocol is used for streaming "raw" data from userspace directly to or from the device. The data contained by the protocol is not interpreted by the kernel, but requires a userspace program to handle it. It can almost be considered a "vendor specific" protocol in that the format of the data is unspecified, and will vary by device.

The operations in the Greybus Raw Protocol are:

`int cport_shutdown(u8 phase);`
See *Common Greybus Protocol CPort Shutdown Operation*.

`int send(u32 len, char *data);`
Sends a stream of data from the AP to the device.

10.11.1 Greybus Raw Message Types

Table 10.147 describes the Greybus Raw operation types and their values. A message type consists of an operation type combined with a flag (0x80) indicating whether the operation is a request or a response.

Symbol	Brief Description	Mask Value
GB_POWER_SUPPLY_UPDATE	Properties Update Event	0x01
	(All other values reserved)	0x02..0x80

Table 10.146: Power Supply Protocol Event Bit Mask

Raw Operation Type	Request Value	Response Value
CPort Shutdown	0x00	0x80
Reserved	0x01	0x81
Send	0x02	0x82
(all other values reserved)	0x04..0x7e	0x84..0xfe
Invalid	0x7f	0xff

Table 10.147: Raw Operation Types

Offset	Field	Size	Value	Description
0	len	4	Number	length in bytes of the data field
4	data	*len*	Data	data to be sent

Table 10.148: Raw Send Protocol Transfer Request

10.11.2 Greybus Raw CPort Shutdown Operation

The Greybus Raw CPort Shutdown Operation is the *Common Greybus Protocol CPort Shutdown Operation* for the Raw Protocol.

10.11.3 Greybus Raw Send Operation

The Greybus Raw send operation sends data from the requester to the respondent.

Greybus Raw Send Request

Table 10.148 defines the Greybus Raw Send request. The request supplies size of the data that is sent to the device, and the data itself.

Greybus Raw Send Response

The Greybus Raw send response message has no payload.

10.12 Vibrator Protocol

This section defines the operations used on a connection implementing the Greybus vibrator Protocol. This Protocol allows an AP Module to manage a vibrator device present on a Module. The Protocol is very simple, and maps almost directly to the Android HAL vibrator interface.

The operations in the Greybus vibrator Protocol are:

`int cport_shutdown(u8 phase);`
See *Common Greybus Protocol CPort Shutdown Operation*.

`int vibrator_on(void);`
Turns on the vibrator.

`int vibrator_off(void);`
Turns off the vibrator immediately.

Vibrator Operation Type	Request Value	Response Value
CPort Shutdown	0x00	0x80
Reserved	0x01	0x81
Vibrator On	0x02	0x82
Vibrator Off	0x03	0x83
(all other values reserved)	0x04..0x7e	0x84..0xfe
Invalid	0x7f	0xff

Table 10.149: Vibrator Operation Types

10.12.1 Greybus Vibrator Message Types

Table 10.149 describes the Greybus vibrator operation types and their values. A message type consists of an operation type combined with a flag (0x80) indicating whether the operation is a request or a response.

10.12.2 Greybus Vibrator CPort Shutdown Operation

The Greybus Vibrator CPort Shutdown Operation is the *Common Greybus Protocol CPort Shutdown Operation* for the Vibrator Protocol.

10.12.3 Greybus Vibrator On Operation

The Greybus vibrator on operation allows the AP Module to request the vibrator be enabled.

Greybus Vibrator On Request

The Greybus vibrator on request message has no payload.

Greybus Vibrator On Response

The Greybus vibrator on response message has no payload.

10.12.4 Greybus Vibrator Off Operation

The Greybus Vibrator off operation allows the AP Module to request the vibrator be turned off as soon as possible.

Greybus Vibrator Off Request

The Greybus vibrator off request message has no payload.

Greybus Vibrator Off Response

The Greybus vibrator off response message has no payload.

Chapter 11

Bridged PHY Connection Protocols

This section defines a group of Protocols whose purpose is to support communication with Modules on the Greybus network which do not comply with an existing device class Protocol, and which include integrated circuits using alternative physical interfaces to UniPro. Modules which implement any of the Protocols defined in this section are said to be *non-device class conformant*.

11.1 USB Protocol

We support bulk, control, and interrupt transfers, but not isochronous at this point in time.

Details TBD.

11.2 GPIO Protocol

A connection using the GPIO Protocol on a UniPro network is used to manage a simple GPIO controller. Such a GPIO controller implements from one to 256 GPIO lines. Each of the operations defined below specifies the line to which the operation applies.

Conceptually, the GPIO Protocol operations are:

`int cport_shutdown(u8 phase);`
: See *Common Greybus Protocol CPort Shutdown Operation*.

`int line_count(u8 *count);`
: Returns one less than the number of lines managed by the Greybus GPIO controller. This means the minimum number of lines is 1 and the maximum is 256.

`int activate(u8 which);`
: Notifies the GPIO controller that one of its lines has been assigned for use.

`int deactivate(u8 which);`
: Notifies the GPIO controller that a previously activated line has been unassigned and can be deactivated.

`int get_direction(u8 which, u8 *direction);`
: Requests the GPIO controller return a line's configured direction (0 for output, 1 for input).

`int direction_input(u8 which);`
: Requests the GPIO controller configure a line for input.

GPIO Operation Type	Request Value	Response Value
CPort Shutdown	0x00	0x80
Reserved	0x01	0x81
Line Count	0x02	0x82
Activate	0x03	0x83
Deactivate	0x04	0x84
Get Direction	0x05	0x85
Direction Input	0x06	0x86
Direction Output	0x07	0x87
Get	0x08	0x88
Set	0x09	0x89
Set Debounce	0x0a	0x8a
IRQ Type	0x0b	0x8b
IRQ Mask	0x0c	0x8c
IRQ Unmask	0x0d	0x8d
IRQ Event	0x0e	N/A
(all other values reserved)	0x0f..0x7e	0x8f..0xfe
Invalid	0x7f	0xff

Table 11.1: GPIO Operation Types

`int direction_output(u8 which, u8 value);`
Requests the GPIO controller configure a line for output, and sets its initial output value (0 for low, 1 for high).

`int get_value(u8 which, u8 *value);`
Requests the GPIO controller return the current value sensed on a line (0 for low, 1 for high).

`int set_value(u8 which, u8 value);`
Requests the GPIO controller set the value (0 for low, 1 for high) for a line configured for output.

`int set_debounce(u8 which, u16 usec);`
Requests the GPIO controller set the debounce period (in microseconds).

`int irq_type(u8 which, u8 type);`
Requests the GPIO controller set the IRQ trigger type (none, falling/rising edge, or low/high level).

`int irq_mask(u8 which);`
Requests the GPIO controller mask the specified gpio irq line.

`int irq_unmask(u8 which);`
Requests the GPIO controller unmask the specified gpio irq line.

`void irq_event(u8 which);`
GPIO controller request to recipient signaling an event on the specified gpio irq line.

11.2.1 Greybus GPIO Protocol Operations

All operations sent to a GPIO controller are contained within a Greybus GPIO request message. Every operation request results in a matching response from the GPIO controller, also taking the form of a GPIO controller message. The request and response messages for each GPIO operation are defined below.

Table 11.1 defines the Greybus GPIO Protocol operation types and their values. Both the request type and response type values are shown.

Offset	Field	Size	Value	Description
0	count	1	Number	Number of GPIO lines minus 1

Table 11.2: GPIO Protocol Line Count Response

Offset	Field	Size	Value	Description
0	which	1	Number	Controller-relative GPIO line number

Table 11.3: GPIO Protocol Activate Request

11.2.2 Greybus GPIO CPort Shutdown Operation

The Greybus GPIO CPort Shutdown Operation is the *Common Greybus Protocol CPort Shutdown Operation* for the GPIO Protocol.

11.2.3 Greybus GPIO Line Count Operation

The Greybus GPIO line count operation allows the requestor to determine how many GPIO lines are implemented by the GPIO controller.

Greybus GPIO Line Count Request

The Greybus GPIO line count request message has no payload.

Greybus GPIO Line Count Response

Table 11.2 describes the Greybus GPIO line count response. The response contains a one-byte value defining the number of lines managed by the controller, minus one. That is, a count value of zero represents a single GPIO line, while a (maximal) count value of 255 represents 256 lines. GPIOs shall be numbered sequentially starting at zero.

11.2.4 Greybus GPIO Activate Operation

The Greybus GPIO activate operation notifies the GPIO controller that one of its GPIO lines has been allocated for use. This provides a chance to do initial setup for the line, such as enabling power and clock signals.

Greybus GPIO Activate Request

Table 11.3 defines the Greybus GPIO activate request. The request supplies only the number of the line to be activated.

Greybus GPIO Activate Response

The Greybus GPIO activate response message has no payload.

Offset	Field	Size	Value	Description
0	which	1	Number	Controller-relative GPIO line number

Table 11.4: GPIO Protocol Deactivate Request

Offset	Field	Size	Value	Description
0	which	1	Number	Controller-relative GPIO line number

Table 11.5: GPIO Protocol Get Direction Request

11.2.5 Greybus GPIO Deactivate Operation

The Greybus GPIO deactivate operation notifies the GPIO controller that a previously activated line is no longer in use and can be deactivated.

Greybus GPIO Deactivate Request

Table 11.4 defines the Greybus GPIO deactivate request. The request supplies only the number of the line to be deactivated.

Greybus Deactivate Response

The Greybus GPIO deactivate response message has no payload.

11.2.6 Greybus GPIO Get Direction Operation

The Greybus GPIO get direction operation requests the GPIO controller respond with the direction of transfer (in or out) for which a line is configured.

Greybus GPIO Get Direction Request

Table 11.5 defines the Greybus GPIO get direction request. The request supplies only the target line number.

Greybus GPIO Get Direction Response

Table 11.6 defines the Greybus GPIO get direction response. The response contains one byte indicating whether the line in question is configured for input or output.

Offset	Field	Size	Value	Description
0	direction	1	Number	Direction (0 for output, 1 for input)

Table 11.6: GPIO Protocol Get Direction Response

Offset	Field	Size	Value	Description
0	which	1	Number	Controller-relative GPIO line number

Table 11.7: GPIO Protocol Direction Input Request

Offset	Field	Size	Value	Description
0	which	1	Number	Controller-relative GPIO line number
1	value	1	Number	Initial value (0 is low, 1 is high)

Table 11.8: GPIO Protocol Direction Output Request

11.2.7 Greybus GPIO Direction Input Operation

The Greybus GPIO direction input operation requests the GPIO controller to configure a line to be used for input.

Greybus GPIO Direction Input Request

Table 11.7 defines the Greybus GPIO direction input request. The request supplies only the number of the line.

Greybus GPIO Direction Input Response

The Greybus GPIO direction input response message has no payload.

11.2.8 Greybus GPIO Direction Output Operation

The Greybus GPIO direction output operation requests the GPIO controller to configure a line to be used for output, and specifies its initial value.

Greybus GPIO Direction Output Request

Table 11.8 defines the Greybus GPIO direction output request. The request supplies the number of the line and its initial value.

Greybus GPIO Direction Output Response

The Greybus GPIO direction output response message has no payload.

11.2.9 Greybus GPIO Get Operation

The Greybus GPIO get operation requests the GPIO controller respond with the current value (high or low) on a line.

Greybus GPIO Get Request

Table 11.9 defines the Greybus GPIO get request. The request supplies only the target line number.

Offset	Field	Size	Value	Description
0	which	1	Number	Controller-relative GPIO line number

Table 11.9: GPIO Protocol Get Request

Offset	Field	Size	Value	Description
0	value	1	Number	Value (0 is low, 1 is high)

Table 11.10: GPIO Protocol Get Response

Greybus GPIO Get Response

Table 11.10 defines the Greybus GPIO get response. The response contains one byte indicating the value on the line in question.

11.2.10 Greybus GPIO Set Operation

The Greybus GPIO set operation requests the GPIO controller to set a line configured to be used for output to have either a low or high value.

Greybus GPIO Set Request

Table 11.11 defines the Greybus GPIO set request. The request supplies the number of the line and the value to be set.

Todo

Possibly make this a mask to allow multiple values to be set at once.

Greybus GPIO Set Response

The Greybus GPIO set response message has no payload.

11.2.11 Greybus GPIO Set Debounce Operation

The Greybus GPIO set debounce operation requests the GPIO controller to set the debounce delay configured to be used for a line.

Offset	Field	Size	Value	Description
0	which	1	Number	Controller-relative GPIO line number
1	value	1	Number	Initial value (0 is low, 1 is high)

Table 11.11: GPIO Protocol Set Request

Offset	Field	Size	Value	Description
0	which	1	Number	Controller-relative GPIO line number
1	usec	2	Number	Debounce period (microseconds)

Table 11.12: GPIO Protocol Set Debounce Request

Offset	Field	Size	Value	Description
0	which	1	Number	Controller-relative GPIO line number
1	type	1	Number	*Greybus GPIO IRQ Type Bits*

Table 11.13: GPIO IRQ Type Request

Greybus GPIO Set Debounce Request

Table 11.12 defines the Greybus GPIO set debounce request. The request supplies the number of the line and the time period (in microseconds) to be used for the line. If the period specified is 0, debounce is disabled.

Greybus GPIO Set Debounce Response

The Greybus GPIO set debounce response message has no payload.

11.2.12 Greybus GPIO IRQ Type Operation

The Greybus GPIO IRQ type operation requests the GPIO controller to set the interrupt trigger type to be used for a line.

Greybus GPIO IRQ Type Request

Table 11.13 defines the Greybus GPIO IRQ type request. This request supplies the number of the line and the type to be used for the line.

Greybus GPIO IRQ Type Bits

Table 11.14 describes the defined interrupt trigger type bit values defined for Greybus GPIO IRQ chips. Only the listed trigger type values are valid.

Symbol	Brief Description	Value
IRQ_TYPE_NONE	No trigger specified, uses default/previous setting	0x00
IRQ_TYPE_EDGE_RISING	Rising edge triggered	0x01
IRQ_TYPE_EDGE_FALLING	Falling edge triggered	0x02
IRQ_TYPE_EDGE_BOTH	Rising and falling edge triggered	0x03
IRQ_TYPE_LEVEL_HIGH	Level triggered high	0x04
IRQ_TYPE_LEVEL_LOW	Level triggered low	0x08
	(All other values reserved)	0x10..0xff

Table 11.14: GPIO IRQ Type Bits

Offset	Field	Size	Value	Description
0	which	1	Number	Controller-relative GPIO line number

Table 11.15: GPIO IRQ Mask Request

Offset	Field	Size	Value	Description
0	which	1	Number	Controller-relative GPIO line number

Table 11.16: GPIO IRQ Unmask Request

Greybus GPIO IRQ Type Response

The Greybus GPIO IRQ type response message has no payload.

11.2.13 Greybus GPIO IRQ Mask Operation

The Greybus GPIO IRQ mask operation requests the GPIO controller to mask a GPIO IRQ line.

Greybus GPIO IRQ Mask Request

Table 11.15 defines the Greybus GPIO IRQ mask request. This request supplies the number of the line to be masked.

Greybus GPIO IRQ Mask Response

The Greybus GPIO IRQ mask response message has no payload.

11.2.14 Greybus GPIO IRQ Unmask Operation

The Greybus GPIO IRQ unmask operation requests the GPIO controller to unmask a GPIO IRQ line.

Greybus GPIO IRQ Unmask Request

Table 11.16 defines the Greybus GPIO IRQ unmask request. This request supplies the number of the line to be unmasked.

Greybus GPIO IRQ Unmask Response

The Greybus GPIO IRQ unmask response message has no payload.

11.2.15 Greybus GPIO IRQ Event Operation

The Greybus GPIO IRQ event operation signals to the recipient that a GPIO IRQ event has occurred on the GPIO Controller.

The GPIO controller is responsible for masking the interrupt before sending the event.

Note that the GPIO IRQ event operation is unidirectional and has no response.

Offset	Field	Size	Value	Description
0	which	1	Number	Controller-relative GPIO line number

Table 11.17: GPIO IRQ Event Request

SPI Operation Type	Request Value	Response Value
CPort Shutdown	0x00	0x80
Reserved	0x01	0x81
Master Config	0x02	0x82
Device Config	0x03	0x83
Transfer	0x04	0x84
(all other values reserved)	0x05..0x7e	0x85..0xfe
Invalid	0x7f	0xff

Table 11.18: SPI Protocol Operation Types

Greybus GPIO IRQ Event Request

Table 11.17 defines the Greybus GPIO IRQ Event request. This request supplies the number of the line signaling an event.

11.3 SPI Protocol

This section defines the operations used on a connection implementing the Greybus SPI Protocol. This Protocol allows for management of a SPI device. The Protocol consists of the operations defined in this section.

Conceptually, the operations in the Greybus SPI Protocol are:

`int cport_shutdown(u8 phase);`
See *Common Greybus Protocol CPort Shutdown Operation*.

`int master_config(u16 *mode, u16 *flags, u32 *bpw_mask, u16 *num_chipselect, u32 *min_speed_hz, u32 *max_`
Returns a set of configuration parameters related to SPI master.

`int device_config(u16 cs, u16 *mode, u8 *bpw, u32 *max_speed_hz, u8 *device_type, u8 *name[32]);`
Returns a set of configuration parameters related to SPI device in a chipselect.

`int transfer(u8 chip_select, u8 mode, u8 count, struct gb_spi_transfer *transfers);`
Performs a SPI transaction as one or more SPI transfers, defined in the supplied array.

A transfer is made up of an array of *gb_spi_transfer* descriptors, each of which specifies SPI master configurations during transfers. For write requests, the data is sent following the array of messages; for read requests, the data is returned in a response message from the SPI master.

11.3.1 Greybus SPI Message Types

Table 11.18 defines the Greybus SPI operation types and their values. A message type consists of an operation type combined with a flag (0x80) indicating whether the operation is a request or a response.

Offset	Field	Size	Value	Description
0	bpw_mask	4	Bit Mask	:ref:'spi-bpw-mask‘
4	min_speed_hz	4	Number	Lower limit for transfer speed
8	max_speed_hz	4	Number	Higher limit for transfer speed
10	mode	2	Bit Mask	*Greybus SPI Protocol Mode Bit Masks*
12	flags	2	Bit Mask	*Greybus SPI Protocol Flags Bit Masks*
14	num_chipselect	1	Number	Maximum chipselect supported by Master

Table 11.19: SPI Protocol Master Config Response

Symbol	Brief Description	Mask Value
GB_SPI_MODE_CPHA	Clock phase (0: sample on first clock, 1: on second)	0x0001
GB_SPI_MODE_CPOL	Clock polarity (0: clock low on idle, 1: high on idle)	0x0002
GB_SPI_MODE_CS_HIGH	Chip select active high	0x0004
GB_SPI_MODE_LSB_FIRST	Per-word bits-on-wire	0x0008
GB_SPI_MODE_3WIRE	SI/SO signals shared	0x0010
GB_SPI_MODE_LOOP	Loopback mode	0x0020
GB_SPI_MODE_NO_CS	One dev/bus, no chip select	0x0040
GB_SPI_MODE_READY	Slave pulls low to pause	0x0080
	(All other mask values reserved)	0x0100..0x8000

Table 11.20: SPI Protocol Mode Bit Masks

11.3.2 Greybus SPI CPort Shutdown Operation

The Greybus SPI CPort Shutdown Operation is the *Common Greybus Protocol CPort Shutdown Operation* for the SPI Protocol.

11.3.3 Greybus SPI Protocol Master Config Operation

The Greybus SPI Master Config operation allows the requestor to determine the details of the configuration parameters by the SPI master. This operation can be executed at any time. All other operations should be discarded until the successful execution of this one.

Greybus SPI Protocol Master Config Request

The Greybus SPI Master Config request message has no payload.

Greybus SPI Protocol Master Config Response

Table 11.19 defines the Greybus SPI Master Config response. The response contains a set of values representing the support, limits and default values of certain configurations.

Greybus SPI Protocol Mode Bit Masks

Table 11.20 defines the mode bit masks for Greybus SPI masters.

Symbol	Brief Description	Mask Value
GB_SPI_FLAG_HALF_DUPLEX	Can't do full duplex	0x0001
GB_SPI_FLAG_NO_RX	Can't do buffer read	0x0002
GB_SPI_FLAG_NO_TX	Can't do buffer write	0x0004
	(All other flag values reserved)	0x0008..0x8000

Table 11.21: SPI Protocol Flags

Offset	Field	Size	Value	Description
0	chip_select	1	Number	Chip Select Number

Table 11.22: SPI Device Config Request

Greybus SPI Protocol Bits Per Word Mask

The Greybus SPI bits per word mask allows the requestor to determine the mask indicating which values of bits_per_word are supported by the SPI master. If set, transfer with unsupported bits_per_word should be rejected. If not set, this value is simply ignored, and it's up to the individual driver to perform any validation.

Transfers should be rejected if following expression evaluates to zero:

```
master->bits_per_word_mask & (1 << (tx_desc->bits_per_word - 1))
```

Greybus SPI Protocol Flags Bit Masks

Table 11.21 describes the defined flags bit masks defined for Greybus SPI masters.

11.3.4 Greybus SPI Protocol Device Config Operation

The Greybus SPI Device Config operation allows the requestor to determine the details of the configuration parameters of a access-enable device. This operation can be executed at any time, however it shall be executed after the the Master Config Operation for each chipselect till the number given by the num_chipselect in the Master Config Response. All transfer operations for the device should be discarded until the successful execution of this operation.

Greybus SPI Protocol Device Config Request

Table 11.22 describes the Greybus SPI Device Config request. The request supplies the chip_select which is a unique identifier between 0 and num_chipselect.

Greybus SPI Protocol Device Config Response

Table 11.23 defines the Greybus SPI Device Config response. The response contains a set of values representing the limits and default values of certain configurations of a device.

Offset	Field	Size	Value	Description
0	mode	2	Bit Mask	*Greybus SPI Protocol Mode Bit Masks*
2	bpw	1	Number	bits per word supported by device
3	max_speed_hz	4	Number	Higher limit for transfer speed
7	device_type	1	Number	*Greybus SPI Protocol Device Type*
8	name	32	UTF-8	Name and/or Device driver alias

Table 11.23: SPI Protocol Device Config Response

Symbol	Brief Description	Value
GB_SPI_SPI_DEV	SPI device is a generic bit bang SPI device	0x00
GB_SPI_SPI_NOR	SPI device is a SPI NOR device that supports JEDEC id	0x01
GB_SPI_SPI_MODALIAS	SPI device driver can be represented by the name field	0x02
	(All other values reserved)	0x03..0xFF

Table 11.24: SPI Protocol Device Type Values

Greybus SPI Protocol Device Type

Table 11.24 defines the types of device associated with asked chip-select for Greybus SPI devices. The name field in *Greybus SPI Protocol Device Config Response* shall be ignore if the Device Type is not equal to GB_SPI_SPI_MODALIAS.

11.3.5 Greybus SPI Transfer Operation

The Greybus SPI transfer operation requests that the SPI master perform a SPI transaction. The operation consists of a set of one or more *gb_spi_transfer* descriptors, which define data transfers to be performed by the SPI master. The transfer operation request includes data for each *gb_spi_transfer* descriptor involving a write operation. The data shall be sent immediately following the *gb_spi_transfer* descriptors (with no intervening pad bytes). The transfer operation response includes data for each *gb_spi_transfer* descriptor involving a read operation, with all read data transferred contiguously.

Greybus SPI Transfer Request

The Greybus SPI transfer request contains the slave's chip select pin, its mode, a count of message descriptors, an array of message descriptors, and a block of zero or more bytes of data to be written. Table 11.25 defines the **Greybus SPI gb_spi_transfer descriptor**. This describes the configuration of a segment of a SPI transaction.

Table 11.26 defines the Greybus SPI transfer request.

Offset	Field	Size	Value	Description
0	speed_hz	4	Number	Transfer speed in Hz
4	len	4	Number	Size of data to transfer
8	delay_usecs	2	Number	Wait period after completion of transfer
10	cs_change	1	Number	Toggle chip select pin after this transfer completes
11	bits_per_word	1	Number	Select bits per word for this transfer
12	xfer_flags	1	Bit Mask	*Greybus SPI Transfer Flags Bits*

Table 11.25: SPI Protocol gb_spi_transfer descriptor

Offset	Field	Size	Value	Description
0	chip-select	1	Number	chip-select pin for the slave device
1	mode	1	Number	*Greybus SPI Protocol Mode Bit Masks*
2	count	2	Number	Number of *gb_spi_transfer* descriptors
4	op[1]	13	Structure	First SPI *gb_spi_transfer* descriptor in the transfer
...	...	13	Structure	...
4+13*(N-1)	op[N]	13	Structure	Last SPI *gb_spi_transfer* descriptor
4+13*N	data	...	Data	Data for all the write transfers

Table 11.26: SPI Protocol Transfer Request

Symbol	Brief Description	Mask Value
GB_SPI_XFER_READ	Read Transfer Descriptor	0x01
GB_SPI_XFER_WRITE	Write Transfer Descriptor	0x02
GB_SPI_XFER_INPROGRESS	Indicate current operation will continue in next transfer	0x04
	(All other values reserved)	0x08..0x80

Table 11.27: SPI Transfer Flags Bits

Any data to be written follows the last *gb_spi_transfer* descriptor. Data for the first write *gb_spi_transfer* descriptor in the array immediately follows the last *gb_spi_transfer* descriptor in the array, and no padding shall be inserted between data sent for distinct SPI *gb_spi_transfer* descriptors.

Greybus SPI Transfer Flags Bits

Table 11.27 describes possible transfer descriptors flags. Only the listed values are valid.

Greybus SPI Transfer Response

Table 11.28 defines the Greybus SPI transfer response. The response contains the data read as a result of the request.

11.4 UART Protocol

A connection using the UART Protocol on a UniPro network is used to manage a simple UART controller. This Protocol is very close to the CDC protocol for serial modems from the USB-IF specification, and consists of the operations defined in this section.

The operations that can be performed on a Greybus UART controller are conceptually:

`int cport_shutdown(u8 phase);`
See *Common Greybus Protocol CPort Shutdown Operation*.

Offset	Field	Size	Value	Description
0	data		Data	Data for first read *gb_spi_transfer* descriptor on the transfer
...	...	...	Data	...
...	...	...	Data	Data for Last read *gb_spi_transfer* descriptor on the transfer

Table 11.28: SPI Protocol Transfer Response

UART Operation Type	Request Value	Response Value
CPort Shutdown	0x00	0x80
Reserved	0x01	0x81
Send Data	0x02	0x82
Receive Data	0x03	0x83
Set Line Coding	0x04	0x84
Set Control Line State	0x05	0x85
Send Break	0x06	0x86
Serial State	0x07	0x87
Receive Credits	0x08	0x88
Flush	0x09	0x89
(all other values reserved)	0x10..0x7e	0x90..0xfe
Invalid	0x7f	0xff

Table 11.29: UART Operation Types

`int send_data(u16 size, u8 *data);`
Requests that the UART device begins transmitting characters. One or more bytes to be transmitted shall be supplied by the sender.

`int receive_data(u16 size, u8 flags, u8 *data);`
Receive data from the UART and any line errors that might have occurred.

`int set_line_coding(u32 rate, u8 format, u8 parity, u8 data);`
Sets the line settings of the UART to the specified baud rate, format, parity, and data bits.

`int set_control_line_state(u8 state);`
Controls RTS and DTR line states of the UART.

`int send_break(u8 state);`
Requests that the UART generate a break condition on its transmit line.

`int serial_state(u8 state);`
Receives the state of the UART's control lines.

`int receive_credits(u16 count);`
Receive transmit credits form the uart controller.

`int flush_buffer(u8 flags);`
Clear the internal uart queues.

11.4.1 UART Protocol Operations

This section defines the operations for a connection using the UART Protocol. The UART Protocol allows a requestor to control a UART device contained within a Greybus Module.

Greybus UART Protocol Operations

Table 11.29 defines the Greybus UART operation types and their values. A message type consists of an operation type combined with a flag (0x80) indicating whether the operation is a request or a response.

Offset	Field	Size	Value	Description
0	size	2	Number	Size in bytes of data to be transmitted
2	data	*size*	Data	1 or more bytes of data to be transmitted

Table 11.30: UART Protocol Send Data Request

Offset	Field	Size	Value	Description
0	size	2	Number	Size in bytes of received data
2	flags	1	Bit mask	*Greybus UART Receive Data Status Flags*
3	data	*size*	Data	1 or more bytes of received data

Table 11.31: UART Protocol Receive Data Request

11.4.2 Greybus UART CPort Shutdown Operation

The Greybus UART CPort Shutdown Operation is the *Common Greybus Protocol CPort Shutdown Operation* for the UART Protocol.

11.4.3 Greybus UART Send Data Operation

The Greybus UART Send Data operation requests that the UART device begin transmission of characters. One or more characters to be transmitted may optionally be provided with this request.

Greybus UART Send Data Request

Table 11.30 defines the Greybus UART send data request. This requests that the UART device begin transmitting. The request optionally contains one or more characters to be transmitted.

Greybus UART Send Data Response

The Greybus UART send data response message has no payload.

11.4.4 Greybus UART Receive Data Operation

Unlike most other Greybus UART operations, the Greybus UART event operation is initiated by the device implementing the UART Protocol. It notifies its peer that a data has been received by the UART.

Note that the UART Receive Data Operation is unidirectional and has no response.

Greybus UART Receive Data Request

Table 11.31 defines the Greybus UART receive data request. The request contains the size of the data to be received, associated line-status flags, and the data bytes to be received. Every receive-data-request message must have a size field >= 1, with firmware inserting a NUL byte as necessary when reporting a break event. Note that overrun is special in that it is not associated with any particular character.

Flag	Value	Description
Framing Error	0x01	Framing error detected
Parity Error	0x02	Parity error detected
Overrun	0x04	Received data lost due to overrun
Break	0x08	Break condition detected
(all other values reserved)	0x10..0x80	

Table 11.32: UART Modem Receive Data Status Flags

Offset	Field	Size	Value	Description
0	rate	4	Number	Baud Rate setting
4	format	1	Number	*Greybus UART Stop Bit Format*
5	parity	1	Number	*Greybus UART Parity format*
6	data_bits	1	Number	Number of data bits
7	flow_control	1	Bit mask	*Greybus UART Flow Control Flags*

Table 11.33: UART Protocol Set Line Coding State Request

Greybus UART Receive Data Status Flags

Table 11.32 defines the values supplied as flag values for the Greybus UART receive data request. Any combination of these values may be supplied in a single request.

11.4.5 Greybus UART Set Line Coding Operation

The Greybus UART set line coding operation allows for configuration of the UART to a specific set of line coding values.

Greybus UART Set Line Coding State Request

Table 11.33 defines the Greybus UART set line coding state request. The request contains the specific line coding values to be set.

Greybus UART Stop Bit Format

Table 11.34 defines the Greybus UART stop bit formats.

Greybus UART Parity format

Table 11.35 defines the Greybus UART parity formats.

1 Stop Bit	0x00
1.5 Stop Bits	0x01
2 Stop Bits	0x02
(All other values reserved)	0x03..0xff

Table 11.34: UART Protocol Stop Bit Format

No Parity	0x00
Odd Parity	0x01
Even Parity	0x02
Mark Parity	0x03
Space Parity	0x04
(All other values reserved)	0x05..0xff

Table 11.35: UART Protocol Parity Format

Flag	Value	Description
AUTO_RTSCTS	0x01	Enable automatic rts/cts
(all other values reserved)	0x02..0x80	

Table 11.36: UART Protocol Flow Control Flags

Greybus UART Flow Control Flags

Table 11.36 defines the Greybus UART flow control bit mask.

Greybus UART Set Line Coding State Response

The Greybus UART set line coding state response message has no payload.

11.4.6 Greybus UART Set Control Line State Operation

The Greybus UART set control line state operation requests that the UART device set "outbound" UART status values.

Greybus UART Set Control Line State Request

Table 11.37 defines the Greybus UART set control line state request. The request contains a bit mask of modem status flags to set.

Greybus UART Modem Status Flags

Table 11.38 defines the values supplied as flag values for the Greybus UART set control line state request. Any combination of these values may be supplied in a single request.

Greybus UART Set Control Line State Response

The Greybus UART set control line state response message has no payload.

Offset	Field	Size	Value	Description
0	control	1	Bit mask	*Greybus UART Modem Status Flags*

Table 11.37: UART Protocol Set Control Line State Request

Flag	Value	Description
DTR	0x01	Data Terminal Ready
RTS	0x02	Request To Send
(all other values reserved)	0x04..0x80	

Table 11.38: UART Modem Status Flags

Offset	Field	Size	Value	Description
0	state	1	Number	0 is off, 1 is on

Table 11.39: UART Protocol Break Control Request

11.4.7 Greybus UART Send Break Operation

The Greybus UART send break operation requests that the UART device set the break condition on its transmit line to be either on or off.

Greybus UART Break Control Request

Table 11.39 defines the Greybus UART break control request. The requestq supplies the duration of the break condition that should be generated by the UART device transmit line.

Greybus UART Break Control Response

The Greybus UART break control response message has no payload.

11.4.8 Greybus UART Serial State Operation

Unlike most other Greybus UART operations, the Greybus UART serial state operation is initiated by the Module implementing the UART Protocol. It notifies the peer that a control line status has changed, or that there is an error with the UART.

Note that the UART Serial State Operation is unidirectional and has no response.

Greybus UART Serial State Request

Table 11.40 defines the Greybus UART serial state request. The request contains the control value that the UART is currently in.

Greybus UART Control Flags

Table 11.41 defines the flag values used for a Greybus UART serial state request.

Offset	Field	Size	Value	Description
0	control	1	Bit mask	*Greybus UART Control Flags*

Table 11.40: UART Protocol Serial State Request

Flag	Value	Description
DCD	0x01	Carrier Detect line enabled
DSR	0x02	DSR signal
RI	0x04	Ring Signal detected
(all other values reserved)	0x08..0x80	

Table 11.41: UART Control Flags

Offset	Field	Size	Value	Description
0	count	2	Number	number of credits

Table 11.42: UART Protocol Receive Credits Request

11.4.9 Greybus UART Receive Credits Operation

The Greybus UART receive credits operation is initiated by the Module implementing the UART Protocol. It notifies the peer that output data has been processed and more space has been made available in its internal output buffers. For optimal performance, and when suitable, the module should aggregate credits.

Note that the UART receive credits operation is unidirectional and has no response.

Greybus UART Receive Credits Request

Table 11.42 defines the Greybus UART receive credits request. The request contains the amount of credits that has been processed and that is now available to the peer for more output data.

11.4.10 Greybus UART FLUSH Operation

The Greybus UART Flush operation requests the UART device to discard data that has been stored internally and is waiting to be processed.

Greybus UART Flush Request

Table 11.43 defines the Greybus UART flush request. This requests the UART device to discard data that has been queued.

Greybus UART Flush Flags

Table 11.44 defines the Greybus UART flush flags bit mask.

Greybus UART Flush Response

The Greybus UART flush response message has no payload.

Offset	Field	Size	Value	Description
0	flags	1	Bit mask	*Greybus UART Flush Flags*

Table 11.43: UART Protocol Flush Request

Flag	Value	Description
FLUSH_TRANSMITTER	0x01	Flush transmitter queue(s)
FLUSH_RECEIVER	0x02	Flush receiver queue(s)
(all other values reserved)	0x04..0x80	

Table 11.44: UART Protocol Flush Flags

11.5 PWM Protocol

A connection using PWM Protocol on a UniPro network is used to manage a simple PWM controller. Such a PWM controller implements one or more (up to 256) PWM devices, and each of the operations below specifies the line to which the operation applies. This Protocol consists of the operations defined in this section.

Conceptually, the PWM Protocol operations are:

`int cport_shutdown(u8 phase);`
See *Common Greybus Protocol CPort Shutdown Operation*.

`int pwm_count(u8 *count);`
Returns one less than the number of instances managed by the Greybus PWM controller. This means the minimum number of PWMs is 1 and the maximum is 256.

`int activate(u8 which);`
Notifies the PWM controller that one of its instances has been assigned for use.

`int deactivate(u8 which);`
Notifies the PWM controller that a previously activated instance has been unassigned and can be deactivated.

`int config(u8 which, u32 duty, u32 period);`
Requests the PWM controller configure an instance for a particular duty cycle and period (in units of nanoseconds).

`int set_polarity(u8 which, u8 polarity);`
Requests the PWM controller configure an instance as normally active or inverted.

`int enable(u8 which);`
Requests the PWM controller enable a PWM instance to begin toggling.

`int disable(u8 which);`
Requests the PWM controller disable a previously enabled PWM instance

11.5.1 Greybus PWM Protocol Operations

All operations sent to a PWM controller are contained within a Greybus PWM request message. Every operation request results in a response from the PWM controller, also taking the form of a PWM controller message. The request and response messages for each PWM operation are defined below.

Table 11.45 describes the Greybus PWM Protocol operation types and their values. Both the request type and response type values are shown.

PWM Operation Type	Request Value	Response Value
CPort Shutdown	0x00	0x80
Reserved	0x01	0x81
PWM count	0x02	0x82
Activate	0x03	0x83
Deactivate	0x04	0x84
Config	0x05	0x85
Set Polarity	0x06	0x86
Enable	0x07	0x87
Disable	0x08	0x88
(all other values reserved)	0x09..0x7e	0x89..0xfe
Invalid	0x7f	0xff

Table 11.45: PWM Operation Types

Offset	Field	Size	Value	Description
0	count	1	Number	Number of PWM instances minus 1

Table 11.46: PWM Protocol Count Response

11.5.2 Greybus PWM CPort Shutdown Operation

The Greybus PWM CPort Shutdown Operation is the *Common Greybus Protocol CPort Shutdown Operation* for the PWM Protocol.

11.5.3 Greybus PWM Count Operation

The Greybus PWM count operation allows the requestor to determine how many PWM instances are implemented by the PWM controller.

Greybus PWM Count Request

The Greybus PWM count request message has no payload.

Greybus PWM Count Response

Table 11.46 defines the Greybus PWM count response. The response contains a one-byte value defining the number of PWM instances managed by the controller, minus one. That is, a count value of zero represents a single PWM instance, while a (maximal) count value of 255 represents 256 instances. The lines are numbered sequentially starting at zero.

11.5.4 Greybus PWM Activate Operation

The Greybus PWM activate operation notifies the PWM controller that one of its PWM instances has been allocated for use. This provides a chance to do initial setup for the PWM instance, such as enabling power and clock signals.

Offset	Field	Size	Value	Description
0	which	1	Number	Controller-relative PWM instance number

Table 11.47: PWM Protocol Activate Request

Offset	Field	Size	Value	Description
0	which	1	Number	Controller-relative PWM instance number

Table 11.48: PWM Protocol Deactivate Request

Greybus PWM Activate Request

Table 11.47 defines the Greybus PWM activate request. The request supplies only the number of the instance to be activated.

Greybus PWM Activate Response

The Greybus PWM activate response message has no payload.

11.5.5 Greybus PWM Deactivate Operation

The Greybus PWM instance deactivate operation notifies the PWM controller that a previously activated instance is no longer in use and can be deactivated.

Greybus PWM Deactivate Request

Table 11.48 defines the Greybus PWM deactivate request. The request supplies only the number of the instance to be deactivated.

Greybus PWM Deactivate Response

The Greybus PWM deactivate response message has no payload.

11.5.6 Greybus PWM Configure Operation

The Greybus PWM configure operation requests the PWM controller configure a PWM instance with the given duty cycle and period.

Greybus PWM Configure Request

Table 11.49 defines the Greybus PWM configure request. The request supplies the target instance number, duty cycle, and period of the cycle.

Greybus PWM Configure Response

The Greybus PWM configure response message has no payload.

Offset	Field	Size	Value	Description
0	which	1	Number	Controller-relative PWM instance number
1	duty	4	Number	Duty cycle (in nanoseconds)
5	period	4	Number	Period (in nanoseconds)

Table 11.49: PWM Protocol Configure Request

Offset	Field	Size	Value	Description
0	which	1	Number	Controller-relative PWM instance number
1	polarity	1	Number	0 for normal, 1 for inverted

Table 11.50: PWM Protocol Polarity Request

11.5.7 Greybus PWM Polarity Operation

The Greybus PWM polarity operation requests the PWM controller configure a PWM instance with the given polarity.

Greybus PWM Polarity Request

Table 11.50 defines the Greybus PWM polarity request. The request supplies the target instance number and polarity (normal or inverted). The polarity may not be configured when a PWM instance is enabled.

Greybus PWM Polarity Response

The Greybus PWM polarity response message has no payload.

11.5.8 Greybus PWM Enable Operation

The Greybus PWM enable operation enables a PWM instance to begin toggling.

Greybus PWM Enable Request

Table 11.51 defines the Greybus PWM enable request. The request supplies only the number of the instance to be enabled.

Greybus PWM Enable Response

The Greybus PWM enable response message has no payload.

Offset	Field	Size	Value	Description
0	which	1	Number	Controller-relative PWM instance number

Table 11.51: PWM Protocol Enable Request

Offset	Field	Size	Value	Description
0	which	1	Number	Controller-relative PWM instance number

Table 11.52: PWM Protocol Disable Request

11.5.9 Greybus PWM Disable Operation

The Greybus PWM disable operation stops a PWM instance that has previously been enabled.

Greybus PWM Disable Request

Table 11.52 defines the Greybus PWM disable request. The request supplies only the number of the instance to be disabled.

Greybus PWM Disable Response

The Greybus PWM disable response message has no payload.

11.6 I2C Protocol

This section defines the operations used on a connection implementing the Greybus I2C Protocol. This Protocol allows for management of an I2C device present on a Module. The Protocol consists of five basic operations, whose request and response message formats are defined here.

Conceptually, the five operations in the Greybus I2C Protocol are:

`int cport_shutdown(u8 phase);`
See *Common Greybus Protocol CPort Shutdown Operation*.

`int get_functionality(u32 *functionality);`
Returns a bitmask indicating the features supported by the I2C adapter.

`int transfer(u8 op_count, struct i2c_op *ops);`
Performs an I2C transaction made up of one or more "steps" defined in the supplied I2C op array.

A transfer is made up of an array of "I2C ops", each of which specifies an I2C slave address, flags controlling message behavior, and a length of data to be transferred. For write requests, the data is sent following the array of messages; for read requests, the data is returned in a response message from the I2C adapter.

11.6.1 Greybus I2C Message Types

Table 11.53 defines the Greybus I2C operation types and their values. A message type consists of an operation type combined with a flag (0x80) indicating whether the operation is a request or a response.

11.6.2 Greybus I2C CPort Shutdown Operation

The Greybus I2C CPort Shutdown Operation is the *Common Greybus Protocol CPort Shutdown Operation* for the I2C Protocol.

I2C Operation Type	Request Value	Response Value
CPort Shutdown	0x00	0x80
Reserved	0x01	0x81
Functionality	0x02	0x82
Reserved	0x03	0x83
Reserved	0x04	0x84
Transfer	0x05	0x85
(all other values reserved)	0x06..0x7e	0x86..0xfe
Invalid	0x7f	0xff

Table 11.53: I2C Operation Types

Offset	Field	Size	Value	Description
0	functionality	4	Number	*Greybus I2C Functionality Bit Masks*

Table 11.54: I2C Protocol Functionality Response

11.6.3 Greybus I2C Functionality Operation

The Greybus I2C functionality operation allows the requestor to determine the details of the functionality provided by the I2C adapter.

Greybus I2C Functionality Request

The Greybus I2C functionality request message has no payload.

Greybus I2C Functionality Response

Table 11.54 defines the Greybus I2C functionality response. The response contains a four-byte value whose bits represent support or presence of certain functionality in the I2C adapter.

Greybus I2C Functionality Bit Masks

Table 11.55 defines the functionality bit masks for Greybus I2C adapters. These include a set of bits describing SMBus capabilities. These values are taken directly from the <linux/i2c.h> header file.

11.6.4 Greybus I2C Transfer Operation

The Greybus I2C transfer operation requests that the I2C adapter perform an I2C transaction. The operation consists of a set of one or more "I2C ops" to be performed by the I2C adapter. The transfer operation request includes data for each I2C op involving a write operation. The data is concatenated (without padding) and is sent immediately after the set of I2C op descriptors. The transfer operation response includes data for each I2C op involving a read operation, with all read data transferred contiguously.

Greybus I2C Transfer Request

The Greybus I2C transfer request contains a message count, an array of message descriptors, and a block of zero or more bytes of data to be written.

Linux Symbol	Brief Description	Mask Value
I2C_FUNC_I2C	Basic I2C protocol (not SMBus) support	0x00000001
I2C_FUNC_10BIT_ADDR	10-bit addressing is supported	0x00000002
	(Reserved)	0x00000004
I2C_FUNC_SMBUS_PEC	SMBus CRC-8 byte added to transfers (PEC)	0x00000008
I2C_FUNC_NOSTART	Repeated start sequence can be skipped	0x00000010
	(Reserved range)	0x00000020..0x00004000
I2C_FUNC_SMBUS_BLOCK_PROC_CALL	SMBus block write-block read process call supported	0x00008000
I2C_FUNC_SMBUS_QUICK	SMBus write_quick command supported	0x00010000
I2C_FUNC_SMBUS_READ_BYTE	SMBus read_byte command supported	0x00020000
I2C_FUNC_SMBUS_WRITE_BYTE	SMBus write_byte command supported	0x00040000
I2C_FUNC_SMBUS_READ_BYTE_DATA	SMBus read_byte_data command supported	0x00080000
I2C_FUNC_SMBUS_WRITE_BYTE_DATA	SMBus write_byte_data command supported	0x00100000
I2C_FUNC_SMBUS_READ_WORD_DATA	SMBus read_word_data command supported	0x00200000
I2C_FUNC_SMBUS_WRITE_WORD_DATA	SMBus write_word_data command supported	0x00400000
I2C_FUNC_SMBUS_PROC_CALL	SMBus process_call command supported	0x00800000
I2C_FUNC_SMBUS_READ_BLOCK_DATA	SMBus read_block_data command supported	0x01000000
I2C_FUNC_SMBUS_WRITE_BLOCK_DATA	SMBus write_block_data command supported	0x02000000
I2C_FUNC_SMBUS_READ_I2C_BLOCK	SMBus read_i2c_block_data command supported	0x04000000
I2C_FUNC_SMBUS_WRITE_I2C_BLOCK	SMBus write_i2c_block_data command supported	0x08000000
	(All other values reserved)	0x10000000..0x80000000

Table 11.55: I2C Functionality Bit Masks

Offset	Field	Size	Value	Description
0	addr	2	Number	Slave address
2	flags	2	Number	*Greybus I2C Op Flag Bit Masks*
4	size	2	Number	Size of data to transfer

Table 11.56: I2C Op

Table 11.56 defines the **Greybus I2C op**. An I2C op describes a segment of an I2C transaction.

Greybus I2C Op Flag Bit Masks

Table 11.57 defines the defined flag bit masks defined for Greybus I2C ops. They are taken directly from the <linux/i2c.h> header file.

Table 11.58 defines the Greybus I2C transfer request.

Any data to be written follows the last op descriptor. Data for the first write op in the array immediately follows the last op in the array, and no padding shall be inserted between data sent for distinct I2C ops.

Linux Symbol	Brief Description	Mask Value
I2C_M_RD	Data is to be read (from slave to master)	0x0001
	(Reserved range)	0x0002..0x0008
I2C_M_TEN	10-bit addressing is supported	0x0010
	(Reserved range)	0x0020..0x0200
I2C_M_RECV_LEN	First byte received contains length	0x0400
	(Reserved range)	0x0800..0x2000
I2C_M_NOSTART	Skip repeated start sequence	0x4000
	(Reserved)	0x8000

Table 11.57: I2C Protocol Op Flag Bit Masks

Offset	Field	Size	Value	Description
0	op_count	2	Number	Number of I2C ops in transfer
2	op[1]	6	Structure	Descriptor for first I2C op in the transfer
...	...	6	Structure	...
2+6*(N-1)	op[N]	6	Structure	Descriptor for last I2C op
2+6*N	data	6	Data	Data for first write op in the transfer
...	...	...	Data	Data for last write op on the transfer

Table 11.58: I2C Protocol Transfer Request

Offset	Field	Size	Value	Description
0	data		Data	Data for first read op on the transfer
...	...	...	Data	...
...	...	...	Data	Data for last read op on the transfer

Table 11.59: I2C Protocol Transfer Response

Greybus I2C Transfer Response

Table 11.59 defines the Greybus I2C transfer response. The response contains the data read as a result of messages.

11.7 SDIO Protocol

This section defines the operations used on a connection implementing the Greybus SDIO Protocol. This Protocol allows for management of a SDIO device present on a Module. The Protocol consists of operations, whose request and response message formats are defined here.

Conceptually, the operations in the Greybus SDIO Protocol are:

`int cport_shutdown(u8 phase);`
See *Common Greybus Protocol CPort Shutdown Operation*.

`int get_capabilities(u32 *caps, u32 *ocr, u16 *max_blk_count, u16 *max_blk_size);`
Request the SDIO controller to return a set of capabilities available, supported voltage ranges and maximum block count/size per data command transfer.

`int set_ios(struct gb_sdio_ios *ios);`
Request the SDIO controller to setup various parameters related with the interface.

`int command(u8 cmd, u8 cmd_flags, u8 cmd_type, u32 arg, u32 *resp[4]);`
Send a control command as specified by the SD Association and return the correspondent response.

`int transfer(u8 data_flags, u16 *data_blocks, u16 *data_blksz, u8 *data);`
Performs a SDIO data transaction defined by the size to be send/received.

`int sdio_event(u8 event);`
The SDIO controller notifies the recipient of SD card related events.

11.7.1 Greybus SDIO Protocol Operations

All operations sent to a SDIO controller are contained within a Greybus SDIO request message. Every operation request results in a matching response from the SDIO controller, also taking the form of a SDIO

SDIO Operation Type	Request Value	Response Value
CPort Shutdown	0x00	0x80
Reserved	0x01	0x81
Get Capabilities	0x02	0x82
Set Ios	0x03	0x83
Command	0x04	0x84
Transfer	0x05	0x85
Event	0x06	N/A
(all other values reserved)	0x07..0x7e	0x87..0xfe
Invalid	0x7f	0xff

Table 11.60: SDIO Operation Types

Offset	Field	Size	Value	Description
0	caps	4	Bit Mask	*Greybus SDIO Get Capabilities Bit Masks*
4	ocr	4	Bit Mask	*Greybus SDIO Protocol Voltage Range Bit Mask*
8	f_min	4	Number	Minimum frequency supported by the controller
12	f_max	4	Number	Maximum frequency supported by the controller
16	max_blk_count	2	Number	Maximum Number of blocks per data command transfer
18	max_blk_size	2	Number	Maximum size of each block to transfer

Table 11.61: SDIO Protocol Get Capabilities Response

controller message. The request and response messages for each SDIO operation are defined below.

Table 11.60 defines the Greybus SDIO Protocol operation types and their values. Both the request type and response type values are shown.

11.7.2 Greybus SDIO CPort Shutdown Operation

The Greybus SDIO CPort Shutdown Operation is the *Common Greybus Protocol CPort Shutdown Operation* for the SDIO Protocol.

11.7.3 Greybus SDIO Get Capabilities Operation

The Greybus SDIO Get Capabilities operation allows the requester to fetch capabilities that are supported by the Controller.

Greybus SDIO Get Capabilities Request

The Greybus SDIO Get Capabilities request message has no payload.

Greybus SDIO Get Capabilities Response

The Greybus SDIO Get Capabilities response message returns value whose bits represent the support of certain capability from the SDIO controller, as defined in table 11.61.

Symbol	Brief Description	Mask Value
GB_SDIO_CAP_NONREMOVABLE	Device is unremovable from the slot	0x00000001
GB_SDIO_CAP_4_BIT_DATA	Host support 4 bit transfers	0x00000002
GB_SDIO_CAP_8_BIT_DATA	Host support 8 bit transfers	0x00000004
GB_SDIO_CAP_MMC_HS	Host support mmc high-speed timings	0x00000008
GB_SDIO_CAP_SD_HS	Host support SD high-speed timings	0x00000010
GB_SDIO_CAP_ERASE	Host allow erase and trim commands	0x00000020
GB_SDIO_CAP_1_2V_DDR	Host support DDR mode at 1.2V	0x00000040
GB_SDIO_CAP_1_8V_DDR	Host support DDR mode at 1.8V	0x00000080
GB_SDIO_CAP_POWER_OFF_CARD	Host can power off card	0x00000100
GB_SDIO_CAP_UHS_SDR12	Host support UHS SDR12 mode	0x00000200
GB_SDIO_CAP_UHS_SDR25	Host support UHS SDR25 mode	0x00000400
GB_SDIO_CAP_UHS_SDR50	Host support UHS SDR50 mode	0x00000800
GB_SDIO_CAP_UHS_SDR104	Host support UHS SDR104 mode	0x00001000
GB_SDIO_CAP_UHS_DDR50	Host support UHS DDR50 mode	0x00002000
GB_SDIO_CAP_DRIVER_TYPE_A	Host support Driver Type A	0x00004000
GB_SDIO_CAP_DRIVER_TYPE_C	Host support Driver Type C	0x00008000
GB_SDIO_CAP_DRIVER_TYPE_D	Host support Driver Type D	0x00010000
GB_SDIO_CAP_HS200_1_2V	Host support HS200 mode at 1.2V	0x00020000
GB_SDIO_CAP_HS200_1_8V	Host support HS200 mode at 1.8V	0x00040000
GB_SDIO_CAP_HS400_1_2V	Host support HS400 mode at 1.2V	0x00080000
GB_SDIO_CAP_HS400_1_8V	Host support HS400 mode at 1.8V	0x00100000
	(All other mask values reserved)	0x00200000..0x80000000

Table 11.62: SDIO Protocol Get Capabilities Bit Masks

Offset	Field	Size	Value	Description
0	op	14	Structure	SDIO gb_sdio_ios descriptor

Table 11.63: SDIO Protocol Set Ios Request

Greybus SDIO Get Capabilities Bit Masks

Table 11.62 define the Capabilities bit masks for Greybus SDIO.

11.7.4 Greybus SDIO Set Ios Operation

The Greybus SDIO Set Ios operation allows the requester to setup parameters listed in to SDIO controller

Greybus SDIO Set Ios Request

Table 11.63 defines the Greybus SDIO Set Ios request. The request shall pass a descriptor which contains a set of parameters for configuring the SDIO controller.

Table 11.64 defines the Greybus SDIO gb_sdio_ios. This describes the parameters to configure the SDIO controller.

Greybus SDIO Protocol Voltage Range Bit Mask

Table 11.65 defines the voltage ranges bit masks for the Greybus SDIO controllers.

Offset	Field	Size	Value	Description
0	clock	4	Number	clockrateinHz
4	vdd	4	Number	*Greybus SDIO Protocol Voltage Range Bit Mask*
8	bus_mode	1	Number	*Greybus SDIO Protocol Bus Mode*
9	power_mode	1	Number	*Greybus SDIO Protocol Power Mode*
10	bus_width	1	Number	*Greybus SDIO Protocol Bus Width*
11	timing	1	Number	*Greybus SDIO Protocol Timing*
12	signal_voltage	1	Number	*Greybus SDIO Protocol Signal Voltage*
13	drv_type	1	Number	*Greybus SDIO Protocol Driver Type*

Table 11.64: SDIO Protocol Set Ios Descriptor

Symbol	Brief Description	Mask Value
GB_SDIO_VDD_165_195	VDD voltage 1.65 - 1.95	0x00000001
GB_SDIO_VDD_20_21	VDD voltage 2.0 2.1	0x00000002
GB_SDIO_VDD_21_22	VDD voltage 2.1 2.2	0x00000004
GB_SDIO_VDD_22_23	VDD voltage 2.2 2.3	0x00000008
GB_SDIO_VDD_23_24	VDD voltage 2.3 2.4	0x00000010
GB_SDIO_VDD_24_25	VDD voltage 2.4 2.5	0x00000020
GB_SDIO_VDD_25_26	VDD voltage 2.5 2.6	0x00000040
GB_SDIO_VDD_26_27	VDD voltage 2.6 2.7	0x00000080
GB_SDIO_VDD_27_28	VDD voltage 2.7 2.8	0x00000100
GB_SDIO_VDD_28_29	VDD voltage 2.8 2.9	0x00000200
GB_SDIO_VDD_29_30	VDD voltage 2.9 3.0	0x00000400
GB_SDIO_VDD_30_31	VDD voltage 3.0 3.1	0x00000800
GB_SDIO_VDD_31_32	VDD voltage 3.1 3.2	0x00001000
GB_SDIO_VDD_32_33	VDD voltage 3.2 3.3	0x00002000
GB_SDIO_VDD_33_34	VDD voltage 3.3 3.4	0x00004000
GB_SDIO_VDD_34_35	VDD voltage 3.4 3.5	0x00008000
GB_SDIO_VDD_35_36	VDD voltage 3.5 3.6	0x00010000
	(All other mask values reserved)	0x00020000..0x80000000

Table 11.65: SDIO Protocol Voltage Range Bit Masks

Symbol	Brief Description	Value
GB_SDIO_BUSMODE_OPENDRAIN	SDIO open drain bus mode	0x00
GB_SDIO_BUSMODE_PUSHPULL	SDIO push-pull bus mode	0x01
	(All other values reserved)	0x02..0xff

Table 11.66: SDIO Protocol Bus Mode

Symbol	Brief Description	Value
GB_SDIO_POWER_OFF	SDIO power off	0x00
GB_SDIO_POWER_UP	SDIO power up	0x01
GB_SDIO_POWER_ON	SDIO power on	0x02
GB_SDIO_POWER_UNDEFINED	SDIO power undefined	0x03
	(All other values reserved)	0x04..0xff

Table 11.67: SDIO Protocol Power Mode

Greybus SDIO Protocol Bus Mode

Table 11.66 defines the Mode in which the Bus should be set for operation.

Greybus SDIO Protocol Power Mode

Table 11.67 defines the power supply mode in which the slot should be set.

Greybus SDIO Protocol Bus Width

Table 11.68 defines the values in which the data bus width can be set.

Greybus SDIO Protocol Timing

Table 11.69 defines the timing specification values for the bus.

Greybus SDIO Protocol Signal Voltage

Table 11.70 defines the signal voltage values allowed to be set for the bus.

Greybus SDIO Protocol Driver Type

Table 11.71 defines the driver strength types in which the Controller shall be configured.

Symbol	Brief Description	Value
GB_SDIO_BUS_WIDTH_1	SDIO data bus width 1 bit mode	0x00
GB_SDIO_BUS_WIDTH_4	SDIO data bus width 4 bit mode	0x02
GB_SDIO_BUS_WIDTH_8	SDIO data bus width 8 bit mode	0x03
	(All other values reserved)	0x04..0xff

Table 11.68: SDIO Protocol Bus Width

Symbol	Brief Description	Value
GB_SDIO_TIMING_LEGACY	Default speed	0x00
GB_SDIO_TIMING_MMC_HS	MMC High speed	0x01
GB_SDIO_TIMING_SD_HS	SD High speed	0x02
GB_SDIO_TIMING_UHS_SDR12	Ultra High Speed SDR12	0x03
GB_SDIO_TIMING_UHS_SDR25	Ultra High Speed SDR25	0x04
GB_SDIO_TIMING_UHS_SDR50	Ultra High Speed SDR50	0x05
GB_SDIO_TIMING_UHS_SDR104	Ultra High Speed SDR104	0x06
GB_SDIO_TIMING_UHS_DDR50	Ultra High Speed DDR50	0x07
GB_SDIO_TIMING_MMC_DDR52	MMC DDR52	0x08
GB_SDIO_TIMING_MMC_HS200	MMC HS200	0x09
GB_SDIO_TIMING_MMC_HS400	MMC HS400	0x0A
	(All other values reserved)	0x0B..0xff

Table 11.69: SDIO Protocol Timing

Symbol	Brief Description	Value
GB_SDIO_SIGNAL_VOLTAGE_330	Signal Voltage = 3.30V	0x00
GB_SDIO_SIGNAL_VOLTAGE_180	Signal Voltage = 1.80V	0x01
GB_SDIO_SIGNAL_VOLTAGE_120	Signal Voltage = 1.20V	0x02
	(All other values reserved)	0x03..0xff

Table 11.70: SDIO Protocol Signal Voltage

Symbol	Brief Description	Value
GB_SDIO_SET_DRIVER_TYPE_B	Driver Type B	0x00
GB_SDIO_SET_DRIVER_TYPE_A	Driver Type A	0x01
GB_SDIO_SET_DRIVER_TYPE_C	Driver Type C	0x02
GB_SDIO_SET_DRIVER_TYPE_D	Driver Type D	0x03
	(All other values reserved)	0x04..0xff

Table 11.71: SDIO Protocol Driver Type

Offset	Field	Size	Value	Description
0	cmd	1	Number	SDIO command operation code, as specified by SD Association
1	cmd_flags	1	Bit Mask	*Greybus SDIO Protocol Command Flags*
2	cmd_type	1	Number	*Greybus SDIO Protocol Command Type*
3	arg	4	Number	SDIO command arguments, as specified by SD Association
7	data_blocks	2	Number	If data is available, represents the number of total blocks to transfer, 0 otherwise
9	data_blksz	2	Number	If data is available, represents the size of the blocks to transfer, 0 otherwise

Table 11.72: SDIO Protocol Command Request

Symbol	Brief Description	Mask Value
GB_SDIO_RSP_NONE	No Response is expected by the command	0x00
GB_SDIO_RSP_PRESENT	Response is expected by the command	0x01
GB_SDIO_RSP_136	Long response is expected by the command	0x02
GB_SDIO_RSP_CRC	A valid CRC is expected by the command	0x04
GB_SDIO_RSP_BUSY	Card may send a busy response	0x08
GB_SDIO_RSP_OPCODE	Response contains opcode	0x10
	(All other values reserved)	0x20..0xff

Table 11.73: SDIO Protocol Command Flags

Greybus SDIO Set Ios Response

The Greybus SDIO Set Ios response message has no payload.

11.7.5 Greybus SDIO Command Operation

The Greybus SDIO Command operation allows the requester to send control commands as specified by the SD Association to the SDIO controller.

Greybus SDIO Command Request

Table 11.72 defines the Greybus SDIO Command request.

Greybus SDIO Protocol Command Flags

Table 11.73 defines the flags that can be passed to a command.

Greybus SDIO Protocol Command Type

Table 11.74 defines the command type passed to the MMC/SD card.

Greybus SDIO Command Response

Table 11.75 defines the Greybus SDIO Command response.

Symbol	Brief Description	Value
GB_SDIO_CMD_AC	Addressed Command	0x00
GB_SDIO_CMD_ADTC	Addressed Data Transfer Command	0x01
GB_SDIO_CMD_BC	Broadcasted Command, no response	0x02
GB_SDIO_CMD_BCR	Broadcasted Command with response	0x03
	(All other values reserved)	0x04..0xff

Table 11.74: SDIO Protocol Command Type

Offset	Field	Size	Value	Description
0	resp	16	Number	SDIO command response, as specified by SD Association

Table 11.75: SDIO Protocol Command Response

11.7.6 Greybus SDIO Transfer Operation

The Greybus SDIO Transfer operation allows the requester to send or receive data blocks and shall be preceded by a Greybus Command Request for data transfer command as specified by SD Association.

Greybus SDIO Transfer Request

Table 11.76 defines the Greybus SDIO Transfer request.

If data_flags field have the GB_SDIO_DATA_WRITE flag set, the size field define the length in bytes of data to be transfer in the data field. If data_flags field have the GB_SDIO_DATA_READ set, the size field define the length of data to be read and for that the data field is empty.

Greybus SDIO Transfer Response

Table 11.78 defines the Greybus SDIO Transfer response.

If Request data_flags field have the GB_SDIO_DATA_WRITE flag set, the size field represent the size of data received in the Request in case of success. If data_flags field have the GB_SDIO_DATA_READ set, the size field defines the length of the data appended in the data field.

11.7.7 Greybus SDIO Event Operation

The Greybus SDIO Event operation signals to the recipient that a change in the device setup have occurred in the SDIO controller.

This operation is unidirectional and does not have a correspondent response.

Offset	Field	Size	Value	Description
0	data_flags	1	Number	SDIO data flags
1	data_blocks	2	Number	SDIO number of blocks of data to transfer
3	data_blksz	2	Number	SDIO size of the blocks of data to transfer
5	data	...	Data	SDIO Data

Table 11.76: SDIO Protocol Transfer Request

Symbol	Brief Description	Value
GB_SDIO_DATA_WRITE	Data present in data_blocks request to be written	0x01
GB_SDIO_DATA_READ	Data present in data_blocks response to be read	0x02
GB_SDIO_DATA_STREAM	Data will be transfer until a cancel command is send	0x04
	(All other values reserved)	0x08..0x80

Table 11.77: SDIO Protocol Transfer Data Flags

Offset	Field	Size	Value	Description
0	data_blocks	2	Number	SDIO number of blocks of data to transfer
2	data_blksz	2	Number	SDIO size of the blocks of data to transfer
4	data	...	Data	SDIO Data

Table 11.78: SDIO Protocol Transfer Request

Greybus SDIO Event Request

Table 11.79 defines the Greybus SDIO Event Request. The Request supplies the one-byte event that has occurred on the sending controller.

Greybus SDIO Event Bit Masks

Table 11.80 defines the bit masks which specify the set of events that a controller can trigger related to SD card. If card have the GB_SDIO_CAP_NONREMOVABLE capability, the card detection events shall be ignored.

Offset	Field	Size	Value	Description
0	event	1	Bit Mask	*Greybus SDIO Event Bit Masks*

Table 11.79: SDIO Protocol Detect Event Request

Symbol	Brief Description	Mask Value
GB_SDIO_CARD_INSERTED	Card insertion detect	0x01
GB_SDIO_CARD_REMOVED	Card removed detect	0x02
GB_SDIO_WP	Card Write Protect Switch	0x04
	(All other values reserved)	0x08..0x80

Table 11.80: SDIO Protocol Event Bit Mask

Chapter 12

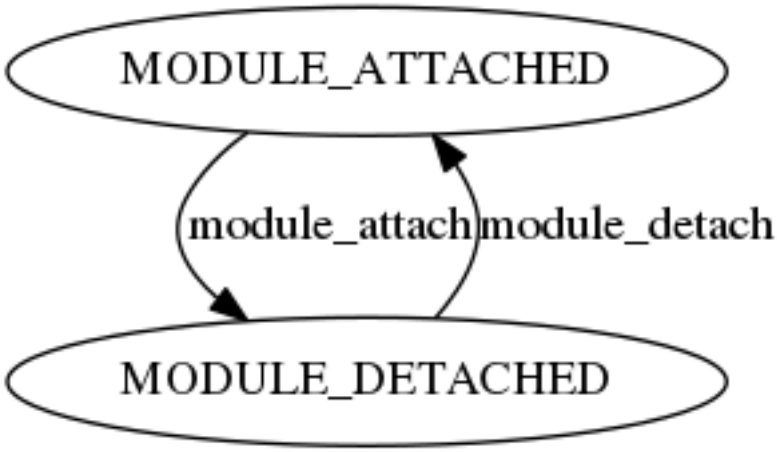
Module and Interface Lifecycles

Chapters *Greybus Hardware Model* and *Special Protocols* have respectively defined the *Interface Lifecycle* and various *Operation*s which affect the related *Interfaces* within Modules and *Interface States* within the Frame in a *Greybus System*.

Using these definitions, this chapter describes an additional state machine, the *Module Lifecycle*, as well as the transitions between nodes in the Interface Lifecycle state machine in more detail.

12.1 The Module Lifecycle

The Module Lifecycle state machine diagram is as follows.



A *Module*'s relationship with the *Greybus System* is simple: the module is either attached to the *Frame* via one or more *Interface Block*s in exactly one *Slot*, in which case the entire Module is in the MODULE_ATTACHED state, or it has been detached entirely, in which case it is not considered a part of the Greybus System.

The following sections describe the relationship between these states, the transitions between them, and certain Greybus *Operation*s.

12.1.1 Module Attach

TODO

12.1.2 Module Detach

TODO

12.2 The Interface Lifecycle

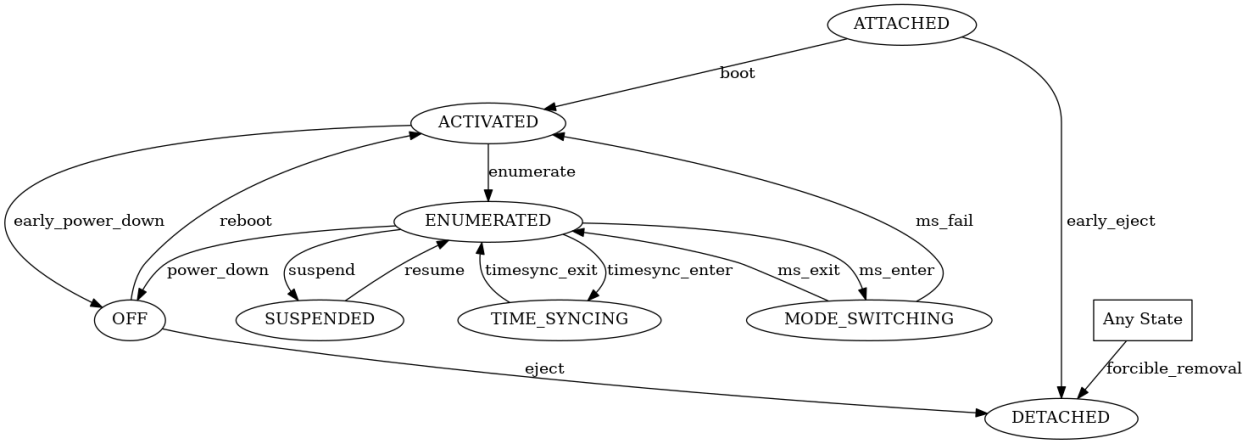
The *Greybus Hardware Model* defined the concept of an *Interface*, and *Interface Lifecycle States* introduced a related set of *Interface Lifecycle States*, along with a state machine which operates on Lifecycle States, the Interface Lifecycle.

A subsequent chapter defined the *Special Protocols*, which include Operation definitions that affect Interfaces' Lifecycle States.

This section describes the relationships between these Protocols and the Interface Lifecycle in more detail, and specifies Operation sequences which may be successfully exchanged to cause Interfaces to change Lifecycle States.

The following sections describe the relationship between these states, as well as how transitions between them may occur in a Greybus System.

For convenience, the Interface Lifecycle state machine diagram and the Interface States associated with each Interface Lifecycle State are reproduced here:



The possible Interface States for when an Interface is ATTACHED are shown in Table 5.13.

The possible Interface States for when an Interface is ACTIVATED are shown in Table 5.15.

The possible Interface States for when an Interface is ENUMERATED are shown in Table 5.16.

The possible Interface States for when an Interface is MODE_SWITCHING are shown in Table 5.17.

The possible Interface States for when an Interface is TIME_SYNCING are shown in Table 5.18.

The possible Interface States for when an Interface is SUSPENDED are shown in Table 5.19.

The possible Interface States for when an Interface is OFF are shown in Table 5.20.

A Module is not attached to the Interface Block in the DETACHED Interface Lifecycle State, which has the unique Interface State shown in Table 5.21.

12.2.1 Connection Management

This section describes the sequences required to manage Greybus Connections during the Interface Lifecycle. Since all Greybus Operations are exchanged via UniPro Messages, these requirements are a superset of those required by UniPro for establishing communication via CPorts.

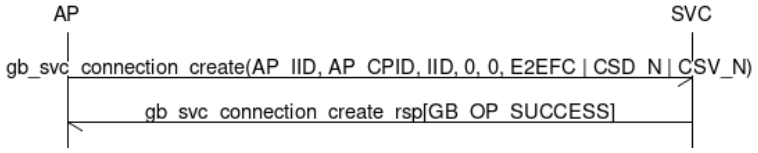
Control Connection Establishment

Note: The content in this section is defined under the assumption that there is exactly one *AP Interface* in the Greybus System.

The results if there are multiple AP Interfaces are undefined.

During SVC Protocol Operation processing defined in *Greybus SVC Interface Activate Operation* and *Greybus SVC Interface Resume Operation*, an Interface may signal to the Frame that it is capable of Greybus Communication, and that its Control CPort user is ready to respond to *Control Protocol* Operations. This can also occur during the processing of a *Greybus Control Mode Switch Operation*.

The following sequence may be used to establish a Control Connection to an Interface for subsequent use.



Though the AP may follow this sequence at any time, the AP should only do so during one of the following transitions in the Interface Lifecycle state machine:

- "enumerate", as described in *Enumerate (ACTIVATED → ENUMERATED)*,
- "resume", as described in *Resume (SUSPENDED → ENUMERATED)*, or
- "ms_exit", as described in *Mode Switch Exit (MODE_SWITCHING → ENUMERATED)*.

If the AP follows this sequence at other times, the results are undefined.

To perform this sequence, the following conditions shall hold.

- The AP Interface and SVC shall have established a Connection implementing the *SVC Protocol*. This is the SVC Connection in this sequence. This implies the AP Interface has a Device ID set.
- Another Interface shall be provided, which has a Control CPort.

If these conditions do not all hold, the sub-sequence shall not be followed. The results of following this sub-sequence in this case are undefined.

The following values are used in this sub-sequence:

- The AP Interface ID is ap_interface_id.
- The CPort ID of a CPort on the AP Interface which is used to establish the Control Connection is ap_cport_id.
- The Interface ID of the other Interface is interface_id.

1. The AP shall initiate a *Greybus SVC Connection Create Operation* to establish the Control Connection.

   The intf1_id and cport1_id fields in the request payload shall respectively equal ap_interface_id and ap_cport_id. The intf2_id and cport2_id fields in the request payload shall respectively equal interface_id and zero.

The tc field in the request payload shall equal zero. The *flags field* in the request payload should equal 0x7 (E2EFC | CSD_N | CSV_N).

The sequence is complete. If this Operation fails, the sequence has failed. If it succeeds, the sequence has succeeded.

2. The sequence is now complete and has succeeded or failed.

If the sequence succeeds, the AP Interface may inititate *Control Protocol* Operations with the Interface by sending requests using CPort ap_cport_id.

If the sequence fails, the AP should not attempt to initiate Control Protocol Operations with the Interface. If the AP does so under this condition, the results are undefined.

Non-Control Connection Establishment

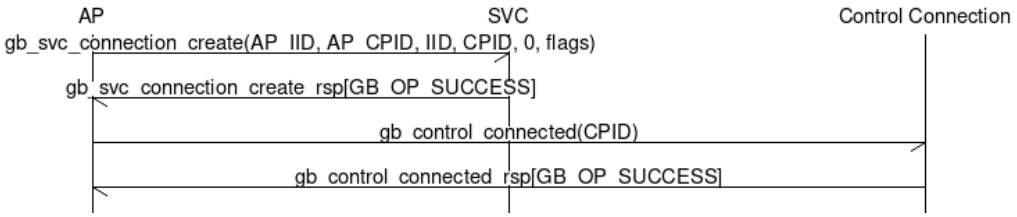
Note: The content in this section is defined under the following assumptions:

- there is exactly one *AP Interface* in the Greybus System.
- The Connection being established is between that AP Interface and another Interface in the System.

The results if there are multiple AP Interfaces, or in the case of non-AP to non-AP Interfaces, are undefined.

If an Interface is *ENUMERATED*, the AP can establish additional Connections to the Interface in addition to the existing Control Connection.

The following sequence may be used to establish such a Connection to an Interface for subsequent use.



Though the AP may follow this sequence at any time, the AP should only do so if the Interface is ENUMERATED. If the AP follows this sequence at other times, the results are undefined.

A CPort ID value interface_cport_id shall be provided for a CPort on the Interface, and is used in this sequence. The value shall have been given in the "id" field of a *CPort Descriptor* in the Interface *Manifest* in the response payload of the *Greybus Control Get Manifest Operation* Operation which was exchanged during the most recent enumeration of the Interface. The AP should additionally ensure that the CPort on the Interface with CPort ID interface_cport_id is not already at one end of an established Greybus Connection.

Another value, ap_cport_id, shall also be provided. The AP Interface shall contain a CPort with CPort ID ap_cport_id. The AP should ensure that this CPort is not part of an established UniPro connection.

The following values are used in this sub-sequence:

- The AP's Interface ID is ap_interface_id.
- The Interface ID of the ENUMERATED Interface is interface_id.

1. The AP shall initiate a *Greybus SVC Connection Create Operation* to establish the Connection.

   The intf1_id and cport1_id fields in the request payload shall respectively equal ap_interface_id and ap_cport_id. The intf2_id and cport2_id fields in the request payload shall respectively equal interface_id and interface_cport_id. The tc field in the request payload shall equal zero.

The flags field in the request payload is *Protocol* -specific.

If this Operation fails, the sequence is complete and has failed. Go directly to step 4.

2. The AP shall initiate a *Greybus Control Connected Operation* request on the Interface's Control Connection. The cport_id field in the request payload shall equal interface_cport_id.

   If this Operation fails, the sequence has failed.

   If it succeeds, the sequence has succeeded. Go directly to step 4.

3. Since the sequence has failed, the AP initiates a *Greybus SVC Connection Destroy Operation* Operation to disconnect the CPort which was connected in step 1.

   The intf1_id, cport1_id, intf2_id, and cport2_id fields in the request payload shall respectively equal ap_interface_id, ap_cport_id, interface_id, and interface_cport_id.

4. The sequence is now complete and has succeeded or failed.

If the sequence succeeds, the AP, and on a protocol-specific basis, the Interface, may initiate Greybus Operations on the newly established Connection. In this case, the Greybus Protocol used shall correspond to the "protocol" field for the CPort descriptor referenced in step 1, as defined by Table 6.10.

If the sequence fails, the AP should not, and the Interface shall not, initiate Greybus communication on any of the CPorts referenced in step 1. If this occurs, the results are undefined.

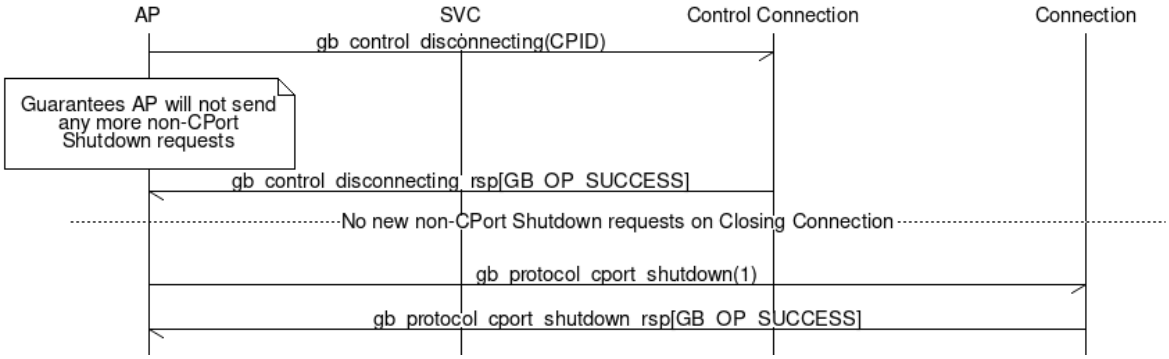
Connection Closure Prologue

Note: The content in this section is defined under the following assumptions:

- there is exactly one *AP Interface* in the Greybus System.
- The Connection being closed is between that AP Interface and another Interface in the System.

The results if there are multiple AP Interfaces, or in the case of non-AP to non-AP Interfaces, are undefined.

This section defines a common sub-sequence, the connection closure prologue sub-sequence, which is used by following sections in order to close a Greybus Connection.



To perform this sub-sequence, the following conditions shall hold.

- The AP Interface and SVC shall have established a Connection implementing the *SVC Protocol*. This is the SVC Connection in this sub-sequence.
- A Connection between the AP Interface and another Interface shall be defined, which is now being closed.

This is the Closing Connection here. The Closing Connection may be the Control Connection, or some other Greybus Connection between the AP Interface and the other Interface.

- The AP Interface and the other Interface shall have established a Control Connection. This is the Control Connection in this sub-sequence.

If these conditions do not all hold, the sub-sequence shall not be followed. The results of following this sub-sequence in this case are undefined.

The following values are used in this sub-sequence:

- The AP Interface ID is ap_interface_id.
- The CPort ID of the CPort on the AP Interface which is at one end of the Closing Connection is ap_cport_id.
- The Interface ID of the other Interface is interface_id.
- The CPort ID on the other Interface which is at the other end of the Closing Connection is interface_cport_id. If the Closing Connection is the Control Connection, interface_cport_id is zero.

1. The AP Interface shall exchange a *Greybus Control Disconnecting Operation* with the Interface on the Control Connection. The cport_id field in the request payload shall equal interface_cport_id.
2. The AP Interface may now issue responses to requests it has already received on the Closing Connection. It shall not issue any such responses after this step.
3. The AP shall exchange a *Common Greybus Protocol CPort Shutdown Operation* with the Interface on the Closing Connection.

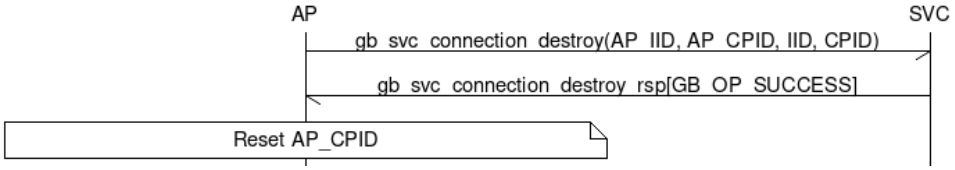
   The connection closure prologue sub-sequence has succeeded.

Connection Closure Epilogue

Note: The content in this section is defined under the assumption that there is exactly one *AP Interface* in the Greybus System.

The results if there are multiple AP Interfaces are undefined.

This section defines a common sub-sequence, the connection closure epilogue sub-sequence, which is used by following sections in order to close a Greybus Connection.



To perform this sub-sequence, the following conditions shall hold.

- The AP Interface and SVC shall have established a Connection implementing the *SVC Protocol*. This is the SVC Connection in this sub-sequence.
- A Connection between the AP Interface and another Interface shall be provided. This is the Closing Connection in this sub-sequence.

If these conditions do not all hold, the sub-sequence shall not be followed. The results of following this sub-sequence in this case are undefined.

The following values are used in this sub-sequence:

- The AP Interface ID is ap_interface_id.
- The CPort ID of the CPort on the AP Interface which is at one end of the Closing Connection is ap_cport_id.
- The Interface ID of the other Interface is interface_id.
- The CPort ID on the other Interface which is at the other end of the Closing Connection is interface_cport_id.

1. The AP Interface shall initiate a *Greybus SVC Connection Destroy Operation* on the SVC Connection.

   The intf1_id and cport1_id fields in the request payload shall respectively equal ap_interface_id and ap_cport_id. The intf2_id and cport2_id fields in the request payload shall respectively equal interface_id and interface_cport_id.

   If this Operation fails, the connection closure epilogue sub-sequence has failed. Go to the next step.

2. The AP Interface shall perform any implementation-defined procedures required to make the CPort with ID ap_cport_id usable if a Greybus Connection is later reestablished on that CPort.

   The AP Interface may set local UniPro attributes related to that CPort to implementation-defined values as part of this process. If such procedures are required by the AP Interface, it shall complete them before going to the next step.

   If the connection closure epilogue sub-sequence did not fail in step 1, it has now succeeded.

3. The connection closure epilogue sub-sequence is now complete, and has succeeded or failed.

Non-Control Connection Closure

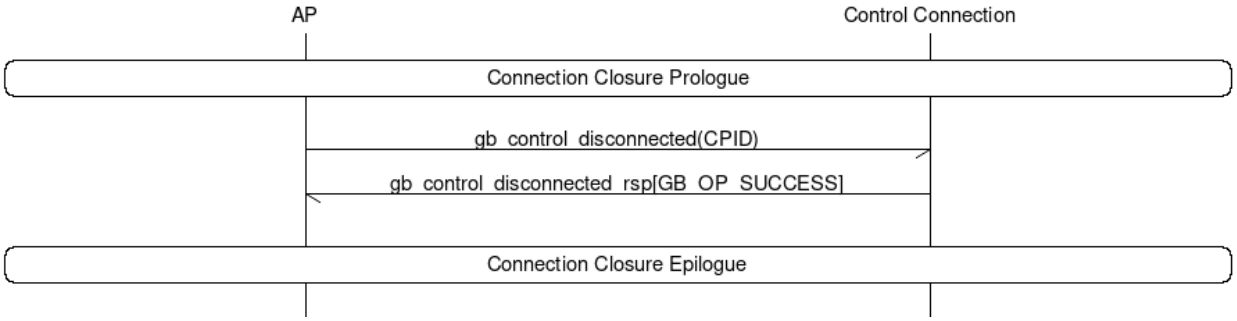
Note: The content in this section is defined under the following assumptions:

- there is exactly one *AP Interface* in the Greybus System.
- The Connection being closed is between that AP Interface and another Interface in the System.

The results if there are multiple AP Interfaces, or in the case of non-AP to non-AP Interfaces, are undefined.

If an Interface is *ENUMERATED* and a Non-Control Connection has been established between the AP and the Interface as described in *Non-Control Connection Establishment*, the AP can subsequently close the Connection to the Interface.

The following sequence may be used to close such a Connection to an Interface.



Though the AP may follow this sequence at any time, the AP should only do so if the Interface whose Connection is being closed is ENUMERATED, or during one of the following Interface Lifecycle state machine transitions which cause the Interface to exit the ENUMERATED Lifecycle State:

- "power_down", as described in *Power Down (ENUMERATED → OFF)*
- "suspend", as described in *Suspend (ENUMERATED → SUSPENDED)*
- "ms_enter", as described in *Mode Switch Enter (ENUMERATED → MODE_SWITCHING)*

If the AP follows this sequence at other times, the results are undefined.

The following values are used in this sub-sequence:

- The AP Interface ID is ap_interface_id.
- The CPort ID of the CPort on the AP Interface which is at one end of the Closing Connection is ap_cport_id.
- The Interface ID of the other Interface is interface_id.
- The CPort ID on the other Interface which is at the other end of the Closing Connection is interface_cport_id.

1. The *Connection Closure Prologue* sub-sequence is followed. The Closing Connection for that sub-sequence is the one being closed in this sequence. If the sub-sequence fails, this sequence has failed. Go directly to step 4.
2. The AP exchanges a *Greybus Control Disconnected Operation* on the Interface's Control Connection. The cport_id field in the request payload shall equal interface_cport_id.
3. The *Connection Closure Epilogue* sub-sequence is followed. The Closing Connection for that sub-sequence is the one being closed in this sequence. If the sub-sequence fails, this sequence has failed. Otherwise, it has succeeded.
4. The sequence is now complete, and has succeeded or failed.

If the sequence succeeds, the AP Interface and the other Interface shall respectively not transmit on CPorts ap_cport_id and interface_cport_id unless a Greybus Connection is subsequently established using either of the two CPorts. Any UniPro Messages received by those Interfaces shall be discarded.

Regardless of success or failure, the AP Interface shall not initiate any communication on the CPort unless it is at one end of a Connection which is successfully established subsequently.

If the sequence fails, the results are undefined.

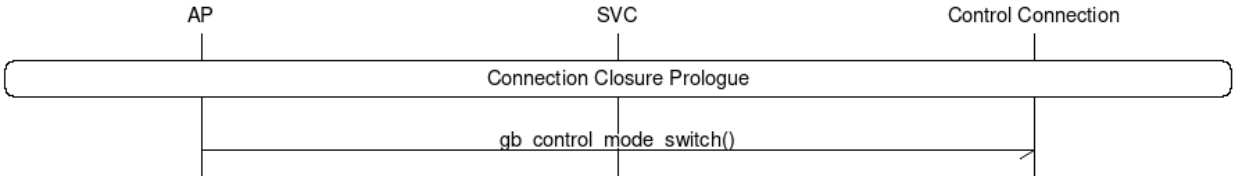
Control Connection Closure for ms_enter

Note: The content in this section is defined under the assumption that there is exactly one *AP Interface* in the Greybus System.

The results if there are multiple AP Interfaces are undefined.

If an Interface is *ENUMERATED*, its Control Connection is established. The AP can subsequently close the Control Connection to the Interface.

The following sequence may be used to close the Control Connection to an Interface while the Interface is entering the MODE_SWITCHING state, and also to signal to the Interface that its Control Connection is closing and it has entered MODE_SWITCHING.



Though the AP may follow this sequence at any time, the AP should only do so if the Interface is ENUMERATED, during the "ms_enter" Interface Lifecycle state machine transition, which causes the Interface to exit the ENUMERATED Lifecycle State as described in *Mode Switch Enter (ENUMERATED → MODE_SWITCHING)*.

If the AP follows this sequence at other times, the results are undefined.

1. The *Connection Closure Prologue* sub-sequence is followed. The Closing Connection for that sub-sequence is the Control Connection for the Interface. If the sub-sequence fails, this sequence has failed. Go directly to step 3.
2. The AP shall send a *Greybus Control Mode Switch Operation* to the Interface. The Operation is unidirectional; this step succeeds. This sequence has succeeded.
3. The sequence is now complete and has succeeded or failed.

If the sequence fails, the results are undefined.

Control Connection Closure for Power Management

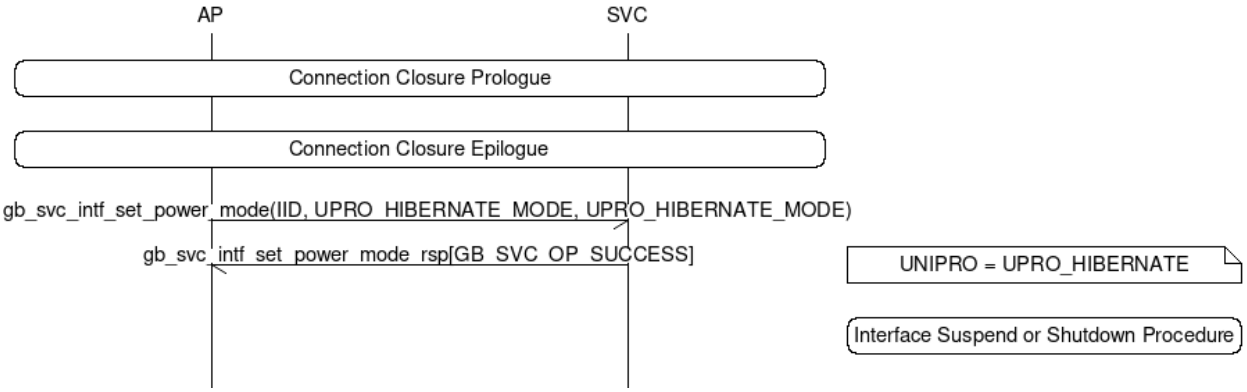
Note: The content in this section is defined under the following assumptions:

- there is exactly one *AP Interface* in the Greybus System.
- The Connection being closed is between that AP Interface and another Interface in the System.

The results if there are multiple AP Interfaces, or in the case of non-AP to non-AP Interfaces, are undefined.

If an Interface is *ENUMERATED*, its Control Connection is established. The AP can subsequently close the Control Connection to the Interface.

The following sequence may be used to close the control Connection to an Interface while the Interface is entering the either the SUSPENDED state or the OFF state.



Though the AP may follow this sequence at any time, the AP should only do so if the Interface is ENUMERATED, during either the "suspend" or "power_down" Interface Lifecycle state machine transitions, which cause the Interface to exit the ENUMERATED Lifecycle State as described in *Suspend (ENUMERATED → SUSPENDED)* and *Power Down (ENUMERATED → OFF)*, respectively.

If the AP follows this sequence at other times, the results are undefined.

The following value is used in this sub-sequence:

- The Interface ID of the other Interface is interface_id.

1. The *Connection Closure Prologue* sub-sequence is followed. The Closing Connection for that sub-sequence is the Control Connection for the other Interface. If the sub-sequence fails, this sequence has failed. If it has failed, go directly to step 5.
2. The *Connection Closure Epilogue* sub-sequence is followed. The Closing Connection for that sub-sequence is the Control Connection for the other Interface. If the sub-sequence fails, this sequence has failed. If it has failed, go directly to step 5.
3. The AP shall exchange a *Greybus SVC Interface Set Power Mode Operation* with the SVC.

   The intf_id field in the request payload shall equal interface_id. The tx_mode and rx_mode fields shall both equal UNIPRO_HIBERNATE_MODE.

   If the Operation fails, this procedure has failed. Go directly to step 5.

   If it succeeds, the SVC shall set the UNIPRO Interface State to UPRO_HIBERNATE. The SVC shall wait an implementation-defined duration in this step to allow the Interface to power down or suspend internally in the next step.

   If the Operation succeeds, this procedure has succeeded.
4. The Interface shall be capable of receiving notification that UNIPRO became UPRO_HIBERNATE.

   The Interface shall have previously been notified whether the change to UPRO_HIBERNATE denotes suspend or power down as described below in *Suspend (ENUMERATED → SUSPENDED)*.

   If the Interface is suspending, it shall perform implementation-specific procedures to ensure it can be resumed successfully if it remains SUSPENDED, then the procedure defined in *Resume (SUSPENDED → ENUMERATED)* is subsequently followed.

   Otherwise, the Interface may now perform implementation-defined procedures used during shutdown.
5. The sequence is now complete, and has succeeded or failed.

12.2.2 Boot and Enumeration

This section describes the procedures required to initialize an *ATTACHED* Interface, putting it in the *ACTIVATED* Lifecycle State.

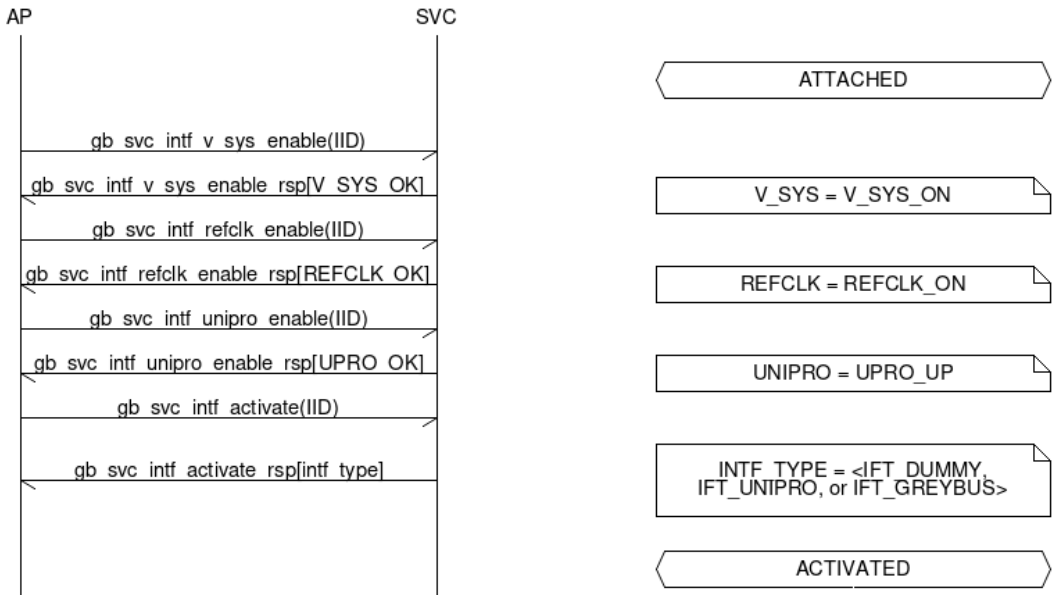
If an ACTIVATED *Interface State's INTF_TYPE* is IFT_GREYBUS, the Interface can be enumerated, as outlined in *ENUMERATED*. The enumeration procedure under these conditions is also defined in this section.

Boot (ATTACHED → ACTIVATED)

Note: The content in this section is defined under the assumption that there is exactly one *AP Interface* in the Greybus System.

The results if there are multiple AP Interfaces are undefined.

The following procedure can be initiated by the AP when an Interface is ATTACHED, in order to attempt to follow the "boot" transition from ATTACHED to ACTIVATED.



To perform this procedure, the following conditions shall hold.

- The AP Interface and SVC shall have established a Connection implementing the *SVC Protocol*. This is the SVC Connection in this procedure.
- An Interface shall be provided, whose Interface Lifecycle State is ATTACHED. No other actions shall have been taken to affect the Interface's Lifecycle State or its corresponding Interface State since the Interface became ATTACHED, except as defined in this procedure.

If these conditions do not all hold, the procedure shall not be followed. The results of following this procedure in this case are undefined.

The following value is used in this procedure:

- The Interface ID of the Interface being activated is interface_id.

1. The AP shall exchange an *Greybus SVC Interface V_SYS Enable Operation* with the SVC. The intf_id field in the request payload shall equal interface_id.

   If the Operation fails, this procedure has failed. Go to step 8.

2. The AP shall exchange an *Greybus SVC Interface REFCLK Enable Operation* with the SVC. The intf_id field in the request payload shall equal interface_id.

   If the Operation fails, this procedure has failed. Go to step 7.

3. The AP shall exchange an *Greybus SVC Interface UNIPRO Enable Operation* with the SVC. The intf_id field in the request payload shall equal interface_id.

   If the Operation fails, this procedure has failed. Go to step 6.

4. The AP shall exchange an *Greybus SVC Interface Activate Operation* with the SVC. The intf_id field in the request payload shall equal interface_id.

   If the Operation fails, this procedure has failed. Go to step 5.

   If the Operation succeeds, this procedure has succeeded. The Interface is now ACTIVATED. Go to step 8.

5. The AP shall exchange a *Greybus SVC Interface UNIPRO Disable Operation* with the SVC. The intf_id field in the request payload shall equal interface_id.

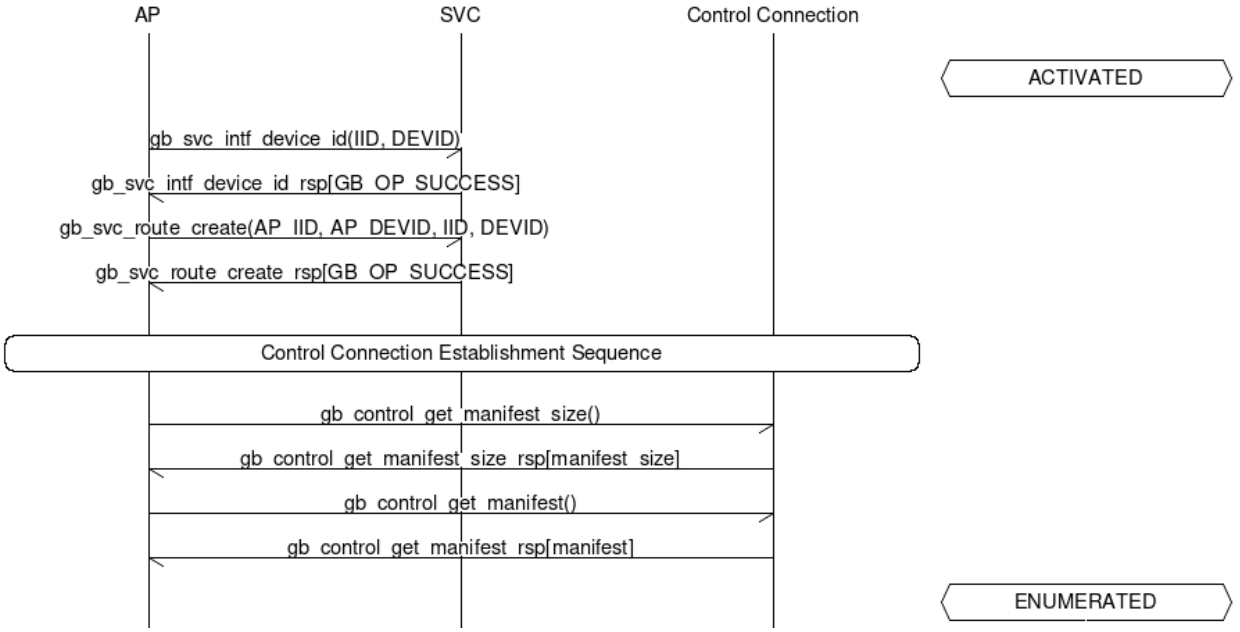
6. The AP shall exchange a *Greybus SVC Interface REFCLK Disable Operation* with the SVC. The intf_id field in the request payload shall equal interface_id.
7. The AP shall exchange a *Greybus SVC Interface V_SYS Disable Operation* with the SVC. The intf_id field in the request payload shall equal interface_id.
8. The procedure is complete and has succeeded or failed. If the procedure failed and all of the steps 5, 6, and 7 which were reached succeeded, the Interface is now ATTACHED.

Enumerate (ACTIVATED → ENUMERATED)

Note: The content in this section is defined under the assumption that there is exactly one *AP Interface* in the Greybus System.

The results if there are multiple AP Interfaces are undefined.

The following procedure can be initiated by the AP when an Interface is ACTIVATED and its *INTF_TYPE* is IFT_GREYBUS, in order to attempt to follow the "enumerate" transition from ACTIVATED to ENUMERATED.



To perform this procedure, the following conditions shall hold.

- The AP Interface and SVC shall have established a Connection implementing the *SVC Protocol*. This is the SVC Connection in this procedure.
- An Interface shall be provided, whose Interface Lifecycle State is ACTIVATED, and whose INTF_TYPE is IFT_GREYBUS. No other actions shall have been taken to affect the Interface's Lifecycle State or its corresponding Interface State since the Interface became ACTIVATED.

If these conditions do not all hold, the procedure shall not be followed. The results of following this procedure in this case are undefined.

The following values are used in this procedure:

- The AP Interface Device ID is ap_device_id.

- The Interface ID of the Interface being enumerated is interface_id.

1. The AP shall initiate a *Greybus SVC Interface Device ID Operation* to assign a Device ID to the Interface.

   The intf_id in the request payload shall equal interface_id.

   The device_id field in the request payload shall be unique among all values assigned to Interfaces in the Greybus System.

   Additionally, the AP shall ensure that no other Interface shall currently have been assigned a Device ID within the following inclusive range:

   ```
   device_id, device_id + 1, ..., device_id + (max_conn / 32)
   ```

   Where max_conn is the maximum value of the Interface's CPort ID for any Connection the AP subsequently intends to establish with the Interface, including the Control Connection, and "/" denotes division with remainder truncated towards zero.

   If this Operation fails, the sequence is complete and has failed. Go directly to step 9.

2. The AP shall initiate a *Greybus SVC Route Create Operation* to establish a route within the *Switch* between an AP Interface and the Interface.

   The intf1_id and dev1_id fields in the request payload shall respectively equal ap_interface_id and ap_device_id. The intf2_id field in the request payload shall equal interface_id. The dev2_id field in the request payload shall have the same value as the device_id field from step 1.

   If this Operation fails, the sequence is complete and has failed. Go directly to step 9.

3. The sequence to establish a Control Connection to the Interface described in *Control Connection Establishment* shall be followed.

   If the sequence fails, this procedure has failed. Go to step 8.

4. The AP shall exchange a *Greybus Control Get Manifest Size Operation* via the Control Connection. If the Operation is successful, the value of the manifest_size field in the response payload is interface_manifest_size.

   If the Operation fails, this procedure has failed. Go to step 7.

5. The AP shall exchange a *Greybus Control Get Manifest Operation* via the Control Connection. If the Operation is successful, the Manifest's value is interface_manifest.

   If the Operation fails, this procedure has failed. Go to step 7.

6. The AP shall perform implementation-defined procedures to parse the *components of the Manifest*.

   The Interface is now ENUMERATED. Go to step 9.

7. The AP shall attempt to close the Control Connection to the Interface as described in *Control Connection Closure for Power Management*. Regardless of the Operation's success or failure, go to the next step.

8. The AP shall perform the procedure described in below in *Early Power Down (ACTIVATED → OFF)*. If the Early Power Down procedure succeeds, and step 7 succeeded if it was reached, the Interface is *OFF*. Its Interface State's INTF_TYPE is still IFT_GREYBUS, and its ORDER has not changed its value since before this Enumerate procedure was followed.

9. The procedure is complete and has succeeded or failed.

If the Interface is now ENUMERATED, additional Connections to the Interface may be established using the sequence defined in *Non-Control Connection Establishment*, and closed using the sequence defined in *Non-Control Connection Closure*; if no errors occur, the Interface remains ENUMERATED.

12.2.3 Power Management

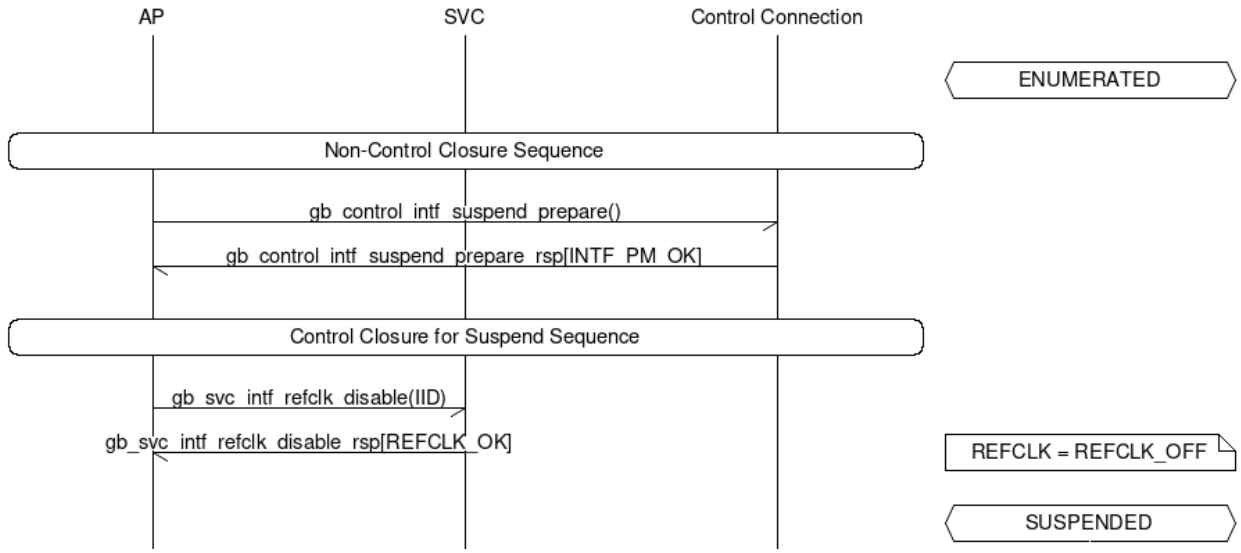
Suspend (ENUMERATED → SUSPENDED)

Note: The content in this section is defined under the following assumptions:

- there is exactly one *AP Interface* in the Greybus System.
- The Non-Control Connections given below are each between that AP Interface and another Interface in the System.

The results if there are multiple AP Interfaces, or in the case of non-AP to non-AP Interfaces, are undefined.

The following procedure can be initiated by the AP when an Interface is ENUMERATED, in order to attempt to follow the "suspend" transition from ENUMERATED to SUSPENDED.



To perform this procedure, the following conditions shall hold.

- The AP Interface and SVC shall have established a Connection implementing the *SVC Protocol*. This is the SVC Connection in this procedure.
- An Interface shall be provided, whose Interface Lifecycle State is ENUMERATED.
- Zero or more additional Non-Control Connections shall be provided, which comprise all such established Connections involving the Interface, and shall each have been established by following the sequence defined in *Non-Control Connection Establishment*.
- For every non-Control Connection: a set of zero or more Bundles shall be provided.

If these conditions do not all hold, the procedure shall not be followed. The results of following this procedure in this case are undefined.

The following values are used in this procedure:

- The AP Interface's ID is ap_interface_id.
- The Interface ID of the Interface being suspended is interface_id.

1. For every Bundle associated with the Interface being suspended which is in the *BUNDLE_ACTIVE* state the AP shall:

(a) follow the *Non-Control Connection Closure* procedure for every CPort for which a Connection is established. If the procedure fails for any CPort, the whole Interface Suspend procedure is failed,

(b) exchange a *Greybus Control Bundle Suspend Operation* with the Interface being suspended. If any Request fails, the whole Interface Suspend is failed.

If any step above fails, the AP Interface should re-establish all Connections previously closed following the *Non-Control Connection Establishment* procedure in order to return to the previous state.

2. The AP Interface shall exchange a *Greybus Control Interface Suspend Prepare Operation* with the Interface being suspended. If the Operation fails (including the cases where the AP may retry sending the Control Suspend Request as defined in the Operation description), this procedure has failed.
3. The sequence defined in *Control Connection Closure for Power Management* shall be followed to close the Control Connection to the Interface.

   If the sequence fails, this procedure has failed. The results are undefined.
4. The Interface shall be capable of receiving notification that UNIPRO became UPRO_HIBERNATE. The Interface shall now enter an implementation-defined suspend state, during which it should attempt to draw minimal power from the Frame.
5. The AP shall exchange an *Greybus SVC Interface REFCLK Disable Operation* with the SVC. The intf_id field in the request payload shall equal interface_id.

   If the Operation succeeds, this procedure has succeeded.

   If the Operation fails, this procedure has failed. The results are undefined.
6. This procedure is now complete, and has either succeeded or failed. If it succeeded, the Interface is now SUSPENDED.

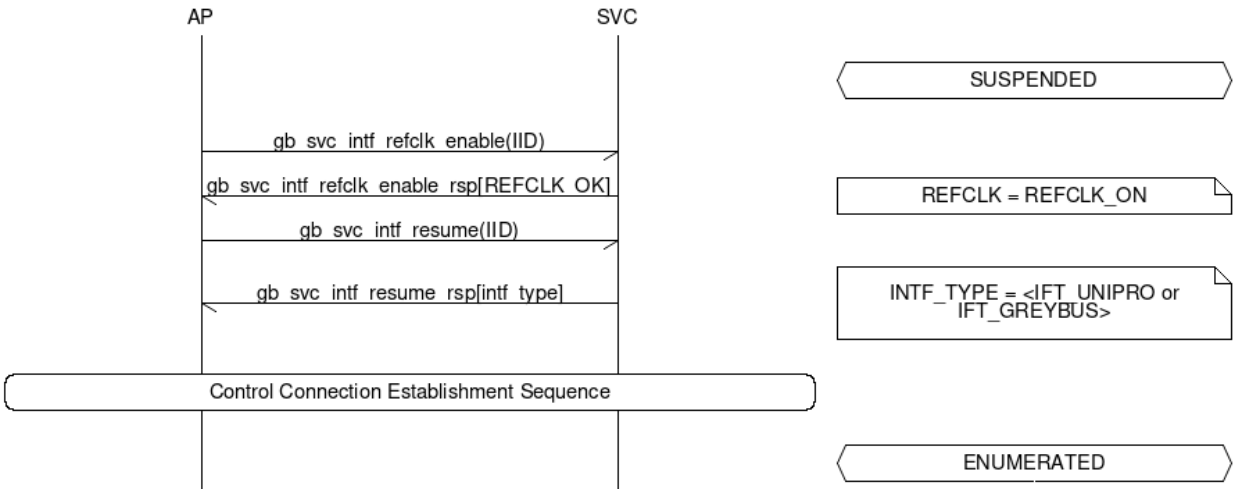
Resume (SUSPENDED → ENUMERATED)

Note: The content in this section is defined under the following assumptions:

- there is exactly one *AP Interface* in the Greybus System.
- The Non-Control Connections given below were each between that AP Interface and another Interface in the System.

The results if there are multiple AP Interfaces, or in the case of non-AP to non-AP Interfaces, are undefined.

The following procedure can be initiated by the AP when an Interface is SUSPENDED, in order to attempt to follow the "resume" transition from SUSPENDED to ENUMERATED.



To perform this procedure, the following conditions shall hold.

- The AP Interface and SVC shall have established a Connection implementing the *SVC Protocol*. This is the SVC Connection in this procedure.
- An Interface shall be provided, whose Interface Lifecycle State is SUSPENDED. The Interface shall have transitioned to the SUSPENDED Lifecycle State by following the suspend procedure defined in *Suspend (ENUMERATED → SUSPENDED)*.
- Zero or more additional Non-Control Connections shall be provided, which comprise all such established Connections involving the Interface when the suspend procedure was followed.
- A Device ID value shall be provided, which is the SUSPENDED Interface's Device ID previously assigned Device ID used to destroy any Routes to the Interface as defined in *Suspend (ENUMERATED → SUSPENDED)*.
- A CPort ID value shall be provided, which was the AP CPort ID which was previously used for the Interface Control Connection before the Interface was suspended.

If these conditions do not all hold, the procedure shall not be followed. The results of following this procedure in this case are undefined.

The following values are used in this procedure:

- The AP Interface's ID is ap_interface_id.
- The AP Interface Device ID is ap_device_id.
- The Provided AP CPort ID used for the Interface Control Connection is ap_cport_id.
- The Interface ID of the Interface being resumed is interface_id.
- The provided Device ID of the Interface being resumed is interface_device_id.

1. The AP shall exchange an *Greybus SVC Interface REFCLK Enable Operation* with the SVC. The intf_id field in the request payload shall equal interface_id.

   If the Operation fails, this procedure has failed. The results are undefined.

2. The AP shall exchange an *Greybus SVC Interface Resume Operation* with the SVC. The intf_id field in the request payload shall equal interface_id.

   If the Operation fails, this procedure has failed. The results are undefined.

3. The sequence to establish a Control Connection to the Interface described in *Control Connection Establishment* shall be followed.

If the sequence fails, this procedure has failed. The results are undefined.

If it succeeds, the procedure has succeeded. The Interface is ENUMERATED. The requirements specified in *Greybus SVC Interface Resume Operation* guarantee that the Interface has the same Manifest defined as that it made available to the AP Interface the most recent time it was ENUMERATED.

4. The procedure is complete and has succeeded or failed.

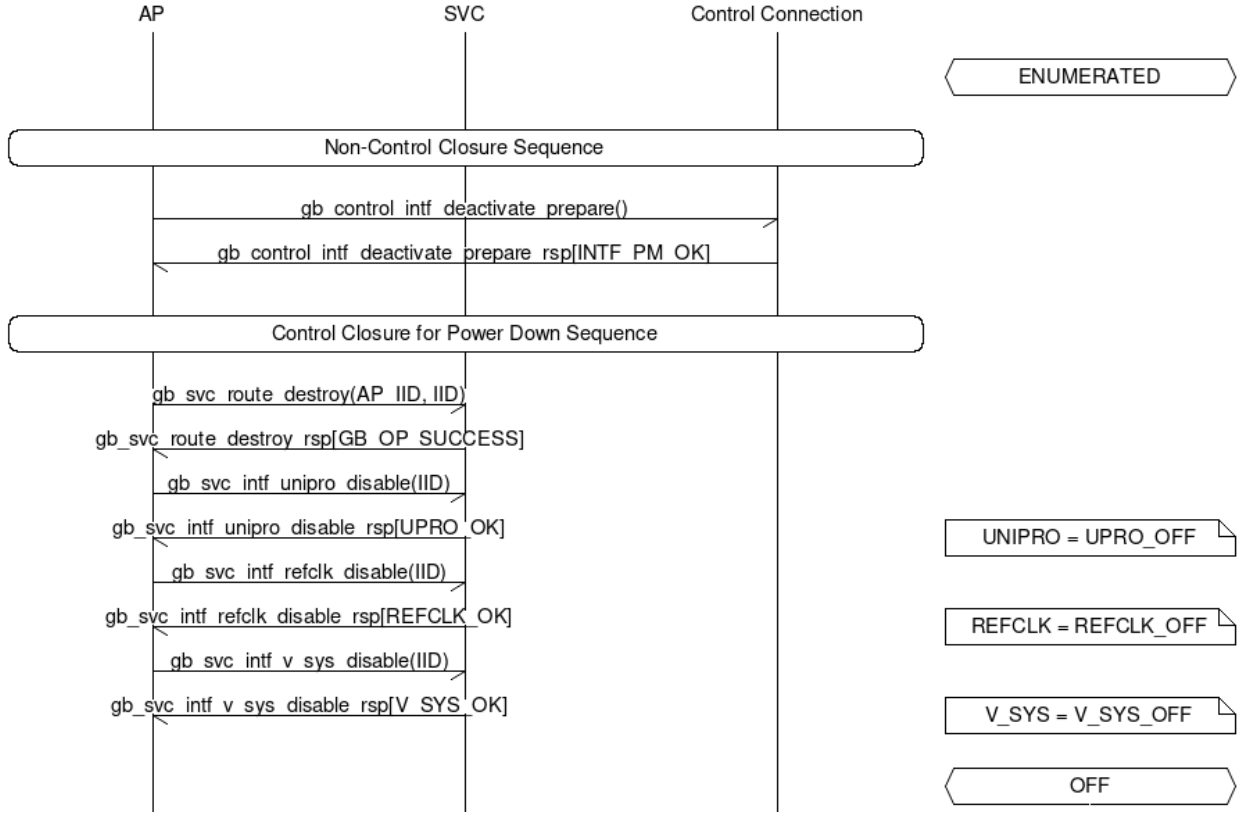
Power Down (ENUMERATED → OFF)

Note: The content in this section is defined under the following assumptions:

- there is exactly one *AP Interface* in the Greybus System.
- The Non-Control Connections given below are each between that AP Interface and another Interface in the System.

The results if there are multiple AP Interfaces, or in the case of non-AP to non-AP Interfaces, are undefined.

The following procedure can be initiated by the AP when an Interface is ENUMERATED, in order to attempt to follow the "power_down" transition from ENUMERATED to OFF.



To perform this procedure, the following conditions shall hold.

- The AP Interface and SVC shall have established a Connection implementing the *SVC Protocol*. This is the SVC Connection in this procedure.
- An Interface shall be provided, whose Interface Lifecycle State is ENUMERATED.

- Zero or more additional Non-Control Connections shall be provided, which comprise all such established Connections involving the Interface, and shall each have been established by following the sequence defined in *Non-Control Connection Establishment*.
- For every non-Control Connection: a set of zero or more Bundles shall be provided.

If these conditions do not all hold, the procedure shall not be followed. The results of following this procedure in this case are undefined.

The following values are used in this procedure:

- The AP Interface's ID is ap_interface_id.
- The Interface ID of the Interface being powered off is interface_id.

1. For every Bundle associated with the Interface being powered off which is not in the *BUNDLE_OFF* state:
   (a) if the Bundle is in the *BUNDLE_ACTIVE* state, the AP shall follow the *Non-Control Connection Closure* procedure for every CPort for which a Connection is established,
   (b) if the Bundle is in the *BUNDLE_SUSPENDED* state, the AP exchange the *Greybus Control Bundle Resume Operation* with the Bundle in order to bring it back to the *BUNDLE_ACTIVE* state and then follow step a),
   (c) exchange a *Greybus Control Bundle Deactivate Operation* with the Interface being powered down,

   If any step above fails, this step shall be considered failed, but the AP should still forcibly power down the Interface by continuing from step 3.
2. The AP Interface shall exchange a *Greybus Control Interface Deactivate Prepare Operation* with the Interface being powered down. If the Operation fails (including the cases where the AP may retry sending the Greybus Control Interface Deactivate Prepare Operation as defined in the Operation description), this procedure has failed.
3. The sequence defined in *Control Connection Closure for Power Management* shall be followed to close the Control Connection to the Interface.

   If the sequence fails, this procedure has failed. The results are undefined.
4. The AP shall exchange a *Greybus SVC Route Destroy Operation* with the SVC. The intf1_id and intf2_id fields in the request payload shall respectively equal ap_interface_id and interface_id.

   If the Operation fails, this procedure has failed. The results are undefined.
5. The AP shall exchange an *Greybus SVC Interface UNIPRO Disable Operation* with the SVC to disable UNIPRO within the Switch.

   If the Operation fails, this procedure has failed. The results are undefined.
6. The AP shall exchange an *Greybus SVC Interface REFCLK Disable Operation* with the SVC. The intf_id field in the request payload shall equal interface_id.

   If the Operation succeeds, this procedure has succeeded.

   If the Operation fails, this procedure has failed. The results are undefined.
7. The AP shall exchange an *Greybus SVC Interface V_SYS Disable Operation* with the SVC. The intf_id field in the request payload shall equal interface_id.

   If the Operation succeeds, this procedure has succeeded.

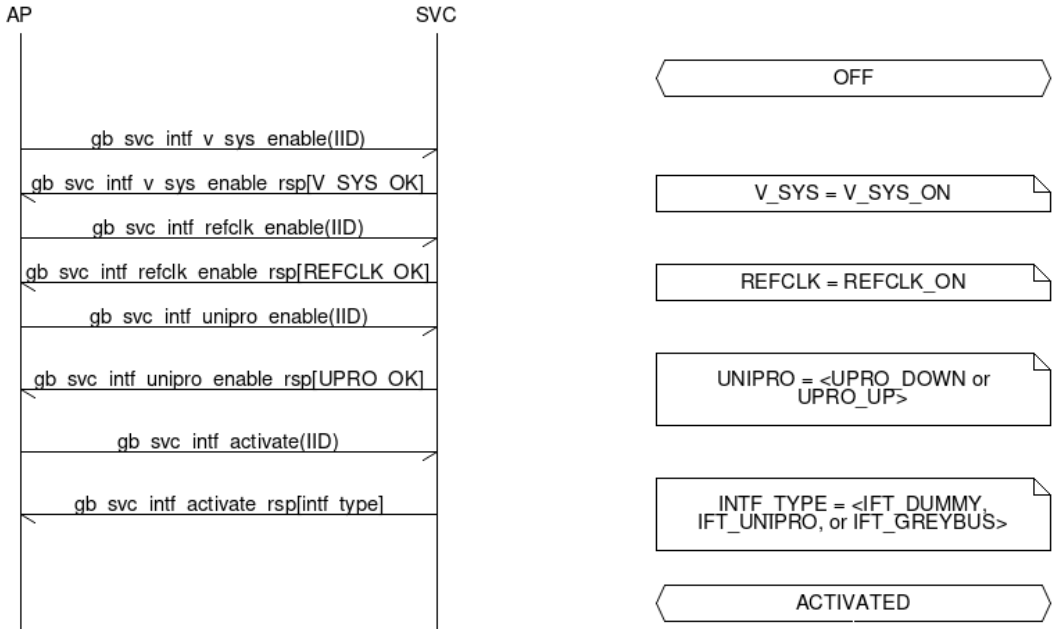
   If the Operation fails, this procedure has failed. The results are undefined.
8. This procedure is now complete, and has either succeeded or failed. If it succeeded, the Interface is now OFF.

Reboot (OFF → ACTIVATED)

Note: The content in this section is defined under the assumption that there is exactly one *AP Interface* in the Greybus System.

The results if there are multiple AP Interfaces are undefined.

The following procedure can be initiated by the AP when an Interface is OFF, in order to attempt to follow the "reboot" transition from OFF to ACTIVATED.



To perform this procedure, the following conditions shall hold.

- The AP Interface and SVC shall have established a Connection implementing the *SVC Protocol*. This is the SVC Connection in this procedure.
- An Interface shall be provided, whose Interface Lifecycle State is OFF.

If these conditions do not all hold, the procedure shall not be followed. The results of following this procedure in this case are undefined.

Other than the initial state which led to the transition, this procedure is otherwise identical to that defined in *Boot (ATTACHED → ACTIVATED)*.

The following value is used in this procedure:

- The Interface ID of the Interface being rebooted is interface_id.

1. The AP shall exchange an *Greybus SVC Interface V_SYS Enable Operation* with the SVC. The intf_id field in the request payload shall equal interface_id.

   If the Operation fails, this procedure has failed. Go to step 8.

2. The AP shall exchange an *Greybus SVC Interface REFCLK Enable Operation* with the SVC. The intf_id field in the request payload shall equal interface_id.

   If the Operation fails, this procedure has failed. Go to step 7.

3. The AP shall exchange an *Greybus SVC Interface UNIPRO Enable Operation* with the SVC. The intf_id field in the request payload shall equal interface_id.

   If the Operation fails, this procedure has failed. Go to step 6.

4. The AP shall exchange an *Greybus SVC Interface Activate Operation* with the SVC. The intf_id field in the request payload shall equal interface_id.

   If the Operation fails, this procedure has failed. Go to step 5.

   If the Operation succeeds, this procedure has succeeded. The Interface is now ACTIVATED. Go to step 8.

5. The AP shall exchange a *Greybus SVC Interface UNIPRO Disable Operation* with the SVC. The intf_id field in the request payload shall equal interface_id.

6. The AP shall exchange a *Greybus SVC Interface REFCLK Disable Operation* with the SVC. The intf_id field in the request payload shall equal interface_id.

7. The AP shall exchange a *Greybus SVC Interface V_SYS Disable Operation* with the SVC. The intf_id field in the request payload shall equal interface_id.

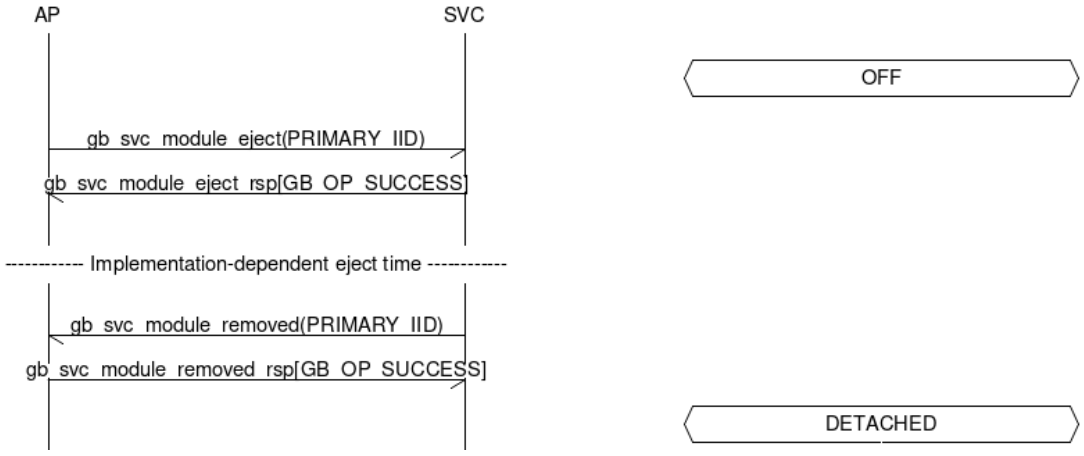
8. The procedure is complete and has succeeded or failed. If the procedure failed and all of the steps 5, 6, and 7 which were reached succeeded, the Interface is now OFF.

12.2.4 Eject (OFF → DETACHED)

Note: The content in this section is defined under the assumption that there is exactly one *AP Interface* in the Greybus System.

The results if there are multiple AP Interfaces are undefined.

The following procedure can be initiated by the AP when an Interface is OFF, in order to attempt to follow the "eject" transition from OFF to DETACHED.



To perform this procedure, the following conditions shall hold.

- The AP Interface and SVC shall have established a Connection implementing the *SVC Protocol*. This is the SVC Connection in this procedure.
- A Module shall be provided which is MODULE_ATTACHED.
- The Interface Lifecycle State is OFF for all Interfaces in the Module.

If these conditions do not all hold, the procedure shall not be followed. The results of following this procedure in this case are undefined.

The following value is used in this procedure:

- The Interface ID of the Primary Interface to the Module being ejected is primary_interface_id.

1. If the AP receives an *Greybus SVC Module Removed Operation* Request from the SVC with primary_intf_id field equal to primary_interface_id, the procedure has succeeded. Immediately go to to step 4.
2. The AP shall exchange an *Greybus SVC Module Eject Operation* with the SVC. The primary_intf_id field in the request payload shall equal primary_interface_id.

   If this Operation fails, the procedure has failed. Go to step 4.
3. After the SVC Interface Eject Response is received, the AP shall start a timer, for an implementation-defined duration.

   If the AP detects the timer has expired and has not received an SVC Module Removed Request from the SVC with primary_intf_id field equal to primary_interface_id, the procedure has failed. Go to the next step.
4. The procedure is now complete and has succeeded or failed. If the procedure succeeded, all Interfaces formerly present in the removed Module are now DETACHED. If the procedure failed, the Interfaces are all still OFF, and the Module is still MODULE_ATTACHED, and the Interfaces are all still OFF.

12.2.5 Mode Switching

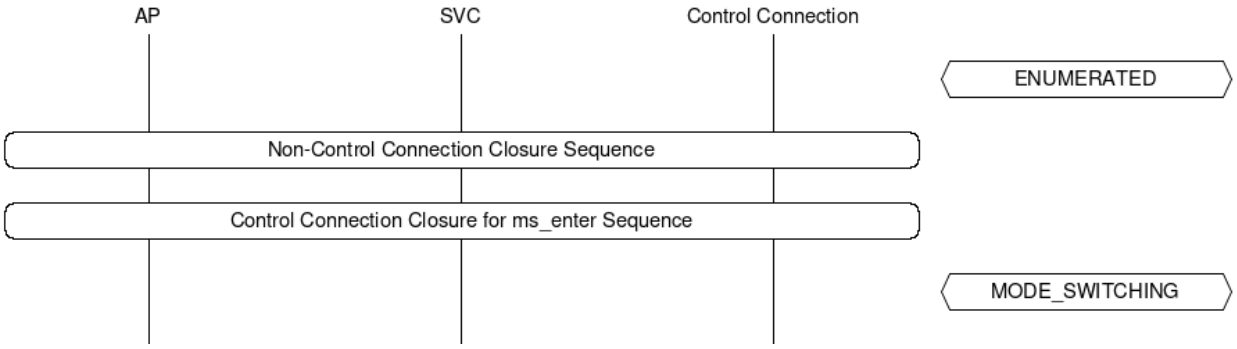
Mode Switch Enter (ENUMERATED → MODE_SWITCHING)

Note: The content in this section is defined under the following assumptions:

- there is exactly one *AP Interface* in the Greybus System.
- The Non-Control Connections given below are each between that AP Interface and another Interface in the System.

The results if there are multiple AP Interfaces, or in the case of non-AP to non-AP Interfaces, are undefined.

The following procedure can be initiated by the AP when an Interface is ENUMERATED, in order to attempt to follow the "ms_enter" transition from ENUMERATED to MODE_SWITCHING.



To perform this procedure, the following conditions shall hold.

- The AP Interface and SVC shall have established a Connection implementing the *SVC Protocol*. This is the SVC Connection in this procedure.
- An Interface shall be provided, whose Interface Lifecycle State is ENUMERATED.
- Zero or more additional Non-Control Connections shall be provided, which comprise all such established Connections involving the Interface, and shall each have been established by following the sequence defined in *Non-Control Connection Establishment*.

If these conditions do not all hold, the procedure shall not be followed. The results of following this procedure in this case are undefined.

The following values are used in this procedure:

- The AP Interface's ID is ap_interface_id.
- The Interface ID of the Interface entering MODE_SWITCHING is interface_id.

1. Through Protocol-specific means, the AP and Interface shall establish that the remaining steps in the Mode Switch Enter procedure shall be followed.
2. The sequence defined in *Non-Control Connection Closure* shall be followed to attempt to close all of the provided Non-Control Connections.

   If any attempt fails, this procedure has failed. The results are undefined.
3. The sequence defined in *Control Connection Closure for ms_enter* shall be followed to inform the Interface its Control Connection is closing and it is entering MODE_SWITCHING.

   If the sequence succeeds, this procedure has succeeded. The Interface is MODE_SWITCHING.

   If the sequence fails, this procedure has failed. The results are undefined.
4. The procedure is now complete and has either succeeded or failed.

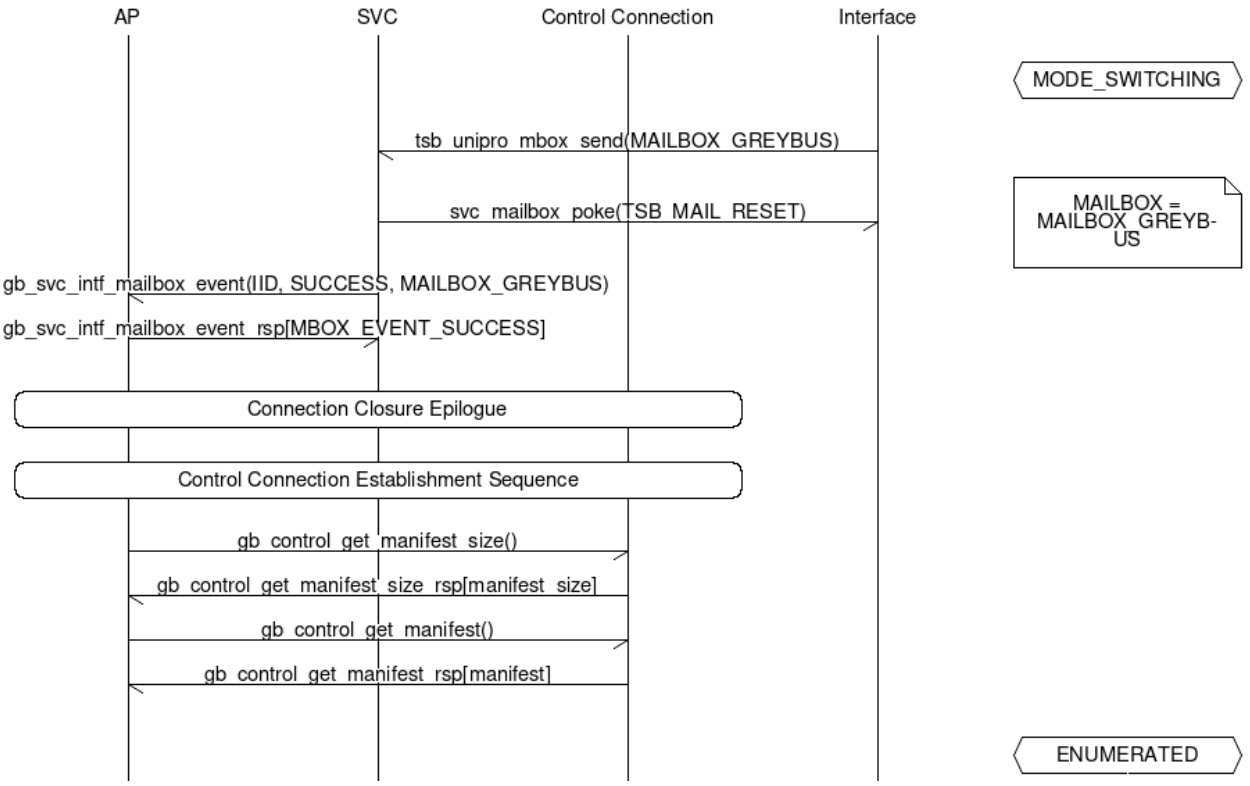
Mode Switch Exit (MODE_SWITCHING → ENUMERATED)

Note: The content in this section is defined under the following assumptions:

- there is exactly one *AP Interface* in the Greybus System.
- The Non-Control Connections given below were each between that AP Interface and another Interface in the System.

The results if there are multiple AP Interfaces, or in the case of non-AP to non-AP Interfaces, are undefined.

The following procedure can be initiated by the Interface when it is is MODE_SWITCHING, in order to attempt to follow the "ms_exit" transition from MODE_SWITCHING to ENUMERATED.



To perform this procedure, the following condition shall hold.

- An Interface shall be provided, whose Interface Lifecycle State is MODE_SWITCHING. The Interface shall have transitioned to the MODE_SWITCHING Lifecycle State by following the ms_enter procedure defined in *Mode Switch Enter (ENUMERATED → MODE_SWITCHING)*.
- Another value, ap_cport_id, shall also be provided. The AP Interface shall contain a CPort with CPort ID ap_cport_id. This CPort on the AP Interface shall not be part of an established UniPro connection.

If these conditions do not all hold, the procedure shall not be followed. The results of following this procedure in this case are undefined.

The following values are used in this procedure:

- The AP Interface ID is ap_interface_id.
- The Interface ID of the Interface which is MODE_SWITCHING is interface_id.

1. The Interface shall conclude any implementation-specific procedures needed while it is in the MODE_SWITCHING Lifecycle State, and write MAILBOX as described in *Greybus Control Mode Switch Operation*.
2. The SVC shall detect this write, and exchange an *Greybus SVC Interface Mailbox Event Operation* Operation with the AP. The intf_id field in the request payload shall equal interface_id.

   If the Operation is not successful, this procedure has failed. The results are undefined.
3. The *Connection Closure Epilogue* sub-sequence is followed. The Closing Connection for that sub-sequence is the Control Connection to the Module which was MODE_SWITCHING.

   If the sub-sequence succeeds, the Control Connection to the Interface is now closed.

   If the sub-sequence fails, this procedure has failed. The results are undefined.

4. The sequence to establish a Control Connection to the Interface described in *Control Connection Establishment* shall be followed.

   If the sequence fails, this procedure has failed. The results are undefined.

5. The AP shall exchange a *Greybus Control Get Manifest Size Operation* via the Control Connection. If the Operation is successful, the value of the manifest_size field in the response payload is interface_manifest_size.

   If the Operation fails, this procedure has failed. The results are undefined.

6. The AP shall exchange a *Greybus Control Get Manifest Operation* via the Control Connection. If the Operation is successful, the Manifest's value is interface_manifest.

   If the Operation fails, this procedure has failed. The results are undefined.

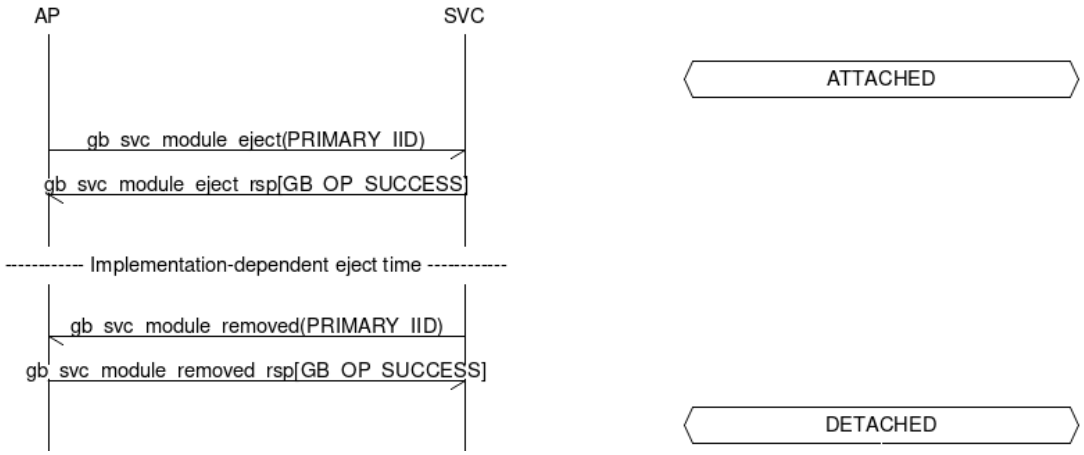
7. The AP shall perform implementation-defined procedures to parse the *components of the Manifest*.

   The procedure is now complete. The Interface is ENUMERATED once more.

No special provision is made within the Greybus Specification for recovery from failure. The AP and Interface may use implementation- or protocol-specific timeouts to detect errors and attempt to recover.

12.2.6 Error Handling

Early Eject (ATTACHED → DETACHED)



Early Power Down (ACTIVATED → OFF)

Make sure cleanup when jumping from failure to enumerate is covered:

- tear down routes
- destroy Device ID
- unipro, refclk, vsys from activation -> off

Mode Switch Fail (MODE_SWITCHING → ACTIVATED)

Forcible Removal (Any → DETACHED)

Appendix A

Firmware Lifecycle on ARA Phone Module (Informative)

This appendix describes the Firmware Lifecycle on an ARA Phone Module's Interface, that includes a bridge ASIC to communicate to the UniPro network. The term 'Interface' will be used by rest of this section for such an ARA Phone Module's Interface.

A.1 Firmware Types and Protocols

Firmware images required for an Interface can be classified broadly into two categories:

- *Interface Firmware*

  These are the Firmware Images that run on the bridge ASIC present on the ARA Phone Module.

- *Interface Backend Firmware*

  These are the Firmware Images that run on a Device Processor sitting behind the bridge ASIC, for example Camera.

The term Mode-Switch will be used by rest of this section, for referring to transition from one Interface Firmware Image to another Interface Firmware Image for an Interface using the *Greybus Control Mode Switch Operation*.

The *Firmware Download Protocol* can be used by an Interface to download Firmware Packages over UniPro.

The *Firmware Management Protocol* can be used by the Application Processor (AP) to prepare the Interface State to enter MODE_SWITCHING Interface Lifecycle State. The Firmware Management Protocol can also be used by the AP to update the Interface Backend Firmware Images for an Interface.

Ideally, every Interface Firmware Stage for the Interface shall contain a CPort for Firmware Management Protocol. Without that, the firmware wouldn't be able to load another firmware and boot into it.

A.2 ARA Boot Stages

The current design of Interface Firmware stages for an Interface on ARA Phone forces the Interface to have three or four stages as defined by Table A.1.

Boot Stage Name	Firmware Tag
boot ROM (S1)	Not Applicable
Stage 2 Loader	“s2l”
Stage 3 Firmware	“s3f”
Stage 3 Backend Firmware Updater	“s3_bfu”

Table A.1: ARA Boot Stages and Firmware-tag

One of the main purpose of the S2 Loader stage is to get the Interface hardware Authenticated. For security reasons, the AP may want to verify if a connected Module is authorized by Google to be part of the ARA phone. The AP and the Interface takes part in the Authentication dialog using the Component Authentication Protocol (CAP). The AP sends a CAP Message to the Interface which contains a cryptographically unpredictable message. The Interface decodes the same using a set of private keys burned into the Interface at the time of Manufacturing. Only an Authorized Interface Firmware can read these keys and get the Module authenticated.

Todo

Add Component Authentication Protocol (CAP) to Greybus Specifications.

The Backend Device Processors can only be made functional while the current Interface Firmware stage is S3F or S3-BFU, as the S2 Loader doesn't have any knowledge of the Backed Device Processors and it can't talk to them.

The Interface Firmware stages shall have the capability to Mode-Switch from:

- boot ROM to Stage 2 Loader (For future boot-ROMs only, boot ROM of ES3 chips is fixed as the chip is already taped out).
- Stage 2 Loader to another Stage 2 Loader Firmware Image (If S2L is updated).
- Stage 2 Loader to Stage 3 Interface Firmware.
- Stage 3 Interface Firmware to Stage 3 Backend Firmware Updater.
- Stage 3 Backend Firmware Updater to Stage 3 Interface Firmware.

A.3 Interface Manifest Layout

This section describes how the Interface *Manifest* received by the AP from an Interface over *Control Protocol* shall look like, in order to support Mode-Switch and updates to Interface Backend Firmware Packages.

The Manifest may contain other Bundles and CPorts as well, like Control CPort, etc..

Firmware Management Bundle (Bundle 1):

- class = 0x16
- (Mandatory) Firmware Management Protocol on CPort 1 talks over *Firmware Management Protocol*.
  - protocol = 0x18
- (Optional) Firmware Download Protocol on CPort 2 talks over *Firmware Download Protocol*.
  - protocol = 0x17
- (Optional) SPI Protocol on CPort 3 talks over *SPI Protocol*.

  - protocol = 0x0b

- (Optional) Component Authentication Protocol (CAP) on CPort 4 talks over CAP Protocol *Component Authentication Protocol*.
  - protocol = 0x19

A.4 Identify Current Interface Firmware Stage

Android userspace or the kernel running on the AP may be required to do different things based on the current Firmware Stage of an Interface. For example, in S2L stage, the AP may Authenticate the Interface using CAP Protocol or update bridge ASIC's SPI flash using SPI Protocol, etc..

And so can be quite useful for the AP to know the current implementation defined Interface Firmware Stage.

This can be retrieved by the AP from the Interface using *Greybus Firmware Management Interface Firmware Version Operation*. The Interface shall return an implementation defined "firmware_tag" to the AP, which can be used by the AP to know the current boot stage. For example, in the current implementation we can keep its values as "s2l", "s3f", "s3_bfu".

A.5 Prepare an Interface Firmware to enter MODE_SWITCHING Lifecycle State

The AP may want to Mode Switch to another Interface Firmware Stage. For that it first needs to ask the Interface to load and validate the next stage Firmware package. Following sequence of events describes how that can be achieved to Mode-Switch from S2L to S3F Interface Firmware stage, by first downloading the Firmware Package over UniPro.

- The AP initiates a *Greybus Firmware Management Interface Firmware Load and Validate Operation* over Firmware Management CPort and passes request-id as '1', firmware-tag as "s3f", and load-method as FIRMWARE_LOAD_METHOD_UNIPRO.
- The Interface responds to the request from the AP immediately and initiates a *Greybus Firmware Download Find Firmware Operation* request over Firmware Download CPort and passes it the firmware-tag received from the AP in Load and Validate Operation.
- The AP finds the requested firmware package and responds with GB_OP_SUCCESS in the status of the response header and provides firmware size as 16380 bytes and unique firmware-ID as 0x05.
- The Interface initiates a number of *Fetch Firmware Operations* using firmware-ID 0x05 and loads the entire firmware package block by block.
- The Interface initiates a *Greybus Firmware Download Release Firmware Operation* using firmware-ID 0x05 to request the AP to release the firmware.
- The Interface parses the firmware image header and validates its signature in an implementation defined way.
- The Interface initiates a *Greybus Firmware Management Interface Firmware Loaded Operation* to the AP and passes the request-id as '1' (same as that received from the AP), status of validation and major/minor version of the loaded firmware.
- The AP finds that the Interface has verified the signatures of the Interface Firmware Package.
- The Interface has an Interface Firmware Package with now and needs to Mode Switch into that.
- The AP starts tearing down of the connections and issue a *Greybus Control Mode Switch Operation*.

A.6 Update S2L and S3F in bridge ASIC's SPI Flash

Lets consider that the Interface is running its S3F stage currently. Following sequence of events will lead to updating Images in the bridge ASIC SPI flash.

- Android receives a MSP update for the Interface and downloads it from Android Play-store (or whatever).
- AP receives the current Interface Firmware version using *Greybus Firmware Management Interface Firmware Version Operation*.
- AP compares that to the version of the firmware it has downloaded and decides if an update is required or not.
- If an update is required, the AP prepares the Interface to Mode Switch into S2L Firmware Stage as described in the *Prepare an Interface Firmware to enter MODE_SWITCHING Lifecycle State* section.
- Once the AP has Mode-Switched to S2L Firmware Stage, the AP will get an additional SPI CPort and the AP can update the SPI flash using *SPI Protocol*.
- If the S2 Loader firmware is also updated, and then we may need to Mode-Switch to the new S2L Firmware Image first, which will eventually Mode-Switch into the S3F. Otherwise, we can directly Mode-Switch from old S2L to the S3F Image. All Mode-Switch operations can be done as defined in *Prepare an Interface Firmware to enter MODE_SWITCHING Lifecycle State* section.

A.7 Update Device Processor Firmware Images

This is perhaps the most complicated of all the use cases we may have.

Consider that the bridge ASIC is running its S3F Interface Firmware Stage and the device processors are running their respective firmware images.

Following sequence of events will lead to updating device firmware images.

- The AP receives the version of the individual Device Processor Firmware Images using *Greybus Firmware Management Interface Backend Firmware Version Operation* over the Firmware Management CPort.
- If the AP finds at least one Device Processor firmware image that needs update, it Mode-Switches the Interface to S3-BFU Interface Firmware Stage as described in *Prepare an Interface Firmware to enter MODE_SWITCHING Lifecycle State* section.
- This is important to guarantee that the Interface and its device processors aren't being used by the AP concurrently while the update in progress.
- During the above Mode Switch, the Device Processors aren't required to be reseted as power to them is never cut-off on Mode Switch, but this is going to be implementation defined really.
- The new Interface personality provided by the S3-BFU will only contain the CPorts necessary for firmware update, i.e. Firmware Management CPort and Firmware Download CPort.
- Once the S3-BFU Interface Firmware Stage has booted, the AP (again) starts again matching versions of all the backend device processor firmwares using *Greybus Firmware Management Interface Backend Firmware Version Operation* over the Firmware Management CPort, as it may not have cached them earlier.
- As soon as a mismatch in version is found between the backend firmware on the Interface and the version available with the AP, the AP starts updating them by issuing *Greybus Firmware Management Interface Backend Firmware Update Operation* requests over the Firmware Management CPort.

- On receiving these requests, S3-BFU Interface Firmware Stage will immediately respond to the AP and start downloading the specific backend device processor firmware using *Firmware Download Protocol* as explained earlier.
- Once the individual device processor firmware is downloaded by the bridge ASIC, it will flash that to the internal flash memory in an implementation dependent way and send a *Greybus Firmware Management Interface Backend Firmware Updated Operation*.
- Similarly all the device processor firmware images, that the AP wants to update or reflash, can be updated.
- Now the AP needs to Mode-Switch the Interface to normal S3F Interface Firmware Stage personality as described in *Prepare an Interface Firmware to enter MODE_SWITCHING Lifecycle State* section.

Bibliography

[HID01] Device Class Definition for Human Interface Devices (HID), Version 1.11, USB Implementers' Forum, 27 June 2001.

[MIPI01] Specification for UniPro, version 1.6, MIPI Alliance Inc., 6 August 2013.

[MIPI02] Specification for M-PHY, version 3.0, MIPI Alliance, Inc., 26 July, 2013.

[US-ASCII] ANSI_X3.4-1968 encoding standard.

[JEDEC-UFS] JESD220B, Universal Flash Storage (UFS), JEDEC Solid State Technology Association, September 2013.

[CSI-2] MIPI Alliance Specification for CSI-2, version 1.3, MIPI Alliance, Inc., 29 May 2014.

[CSI-3] MIPI Alliance Specification for CSI-3, version 1.1, MIPI Alliance, Inc., 2012.

[FIPS180] FIPS PUB 180-4, Secure Hash Standard, National Institute of Standards and Technology, August 2015

[RSA] Internet RFC 3447, Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1 https://tools.ietf.org/html/rfc3447

[ED25519] Internet RFC 7748, Elliptic Curves for Security, section 4.1, https://tools.ietf.org/html/rfc7748

[ED448] Internet RFC 7748, Elliptic Curves for Security, section 4.2, https://tools.ietf.org/html/rfc7748

[IEEE745] IEEE Standard for Floating-Point Arithmetic," in IEEE Std 754-2008 , vol., no., pp.1-70, Aug. 29 2008.