Data-Over-Cable Service Interface Specifications
DCA - MHAv2

Generic Control Plane Specification

CM-SP-GCP-I03-170524

ISSUED

DISCLAIMER

This document is furnished on an "AS IS" basis and neither CableLabs nor its members provides any representation or warranty, express or implied, regarding the accuracy, completeness, noninfringement, or fitness for a particular purpose of this document, or any document referenced herein. Any use or reliance on the information or opinion in this document is at the risk of the user, and CableLabs and its members shall not be liable for any damage or injury incurred by any person arising out of the completeness, accuracy, or utility of any information or opinion contained in the document.

CableLabs reserves the right to revise this document for any reason including, but not limited to, changes in laws, regulations, or standards promulgated by various entities, technology advances, or changes in equipment design, manufacturing techniques, or operating procedures described, or referred to, herein.

This document is not to be construed to suggest that any company modify or change any of its products or procedures, nor does this document represent a commitment by CableLabs or any of its members to purchase any product whether or not it meets the characteristics described in the document. Unless granted in a separate written agreement from CableLabs, nothing contained herein shall be construed to confer any license or right to any intellectual property. This document is not to be construed as an endorsement of any product or company or as the adoption or promulgation of any guidelines, standards, or recommendations.

Document Status Sheet

Document Control Number:	CM-SP-GCP-I03-170524			
Document Title:	Generic Control Plane Specification			
Revision History:	I01 - Released 06/15/2015 I02 - Released 05/12/2016 I03 - Released 05/24/2017			
Date:	May 24, 2017			
Status:	~~Work in Progress~~	~~Draft~~	**Issued**	~~Closed~~
Distribution Restrictions:	~~Author Only~~	~~CL/Member~~	~~CL/ Member/ Vendor~~	**Public**

Key to Document Status Codes

Work in Progress	An incomplete document designed to guide discussion and generate feedback, and may include several alternative solutions for consideration.
Draft	A document in Specification format considered largely complete, but lacking review by Members and Technology Suppliers. Drafts are susceptible to substantial change during the review process.
Issued	A generally public document that has undergone Member and Technology Supplier review, cross-vendor interoperability, and is available for Certification testing. Issued Specifications are subject to the Engineering Change (EC) Process.
Closed	A static document, reviewed, tested, validated, and closed to further ECs.

Trademarks

Table of Contents

List of Figures

List of Tables

This page has been left blank intentionally

1 SCOPE

1.1 Introduction and Purpose

This document outlines a Generic Control Plane (GCP) encapsulation protocol. GCP sets up a control plane tunnel over a generic transport protocol such as TCP. TCP provides a reliable transport, message acknowledgement, message segmentation for large messages, and windowing so that multiple messages can be outstanding at once.

GCP imitates the major functionality that exists over a hardware bus between a CPU and a peripheral interface chip. It can read and write parameters, either directly to and from registers or with higher layer data structures. It can reset the device, power it up, or power it down. The device can send the equivalent of a hardware interrupt with the Notify command. GCP allows chips that were previously embedded to be located across a network interface such as Ethernet.

It is assumed that the remote device has enough intelligence to connect to a network, obtain an IP address, and run a network stack.

1.2 MHAv2 Interface Documents

A list of the documents in the MHAv2 family of specifications is provided below. For updates, refer to http://www.cablelabs.com/specs/specification-search/.

Designation	Title
CM-SP-R-PHY	Remote PHY Specification
CM-SP-R-DEPI	Remote Downstream External PHY Interface Specification
CM-SP-R-UEPI	Remote Upstream External PHY Interface Specification
CM-SP-GCP	Generic Control Plane Specification
CM-SP-R-DTI	Remote DOCSIS Timing Interface Specification
CM-SP-R-OOB	Remote Out-of-Band Specification
CM-SP-R-OSSI	Remote PHY OSS Interface Specification

NOTE: MHAv2 does not explicitly use any of the original Modular Headend Architecture specifications.

1.3 Conventions and Requirements

In this specification the following convention applies any time a bit field is displayed in a figure. The bit field should be interpreted by reading the figure from left to right, then from top to bottom, with the MSB being the first bit so read and the LSB being the last bit so read.

Encoding for message fields is Big-Endian.

Throughout this document, the words that are used to define the significance of particular requirements are capitalized. These words are:

"MUST"	This word means that the item is an absolute requirement of this specification.
"MUST NOT"	This phrase means that the item is an absolute prohibition of this specification.
"SHOULD"	This word means that there may exist valid reasons in particular circumstances to ignore this item, but the full implications should be understood and the case carefully weighed before choosing a different course.
"SHOULD NOT"	This phrase means that there may exist valid reasons in particular circumstances when the listed behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.
"MAY"	This word means that this item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because it enhances the product, for example; another vendor may omit the same item.

Since GCP is a common protocol that may be used in multiple applications, each application needs to determine what set or subset of GCP functions will be supported within that application. As a result, this document will not contain many mandatory statements.

2 REFERENCES

At the time of publication, the editions indicated were valid. All references are subject to revision, and users of this document are encouraged to investigate the possibility of applying the most recent editions of the documents listed below. References are either specific (identified by date of publication, edition number, version number, etc.) or non-specific. For a nonspecific reference, the latest version applies.

2.1 Normative References

In order to claim compliance with this specification, it is necessary to conform to the following standards and other works as indicated, in addition to the other requirements of this specification. Notwithstanding, intellectual property rights may be required to use or implement such normative references.

[Vendor ID] Refers to IETF RFC 3232 "Assigned Number" by the IETF, Jan 2002. This spec refers to the IANA web page which is http://www.iana.org/assignments/enterprise-numbers.

2.2 Informative References

This specification uses the following informative references.

[IANA] Internet Assigned Numbers Authority, http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml

[MULPIv3.0] DOCSIS 3.0 MAC and Upper Layer Protocols Interface Specification, CM-SP-MULPIv3.0-I29-151210, December 10, 2015, Cable Television Laboratories, Inc.

[MULPIv3.1] DOCSIS 3.1 MAC and Upper Layer Protocols Interface Specification, CM-SP-MULPIv3.1-I11-170510, May 10, 2017, Cable Television Laboratories, Inc.

[MULPI] Refers to both [MULPIv3.0] and [MULPIv3.1].

[RFC 3931] IETF RFC 3931, Layer Two Tunneling Protocol - Version 3 (L2TPv3), March 2005.

2.3 Reference Acquisition

- Cable Television Laboratories, Inc., 858 Coal Creek Circle, Louisville, CO 80027;
  Phone +1-303-661-9100; Fax +1-303-661-9199; http://www.cablelabs.com
- Internet Assigned Numbers Authority (IANA), http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml
- Internet Engineering Task Force (IETF) Secretariat, 48377 Fremont Blvd., Suite 117, Fremont, California 94538, USA, Phone: +1-510-492-4080, Fax: +1-510-492-4001, http://www.ietf.org
- SCTE - Society of Cable Telecommunications Engineers Inc., 140 Philips Road, Exton, PA 19341; Phone: +1-610-363-6888 / 800-542-5040; Fax: +1-610-363-5898; http://www.scte.org/

3 TERMS AND DEFINITIONS

This specification uses the following terms:

Endian	Big-endian systems store the most significant byte of a word in the smallest address and the least significant byte is stored in the largest address. Little-endian systems, in contrast, store the least significant byte in the smallest address.
GCP Device	Either a GCP Master or GCP Slave.
GCP Master	The controlling entity in the GCP communication.
GCP Slave	The device that is being controlled in the GCP communication.
Quadrature Amplitude Modulation (QAM)	A modulation technique in which an analog signal's amplitude and phase vary to convey information, such as digital data.
Registered access	The data is exchanged directly by programming (read/write) to the registers on a device.
Structured access	The data is transferred in a structured, predefined construct, such as a TLV tuple.
Upstream Channel Descriptor (UCD)	The MAC Management Message used to communicate the characteristics of the upstream physical layer to the cable modems.

4 ABBREVIATIONS AND ACRONYMS

This specification uses the following abbreviations:

AVP	Attribute Value Pair
CM	Cable Modem
CMTS	Cable Modem Termination System
EDR	Exchange Data Register
EDS	Exchange Data Structure
FIFO	First In, First Out (buffer)
GDM	GCP Device Management
IANA	Internet Assigned Numbers Authority
IP	Internet Protocol
L2TPv3	Layer 2 Transport Protocol version 3
LSB	Least Significant Bit
MSB	Most Significant Bit
MTU	Maximum Transmission Unit
MWR	Mask Write Register
OBFL	Onboard Failure Log
PDU	Protocol Data Unit
PHY	Physical Layer
QAM	Quadrature Amplitude Modulation
TCP	Transmission Control Protocol
TLV	Type Length Value
UCD	Upstream Channel Descriptor
UDP	User Datagram Protocol

5 TECNICAL OVERVIEW

5.1 Introduction

GCP is a generic control plane protocol that exists between a master entity and a slave entity. The intent is for the master entity to control the slave entity. GCP is able to reuse control plane concepts that are defined elsewhere but used here in a new context.

It is typical for remote devices to require configuration. If the technology that is in a new remote device has been well-defined in previous devices, then the previous data structures can be reused with GCP.

For example, if a PHY technology like QAM (Quadrature Amplitude Modulation) has been well-defined in existing protocols like [MULPI], then the TLV (Type Length Value) data structures in [MULPI] could be reused in GCP in this new device. This also provides the possibility for existing state machines and software code to be leveraged.

When GCP is used to tunnel the data structures from another protocol, the following nomenclature can be used:

GCP(tag)

where the tag is representative of the data structure being leveraged. For example, if the TLV data structures from a DOCSIS UCD (Upstream Channel Descriptor) message (see [MULPI]) are used over GCP, the result can be referred to as GCP(UCD).

5.2 Functional Decomposition

GCP has a master entity and a slave entity. The GCP master initiates reads and writes to the GCP slave. The GCP slave can initiate a notify message to get the attention of the GCP master entity. This is shown in Figure 1.

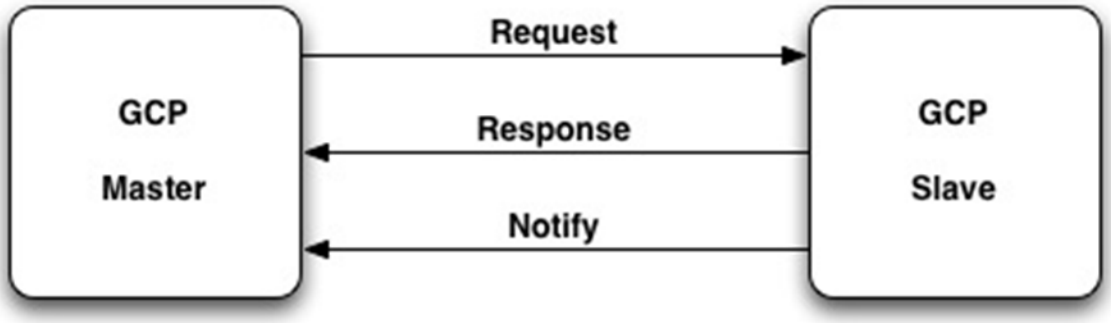

Figure 1 - GCP Block Diagram

GCP also has a peer-to-peer mode that permits both endpoints to simultaneously be both masters and slaves. The peer-to-peer mode is really just two independent master-slave nodes, in opposite directions, on the same port number.

It is acceptable for systems to implement a subset of the GCP message set. GCP is a framework upon which other systems and protocol infrastructures can be built.

GCP can be used in any architecture where devices are connected with a network. It could be used within a chassis or across a continent. GCP uses a protocol such as TCP/IP as its ultimate transport, so it is independent of both network topology and network technology. Specifically, although Ethernet is often used as the Layer 2 framing, any Layer 2 framing could be used, or be replaced on a per-hop network basis.

5.3 Peer-to-Peer Operation

GCP is a master-slave protocol. GCP is allowed to run separately in both directions across the same network interface, such that both end devices are both masters and slaves. The combination of two master-slave protocols running in opposite directions provides a peer-to-peer operation.

Since GCP messages are uniquely numbered, there is no confusion between request and response messages. Any request message not supported that arrives from either direction will get a null response.

Some applications that might use the peer-to-peer mode of operation would be:

- Where both sides use a memory interface.
- Where one side initiates register request messages, but both sides need to initiate data structure messages.

6 GCP PROTOCOL MESSAGES

6.1 Protocol Description

The GCP Master uses a request message to address a GCP Slave. The GCP Slave always responds with a response message. There is no follow-on ACK message since GCP uses a reliable transport. If the GCP slave wants attention, it will issue a Notify message.

6.1.1 Encapsulation

GCP messages follow a TLV (type - length - value) format. The format of the GCP message is shown in Figure 2.

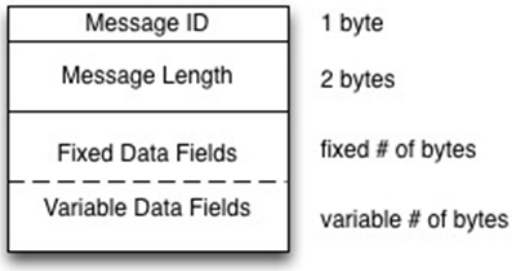

Figure 2 - GCP Message Structure

The Type field is called the Message Identifier and is a one-byte value that defines the message type. The most significant bit is set to a zero in the request message and may be set to a one in the response to indicate an exception condition. A Message Identifier of zero is not valid. The least significant bit is usually set to a zero for a request message and a one for a response message.

The Length field is a two-byte field and is the number of bytes contained starting after the length field and ending at the end of the message. This includes all fixes and variable-length fields following the length field.

There may be one or more fixed parameter fields that are unique to each message. The fixed length fields of GCP often contain a transaction ID and a mode field:

- The Transaction ID field uniquely identifies each transaction. A transaction includes a request and its matching response.
- The Mode field is used for modifying the action of the message. The Mode field is unique per message.

The message may end with a variable length data field that is unique to each message.

6.1.2 Message Length

When GCP is carried over TCP, the length of the message or group of messages does not matter since TCP will segment the message as needed to fit into an IP packet that conforms to the MTU (Maximum Transmission Unit) of the networking interface. When other transport protocols such as UDP or L2TPv3 are used, then a maximum length needs to be observed.

In practice, there may be other system restrictions beyond the scope of this specification that limit the maximum size of a GCP message. Refer to Message Length Considerations for more discussion on this topic.

6.1.3 Port and Channel Conventions

The GCP slave device being addressed may have one or more physical ports. Each port may have one or more channels. Several of the GCP messages allow specific access to a particular port and/or channel.

Additionally, the GCP header in the network protocol payload has an additional Unit Identifier field (see Section 7.1.3) that can be used for device selection within the GCP Slave.

The Port and Channel fields contain a direct reference to the area of the device for which the data structure should be applied. A wild card is defined for these fields, and is described in Table 1.

Table 1 - Wild Card Value for the Port and Channel Fields

Wild Card Value	Meaning	Notes
0xFFFF	Don't care	The specific port or channel assignment does not apply. Attributes are to be applied to all ports or channels.

The wild card value allows the number of signaling messages to be reduced when there is common configuration across many ports and channels. Another scenario where a wild card may be used is if the port and/or channel information is embedded in the data structure and these explicit fields are not required for operation.

Examples of how to use these values are shown in Table 2.

Table 2 - Port and Channel Reserved Values

Port	Channel	Mnemonic	Comments
M	N	M, N	Apply message to Port M, Channel N.
0xFFFF	N	*.N	Apply message to channel N of all ports.
M	0xFFFF	M.*	Apply message to all channels of Port M.
0xFFFF	0xFFFF	*.*	Apply message to all channels on all ports.

The numerical values of the ports and channels are application-dependent and not dictated by GCP. For example, a general message could be used to assign specific port and channel numbers. Subsequent messages could then use these assigned port and channel numbers. Applications may also subdivide the port and/or channel number space any way they want to represent different types of ports, and/or groups of channels and subchannels.

6.1.4 Device Management

Since GCP is a protocol for managing remote devices, GCP contains some basic functions for managing that device. Among these functions are ping, reset, and power management. These modes are coordinated using the Device Management message and the Notify message.

The modes available for Device Management are shown in Table 3.

Table 3 - Device Management Modes

Mode	Description
Null	The device should respond but take no other action. This message can be used as a ping, a keep-alive, or a test function.

Mode	Description
Cold Reset	The device should respond and then perform a full reset. When it comes back on-line, it should issue a Notify with the mode set to Cold Reset.
Warm Reset	The device should respond, reset its session information, but retain its current learned configuration. The definition of a warm reset is ultimately application-dependent. When it comes back on-line, it should issue a Notify with the mode set to Warm Reset.
Standby	The device should respond and then go to a standby state. The device is in a power-reduced state and can come back to operational mode based upon its own criteria. If the device transitions to standby on its own, it should first issue a Notify with mode set to Standby.
Wake-Up	The device should respond and exit standby mode. When it comes back on-line, or if it wakes up on its own, it should issue a Notify with the mode set to Wake-Up.
Power-Down	The device should respond and then power down. If the device powers up on its own, it should issue a Notify message with Power-Up set.
Power-Up	The device should respond and power-up. When it finishes its power-up cycle, it should issue a Notify message.

These commands are only effective if there is an IP address to communicate with and that address is still valid. The Wake-Up and Power-Up messages use the last known IP address or a newly-assigned network address.

The Standby and Power-Down modes require that there is a circuit that is monitoring the network connection and maintaining network addressing so the device can be commanded to exit these states.

6.1.5 Responding to Unsupported Messages

GCP as a protocol will likely evolve. Furthermore, various devices may only need a subset of the GCP messages. In GCP, the rule is that when a message is received that is unknown or unsupported, a specific null response is generated with the format shown in Table 4.

Table 4 - General Error Responses

Description	Length	Contents
Message ID	1 byte	0x81 + Message ID from Request
Message Length	2 bytes	3
Transaction ID	2 bytes	Same as request
Return Code	1 byte	0x01

Since GCP request and response messages tend to differ in the LSB, the GCP Master MUST accept an unsupported message error response with either the same message number or the message number with the LSB set (incremented by one).

6.1.6 Transaction Management

6.1.6.1 Transaction IDs

All GCP messages work in pairs. There is a request and the response. These two messages form a transaction. Each transaction has a transaction ID.

The transaction ID is unique within a period of time. The definition of this period of time is application-dependent and is not specified by GCP. The period of time chosen should exceed the longest expected outstanding transaction.

The transaction ID is unique per message type. That means that with a message type, there will not be a duplicate transaction ID within the application-defined window of time. There may be overlapping transaction IDs between message types. This allows the GCP Slave to choose a number space for transaction IDs that are independent from the GCP Master number space for its transaction IDs.

The transaction ID is unique within a transport session, such as a TCP connection. If a GCP Slave has multiple GCP masters, then the transaction IDs will be considered as separate number spaces because they will be associated with different TCP connections.

There may be one or more messages per packet. In addition to a transaction ID per message, there is a transaction ID per packet (refer to Section 7.1.3). Generally, the transaction ID per message is used when the GCP messages are randomly multiplexed into a packet and transported. If the GCP messages are sent as a set, then the packet transaction ID can be used instead.

A transaction ID of all zeroes is reserved to indicate that the transaction ID should be ignored. The GCP Master can choose to not use the message transaction ID, the packet transaction ID, or both.

6.1.6.2 Network Outage

Since GCP is delivered over a reliable protocol such as TCP, TCP will guarantee delivery of both the request and the response message. If there is a network error that prevents delivery of these messages, TCP will indicate that the network session has been lost. In the event of a lost TCP session, the GCP Master and GCP Slave message state machines should close out all pending transactions.

6.1.6.3 Transaction Time-Out

Both the GCP Master and GCP Slave devices can initiate a request message, depending upon the message type.

In some implementations, it may be possible for the GCP Device to reliably receive a message from TCP and then lose that message internally. If so, the GCP Device that initiated a transaction by sending a request message SHOULD implement a transaction timer so that it can internally declare a failure when a response message is not received.The value of that timer is application-dependent and is not specified by GCP. The intent of this timer is not to replace or supplement the inherent reliability of the transport, but to allow for internal failure of the receiving GCP Device.

In the event of a missing response message, the sending GCP Device SHOULD close the current transaction and start a new transaction rather than resending the message request with the previous transaction ID. This prevents the existence of duplicate messages and permits a more orderly operation of the network.

A race condition may occur if the response message timeout at the sending GCP Device is too quick and the response time from either the GCP Slave or the network in between is too slow. This will result in late response messages and redundant transactions. Separate transaction IDs will allow these transactions to be managed appropriately. The GCP Device SHOULD adjust its timer accordingly or take some other application-dependent action.

6.1.7 Data Path Width

GCP supports different width data paths on the master and slave. Due to the cost sensitivity of the slave device in certain markets, the GCP Slave device may have an 8-bit, 16-bit, or 32-bit data path. The same could be true for a GCP Master device, although a 32-bit master device may be more common. This provides at least three core scenarios:

- 32-bit Master and 32-bit Slave
- 32-bit Master and 16-bit Slave
- 32-bit Master and 8-bit Slave

Since GCP is typically run over a serialized network connection, in general, there is an automatic conversion that occurs between these two environments. To help manage the differences in data width, the GCP register commands allow the internal data bus width of the originating data field to be specified.

6.1.8 Performance Considerations

The embedded processor in the GCP Slave device may have limited control plane bandwidth. The GCP Master SHOULD set the TCP window size so that the GCP Slave CPU is not overloaded. Typical TCP window sizes are 15,000 bytes. GCP does not require TCP slow start.

The GCP Master can choose to suppress the response message. This may be useful when sending a large number of messages in a short timeframe such as during system reset. Multiple GCP messages may be grouped together into a single TCP message for a more efficient transport, or sent one message per TCP packet for the faster response time.

6.1.9 GCP Structured Message Summary

Structured access uses a data construct such as a TLV to program functionality in the GCP Slave. Structured access allows for more hardware independence and is more of an operational model.

The messages used in GCP and their Message Identifiers are shown in Table 5.

Table 5 - GCP Structured Message Summary

REQ	RSP	Error	Function	Initiator
2	3	131	Notify	Slave
4	5	133	Device Management	Master
6	7	135	Exchange Data Structures (EDS)	Master

The messaging numbering has been chosen such that the LSB is zero for the request message and a one for the response message. The MSB is a '0' for a request and for a normal response and a '1' for an error message.

6.1.10 GCP Register Message Summary

GCP messages fall into two general categories: structured and registered. Register access uses direct register reads and writes. Register access can be useful for diagnostics or low-level driver level access. The Notify and Device Management messages can also be used with systems that are only register message based.

The messages used in GCP and their Message Identifiers are shown in Table 6.

Table 6 - GCP Register Message Summary

REQ	RSP	Error	Function	Initiator
16	17	145	Exchange Data Register (EDR)	Master
18	19	147	Mask Write Register (MWR)	Master

GCP unstructured (aka, registered) access does not support an explicit and protected read-modify-write operation. A read-modify-write operation is often used in software drivers when a part of a register is going to be updated and the rest is to remain the same. GCP leaves it up to the higher layer software drivers to ensure that if it does a read-modify-write to a register location, that no other software process writes to the same register location during the read-modify-write. As a quicker and secure alternative to read-modify-write, GCP includes a Mask-Write operation.

6.1.11 GCP Slave Software Update

If the GCP Slave device requires software updates, there are at least four techniques that can be used with GCP:

1. Use FTP, TFTP, or a similar protocol to transfer a software image file toward the GCP Slave. The transfer could be initialized using either the GCP structured commands or register commands. This is a push operation.

2. Provide a network address to the GCP Slave and instruct it to fetch an image from the network using FTP, TFTP, or a similar protocol. The transfer would be initialized using either the GCP structured commands or register commands. This is a pull operation.

3. Write the software image to a FIFO register using the FIFO mode of the EDR command.

4. Provide a network address for a software image download to the GCP Slave with a DHCP option when the GCP Slave initializes. This operation is independent from GCP.

6.2 Structured Access

6.2.1 Notify Message

Table 7 - Structured Access Notify Request

Description	Length	Contents	
Message ID	1 byte	2	
Message Length	2 bytes	8 + N	
Transaction ID	2 bytes	Unique value	
Mode	1 byte	bit 7: bit 6: bit 5-0:	0 = Send normal response 1 = Suppress normal response, 0 = Event data is text 1 = Event data is raw; Reserved. Set to 0.
Status	1 byte	0 - Null (default) 1 - Cold Reset 2 - Warm Reset 3 - Standby 4 - Wake-up 5 - Power-down 6 - Power-up 7 to 255 - Reserved	
Event Code	4 bytes	0x00000000 to 0xFFFFFFFF	
Event Data	N bytes	Optional field. Application dependent data. Text or raw format.	

Table 8 - Structured Access Notify Normal Response

Description	Length	Contents
Message ID	1 byte	3
Message Length	2 bytes	7
Transaction ID	2 bytes	Same as request
Mode	1 byte	0
Event Code	4 bytes	0x00000000 to 0xFFFFFFFF

Table 9 - Structured Access Notify Error Response

Description	Length	Contents
Message ID	1 byte	131
Message Length	2 bytes	3
Transaction ID	2 bytes	Same as request
Return Code	1 byte	See Section 6.4

This message is intended to mirror a hardware interrupt from the GCP slave. The use of a status field is defined in Section 6.1.4. The default status is null. A value other than null is only sent when there is a status change. The contents of the message contain an event code. The definition of the event code is implementation-dependent and may be bit, byte, or word based.

The Notify message may have a variable length text field associated with it. The existence of this field can be determined by examining the length of the Notify message. The Event Data field is used for a text message or a raw data message that is associated with the event code. A bit in the Mode field indicates the format of the data field.

The content of the data field is associated with the event code and is application-dependent. These text messages may be useful for including in a system log or an onboard failure log (OBFL).

Note that if the GCP Master does not respond to a Notify message, the transaction timer in the GCP Slave will expire and the GCP Slave will re-issue the Notify. This prevents Notify messages from going unnoticed.

6.2.2 GCP Device Management (GDM)

The Device Management message defines basic and common powering functions that can be used prior to manage the end device.

Table 10 - GCP Device Management Request

Description	Length	Contents	
Message ID	1 byte	4	
Message Length	2 bytes	8	
Transaction ID	2 bytes	Unique value	
Mode	1 byte	bit 7: bit 6-0:	0 = Send normal response 1 = Suppress normal response, Reserved. Set to 0.
Port	2 bytes	0 to 0xFFFF	
Channel	2 bytes	0 to 0xFFFF	
Command	1 byte	0 - Null (default) 1 - Cold Reset 2 - Warm Reset 3 - Standby 4 - Wake-up 5 - Power-down 6 - Power-up 7 to 255 - Reserved	

Table 11 - GCP Device Management Normal Response

Description	Length	Contents
Message ID	1 byte	5
Message Length	2 bytes	4
Transaction ID	2 bytes	Same as request
Mode	1 byte	0
Return Code	1 byte	0

Table 12 - GCP Device Management Error Response

Description	Length	Contents
Message ID	1 byte	133
Message Length	2 bytes	3
Transaction ID	2 bytes	Same as request
Return Code	1 byte	See Section 6.4

The mode field is explained in Section 6.1.4. Port and channel usage is described in Section 6.1.3.

There is no explicit command for device identification. Device discovery and identification is application-dependent and is not specified by GCP.

6.2.3 Exchange Data Structures (EDS)

Table 13 - Exchange Data Structures Request

Description	Length	Contents
Message ID	1 byte	6
Message Length	2 bytes	12 + N
Transaction ID	2 bytes	Unique value
Mode	1 byte	0x00
Port	2 bytes	0 to 0xFFFF
Channel	2 bytes	0 to 0xFFFF
Vendor ID	4 bytes	0 to 0xFFFF
Vendor Index	1 byte	0 to 255
Data Structures	N bytes	Sending Data

Table 14 - Exchange Data Structures Normal Response

Description	Length	Contents
Message ID	1 byte	7
Message Length	2 bytes	12 + M
Transaction ID	2 bytes	Same as request
Mode	1 byte	0x00
Port	2 bytes	0 to 0xFFFF
Channel	2 bytes	0 to 0xFFFF
Vendor ID	4 bytes	0 to 0xFFFF
Vendor Index	1 byte	0 to 255
Data Structures	M bytes	Returned Data

Table 15 - Exchange Data Structures Error Response

Description	Length	Contents
Message ID	1 byte	135
Message Length	2 bytes	3
Transaction ID	2 bytes	Same as request
Exception Code	1 byte	See Section 6.4

Data structures could potentially be larger than the MTU of an IP packet. TCP will automatically segment a message like this (if necessary) across multiple IP packets.

Port and channel usage is defined in Section 6.1.3.

The Vendor ID field indicates what organization has defined the specific data structures to be used. The vendor identification number is defined in [Vendor ID]. The Vendor Index is unique to the vendor and chooses a data structure within the Vendor's definition. The combination of the Vendor ID and the Vendor Index fields provide a wide variety of data structures.

The data structures in the response message may be completely different and of a different length than the data structures in the request message. The message does not contain the length of data structure field. The length is managed by the transport.

Refer to Example Usage of GCP for an example of how to use GCP with the EDS message.

6.3 Register Access

6.3.1 Exchange Data Registers (EDR)

This message allows one or more registers in the GCP slave device to be written by the GCP master device starting at a specified base address and then have those registers read back in the response message. The message allows a write-only, a read-only, or a write-with-read-back.

Table 16 - Exchange Data Registers Request

Description	Size	Contents	
Message ID	1 byte	16	
Message Length	2 bytes	If No Write, then 9 If Write, the 9 + N	
Transaction ID	2 bytes	Unique value	
Mode	1 byte	bit 7: bit 6: bit 5: bit 4: bit 3: bit 2-0:	0 = Send normal response (default) 1 = Suppress normal response 0 = Write, 1 = No Write 0 = Read, 1 = No Read 0 = Linear Mode (default) 1 = FIFO Mode 0 = Big Endian (default) 1 = Little Endian 000 = Unspecified (default) 001 = 8-bit data width 010 = 16-bit data width 011 = 32-bit data width 100 = 64-bit data width 101, 110, 111: Reserved.
Starting Address	4 bytes	0x00000000 to 0xFFFFFFFF	
Byte Count	2 bytes	1 to 65,536	
Sent Data	N bytes	If Write, then data If No Write, then no data.	

Table 17 - Exchange Data Registers Normal Response

Description	Size	Contents
Message ID	1 byte	17
Message Length	2 bytes	If No Read, then 4 If Read, then 3 + M
Transaction ID	2 bytes	Same as request
Mode	1 byte	Same definition as mode field in request.
Return Code or Data	1 byte M bytes	If No Read, then return code = 0 If Read, then data.

Table 18 - Exchange Data Registers Error Response

Description	Size	Contents
Message ID	1 byte	145
Message Length	2 bytes	3
Transaction ID	2 bytes	Same as request
Return Code	1 byte	See Section 6.4

EDR is fundamentally a register write-with-read-back message, of which write-only and read-only are subsets of the main message. The default Mode field is all zeroes.

To write to multiple registers that are not contiguous in memory, use multiple EDR messages. Note that multiple messages may be grouped together into a single packet.

6.3.1.1 Response Options

The normal response message can be suppressed by asserting the Suppress Normal Response bit. This mode may be useful when a series of back-to-back writes are performed and there is a desire to reduce the workload on the remote device. In the event of an error at the GCP Slave, the GCP Slave will return an error response with the appropriate error code.

The Read bit takes precedence over the Suppress Normal Response bit.

6.3.1.2 Read and Write Modes

Read and write operations for EDR are shown in Table 19.

Table 19 - EDR Read and Write Modes

W = 0, R = 0	The data in the request message is written into a register as dictated by the address pointer. The same address point is then used to read back the data.This command allows the GCP Master to verify that the correct values were written. See the requirement below this table.
W = 0, R = 1	The data in the request message is written into a register. The response message has a return code but no data.
W = 1, R = 0	The request message contains an address but no data. The response message contains data from the register at the specified address location.
W = 1, R = 1	No reads or writes take place. The error codes are still valid and the GCP Slave returns a normal response. This operation is a null operation and may be used to test connectivity or address validity.

For an Exchange Data Registers request with Write mode = 0 and Read mode =0, the GCP Slave device SHOULD perform a true internal write followed by a true internal read rather than just echoing the data back.

Note that a read-modify-write operation would take two separate EDR commands. As an alternative, consider using the MWR message.

6.3.1.3 Address Mode

If linear mode is specified, the address field is incremented with consecutive writes and/or reads. If FIFO mode is specified, then the address field is held constant. The FIFO mode is for a FIFO input or output that is located at a fixed memory location.

6.3.1.4 Data Format

The GCP Master and GCP Slave may be radically different in their internal architectures. For example, the GCP Master may be a high-end processor system with a 32-bit data bus, while the GCP Slave might be a smaller embedded processor with an 8-bit or 16-bit data bus. The variation between systems includes data bus width as well as the number of bytes per address location.

Since GCP moves data from one system to the other over a serial transport, there is a natural conversion between the two systems. In the event that the data requires specific handling, such as the preservation of a 32-bit value, GCP allows the data to be identified as such.

The data handling options are shown in Table 20.

Table 20 - Data Handling within GCP Transport

Format	Description
Byte stream	The address pointer and a byte count can identify the data. The GCP Master and Slave inherently know what to do. This is the default mode for GCP.
N bit	Data provided is from memory or a register set that is N-bit sized, where N = 8, 16, 32, or 64.
Endian	The data can be marked as being big-endian or little-endian. The default mode is big-endian.

These values are always valid for the data being transported in the request or response messages and may be different in the two messages.

6.3.2 Mask Write Register (MWR)

This message is used by the GCP Master to modify the contents of a specified register in the GCP Slave by using a combination of an AND mask, an OR mask, and the register's current contents. The message can be used to set or clear individual bits in the register. As a result, this message can be used to more efficiently replace a read-modify-write operation.

Table 21 - Mask Write Register Request

Description	Length	Contents	
Message ID	1 byte	18	
Message Length	2 bytes	23	
Transaction ID	2 bytes	Unique value	
Mode	1 bytes	bit 7: bit 6: bit 5: bit 4-3: bit 2-0:	0 = Send normal response 1 = Suppress normal response 0 = Write 0 = Read, 1 = No Read Reserved. Set to 0. 001 = 8-bit data width 010 = 16-bit data width 011 = 32 bit data width 100 = 64-bit data width 000, 101, 110, 111: Reserved.
Reference Address	4 bytes	0x00000000 to 0xFFFFFFFF	
AND_Mask	8 bytes	8-bit to 64-bit field	
OR_Mask	8 bytes	8-bit to 64-bit field	

Table 22 - Mask Write Register Normal Response

Description	Length	Contents
Message ID	1 byte	19
Message Length	2 bytes	If No Read, then 4 If Read, then 7
Transaction ID	2 bytes	Same as request
Mode	1 bytes	Same definition as mode field in request.
Return Code or Data	1 byte 4 bytes	If No Read, then return code = 0 If Read, then data.

Table 23 - Mask Write Register Error Response

Description	Length	Contents
Message ID	1 byte	147

Description	Length	Contents
Message Length	2 bytes	3
Transaction ID	2 bytes	Same as request
Return Code	1 byte	See Section 6.4

6.3.2.1 Read and Write Modes

Read and write operations for MWR are shown in Table 24.

Table 24 - MWR Read and Write Modes

W = 0, R = 0	The data in the request message is written into a register as dictated by the address pointer. The same address point is then used to read back the data. This command allows the GCP master to verify that the correct values were written. See the requirement below this table.
W = 0, R = 1	The data in the request message is written into a register. The response message has a return code but no data.

For a Mask Write Register request with Write mode = 0 and Read mode =0, the GCP slave device SHOULD perform a true internal write followed by a true internal read rather than just echoing the data back.

6.3.2.2 Response Options

The normal response message can be suppressed by asserting the Suppress Normal Response bit. This mode may be useful when a series of back-to-back writes are performed and there is a need to reduce the workload on the remote device. In the event of an error at the GCP Slave, the GCP Slave will return an error response with the appropriate error code.

The Read bit takes precedence over the No Response bit.

6.3.2.3 Data Format

The request message specifies the register to be written, the data to be used as the AND mask, and the data to be used as the OR mask. The number of bytes written and read, and hence the size of the mask, is specified in the Mode field. When less than 64 bits are used, it is the least order bytes in the Mask fields that are used. The unused bits in the AND_MASK and the OR_MASK field SHOULD be set to 0 by the GCP Master and ignored by the GCP Slave.

The function's algorithm is:

Result = (Contents AND AND_Mask) OR (OR_Mask AND (NOT AND_Mask))

If the OR_Mask value is zero, the result is simply the logical ANDing of the current contents and AND_Mask. If the AND_Mask value is zero, the result is equal to the OR_Mask value.

This equation yields the following truth table:

Contents	AND_Mask	OR_Mask	Result
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

For example (only 8 bits are shown):

Example	Hex	Binary
Current Contents	12	0001 0010
AND_Mask	F2	1111 0010
OR_Mask	25	0010 0101
NOT AND_Mask	0D	0000 1101
Result	17	0001 0111

6.4 GCP Return Codes

The GCP response message contains a return code that indicates either success or a specific type of failure. These return codes are shown in Table 25.

Table 25 - GCP Message Return Codes

Code	Name	Notify	GDM	EDS	EDR	WMR
0	MESSAGE SUCCESSFUL	Y	Y	Y	Y	Y
1	UNSUPPORTED MESSAGE	n/a	Y	Y	Y	Y
2	ILLEGAL MESSAGE LENGTH	Y	Y	Y	Y	Y
3	ILLEGAL TRANSACTION ID	Y	Y	Y	Y	Y
4	ILLEGAL MODE	Y	Y	Y	Y	Y
5	ILLEGAL PORT	n/a	Y	Y	n/a	n/a
6	ILLEGAL CHANNEL	n/a	Y	Y	n/a	n/a
7	ILLEGAL COMMAND	n/a	Y	n/a	n/a	n/a
8	ILLEGAL VENDOR ID	n/a	n/a	Y	n/a	n/a
9	ILLEGAL VENDOR INDEX	n/a	n/a	Y	n/a	n/a
10	ILLEGAL ADDRESS	n/a	n/a	n/a	Y	Y
11	ILLEGAL DATA VALUE	n/a	n/a	Y	Y	n/a
12	MESSAGE FAILURE	Y	Y	Y	Y	Y
13-127	Reserved	-	-	-	-	-
128-254	User Defined Codes	-	-	-	-	-
255	SLAVE DEVICE FAILURE	n/a	Y	Y	Y	Y

Error codes are assigned in ascending numerical order. Thus, if there are multiple errors, the lowest numerical value error code should be reported.

The MESSAGE FAILURE code is intended to cover a failure that is not covered by the other codes. Use of this error code should be the exception rather than the rule.

7 GCP TRANSPORT

GCP is a messaging protocol and should be used with a reliable transport protocol. The primary choice of a transport protocol for GCP is TCP. TCP uses the IP protocol as a network protocol.

Under special circumstances, where reliability is not required, it is possible to use UDP. GCP messaging can also be used with other protocols such as L2TPv3 to offload the L2TPv3 signaling functionality.

7.1 TCP

TCP, the Transmission Control Protocol, is a reliable transport protocol, and is the recommended transport protocol for GCP.

7.1.1 TCP Advantages

TCP provides GCP the following transport services:

- A reliable transport. TCP does this through packet numbering, acknowledgements, and retransmissions.
- Segmentation. TCP learns the MTU of the media and segment and reassembles messages across IP packets as needed.
- Windowing. TCP is capable of sending out a series of request messages without having to wait for each response. TCP will burst out a number of messages, measured as a total number of bytes.

7.1.2 TCP Parameters

The IANA has assigned a registered port to GCP in its Service Name and Transport Protocol Port Number Registry [IANA]:

- Service Name gcp-rphy
- Port Number 8190
- Transport Protocol tcp
- Description Generic control plane for RPHY

7.1.3 TCP Payload Encapsulation

The TCP payload consists of a fixed GCP header followed by one or more GCP messages. The GCP packet format, using Ethernet as an example of a Layer 2 media, is shown in Figure 3.

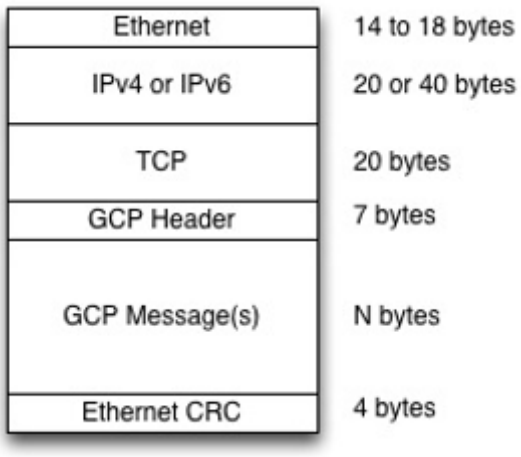

Figure 3 - GCP Packet Format

The GCP header is described is shown in Figure 4.

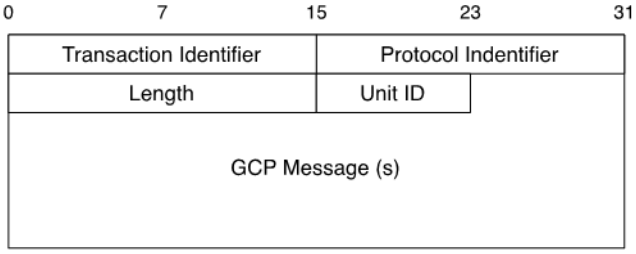

Figure 4 - GCP Header for TCP

The GCP Header for TCP is explained in Table 26.

Table 26 - GCP Header for TCP

Field	Length	Description
Transaction Identifier	2 bytes	Unique transaction ID. A value of 0 means to ignore this field.
Protocol Identifier	2 bytes	1 = GCP Protocol Version 1.
Length	2 bytes	Length of Unit Identifier Field plus Message Field
Unit Identifier	1 byte	Unit addressing with a device. Default is 0.
Message Field	N bytes	One or more GCP messages.

If the per-message transaction IDs are used by the GCP sender, then the per-packet transaction ID can be set to '0' by the GCP sender and ignored by the GCP receiver. If the per-packet transaction ID is used, then the GCP sender can choose an appropriate value and set the local message transaction IDs to zero. The sender may choose to use both the message transaction IDs and the packet Transaction IDs, or not use either of them.

The Unit Identifier can be used for further addressing within the device when the device and all of its units are serviced by a common GCP transport. For example, if the device contains multiple identical ASICs, then each ASIC could be assigned a unit identifier, and the same set of commands could be applied to each ASIC. Note that the global port and channel numbers only pertain within a unit identifier. The exact usage of the Unit Identifier field is application-specific and is not specified by GCP.

7.2 UDP

UDP, the User Datagram Protocol, does not guarantee the delivery of packets and should not be used for GCP unless:

- The system application does not care if packets are dropped, or
- The system has other methods for determining reliability.

For example, a system might use UDP to do a bulk write of messages to a number of GCP Slaves and then at the application level, read back every single parameter written to verify that the parameters "took." This technique might be used if the GCP Slave is not trusted. As such, the redundancy built into the application layer negates the need for a reliable transport. UDP is also applicable if the transport is on an internal system bus and that bus has its own method for ensuring delivery of bytes.

NOTE: The UDP port and encapsulation are identical to the TCP port and encapsulation.

7.3 L2TPv3

L2TPv3, the Layer Two Tunneling Protocol version 3, is a protocol that is used for tunneling Layer 2 services over a Layer 3 network (see [RFC 3931]). An L2TPv3 session consists of an overall tunnel with one or more sessions and a

common set of signaling messages. The signaling messages use fields called AVPs (Attribute Value Pairs). An AVP is a small data structure and is similar in concept to a TLV.

GCP messages can be tunneled over the L2TPv3 control plane by using a special GCP AVP. GCP messages are packaged into the GCP AVP and sent within an L2TPv3 control message. One or more GCP messages (TLVs) can be put into the GCP AVP.

The GCP AVP is shown in Figure 5.

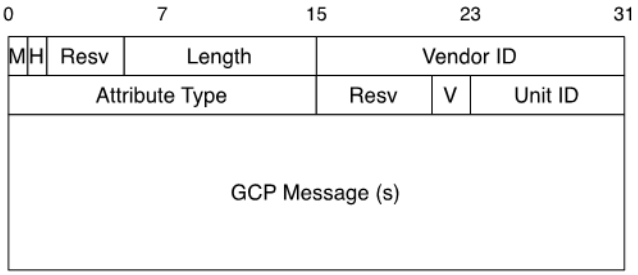

Figure 5 - GCP AVP for L2TPv3

The fields in the GCP AVP are described in Table 27:

Table 27 - GCP AVP Fields

Variable	Length	Description
M	1 bit	Mandatory. Per L2TPv3, this bit is asserted (set to 1) if the sender wants the L2TPv3 session torn down if this AVP is rejected. Typically, this bit is set to 0.
H	1 bit	Hidden. Per L2TPv3, this bit is asserted if the contents of the AVP are encrypted.
Resv	4 bits	Reserved. Set to all zeroes by sender. Ignored by receiver.
Length	10 bits	The length, in bytes, of the entire AVP including all headers and payload.
Vendor ID	16 bits	The Vendor ID field indicates what organization has defined this AVP. The value of this field is TBD. NOTE: Final number assignment will be by IANA or by Vendor.
Attribute Type	16 bits	This is the attribute number that identifies this as the GCP AVP. The value of this field is TBD. NOTE: Final number assignment will be by IANA or by Vendor.
Resv	6 bits	Reserved. Set to all zeroes by sender. Ignored by receiver. This field is the first field of the AVP payload.
V	2 bits	Version of GCP protocol.
Unit ID	8 bits	Unit addressing with a device. Default is 0.

There is no explicit transaction ID within the GCP AVP because L2TPv3 permits a serial number, which is similar to a transaction ID, to be added as a separate AVP. The GCP messages within the GCP payload have their own transaction IDs.

Appendix I Example Usage of GCP (Informative)

The [MULPI] specification describes in detail how an upstream burst demodulator works and performs. As part of its signaling protocol, the CMTS has to signal to the CM all the configuration parameters for the burst demodulator. These parameters are contained in a DOCSIS MAC management message called the Upstream Channel Descriptor (UCD).

If a similar burst demodulator was to be used in a different system but required the same configuration parameters, it might be more convenient to reference the work done in the DOCSIS specification rather than recreating it in a new specification. In addition to leveraging existing work, it also allows new work in an environment (DOCSIS in this case) to be immediately used in another environment without having to update a separate specification.

As an analogy, GCP permits the definition of a higher-level data structure, which then can point to a lower-level data structure that is contained elsewhere.

The GCP pointer is achieved by defining the following variables:

- Vendor ID = 4491 (CableLabs)
- Structure ID = 35 (DOCSIS UCD)

The [MULPI] specification is maintained by CableLabs, so the GCP vendor ID can be set to CableLabs. The UCD message in [MULPI] has a message ID of 35, so it is convenient to use the same number.

This usage of GCP can be referred to as GCP(UCD). A separate specification could be written that would describe GCP(UCD), what data structures are used, and how they are packed into the GCP message payload.

The device shall not have this requirement in the specification.

Appendix II Message Length Considerations (Informative)

When GCP runs over TCP, and since TCP provides segmentation, then in theory messages or groups of messages can be any length. In practice, there may be performance limits that are specific to the system that discourage large messages or groups of messages. Other protocols like UDP that do not have segmentation may need a maximum length. This appendix presents some basic message length calculations.

The first consideration to note is that TCP runs a window. If that window is for example 15,000 bytes, then any register access for larger sizes will have some delay. There is also the speed of the connecting media to consider when doing large register accesses.

If the desire is to keep message length to within an IP packet boundary, such as would be the case when UDP is the transport, then the maximum size of the GCP message depends upon the MTU (Maximum Transmission Unit) of the underlying transport. For example, the typical MTU of Ethernet allows a protocol payload of 1500 bytes and a frame size of 1518 bytes (non-VLAN) or 1522 bytes (on VLAN).

For Ethernet, the resulting maximum GCP message length would be:

GCP Max Message Length	= Ethernet payload (1500 bytes) ..- TCP header (20 bytes) ..- IPv4 header (20 bytes) ..- GCP header (7 bytes) = 1453 bytes

Or,

GCP Max Message Length	= Ethernet payload (1500 bytes) ..- TCP header (20 bytes) ..- IPv4 header (40 bytes) ..- GCP header (7 bytes) = 1433 bytes

Note that the MTU of a media is often negotiated with MTU discovery protocols and may be more or less than the default value.

Appendix III Acknowledgements

On behalf of the cable industry and our member companies, CableLabs would like to thank the following individuals for their contributions to the development of this specification:

Contributor	Company Affiliation
John T. Chapman	Cisco

On behalf of the cable industry and our member companies, CableLabs would like to thank the following individuals for their contributions to the development of the technology and participation in the Remote PHY Working Group.

Contributor	Company Affiliation
Bill Powell	Alcatel-Lucent
Brian Kurtz	Altera
Carlton Lane, Linda Mazaheri	Analog
Tom Ferreira, Steve Foley, Anand Goenka, Jeff Howe, Hari Nair	Arris
Andrew Chagnon, Victor Hou, Niki Pantelias, David Pullen	Broadcom
Stuart Hoggan, Volker Leisse, Jon Schnoor, Karthik Sundaresan, Nikhil Tayal, Jun Tian	CableLabs
Andrew Sundelin	CableLabs Consultant
Naor Goldman	Capacicom
Dave Fox, Maike Geng	Casa Systems
David Claussen	Charter
Nobo Akiya, Alon Bernstein, Brian Bresnahan, John T. Chapman, Hang Jin, Tong Liu, Carlos Pignataro, Sangeeta Ramakrishnan, John Ritchie, Pawel Sowinski, Don Strausberger, Yi Tang, Bill Wall, Gerry White	Cisco
Philippe Perron	Cogeco
John Bevilacqua, Nagesh Nandiraju, Saifur Rahman, Jorge Salinger, Joe Solomon, Douglas Will	Comcast
Jeff Ford, Al Garrett	Complex IQ
Ony Anglade , Mike Cooper,	Cox Communications
Samir Parikh	Gainspeed Networks
João Campos, Even Kristoffersen	Get
Adi Bonen, Mike Patrick	Harmonic
Jim Chen, Hesham ElBakoury, Karl Moerder, Jack Moran, Guangsheng Wu	Huawei
Phil Oakley	LGI
Stan Bochenek, Ajay Kuckreja	Maxim Integrated
Len Dauphinee , David Huang, Louis Park, Sridhar Ramesh, Patrick Tierney, Scott Walley	MaxLinear
Rei Brockett	Pace/Aurora
Nasir Ansari, George Hart	Rogers
Kevin Kwasny	Shaw
Lee Johnson	ST Micro
Paul Brooks, Kirk Erichsen	Time Warner Cable
Colin Howlett Douglas Johnson,	Vecima
Alex Luccisano	Xilinx

Additionally, CableLabs would like to thank the DCA MSO team for their continued support in driving the specification development and the decision-making process.

Karthik Sundaresan and Jon Schnoor, CableLabs

Appendix IV Revision History

Engineering Change for CM-SP-GCP-I02-160512

ECN	Date Accepted	Summary	Author
GCP-N-16.1486-1	04/21/2016	GCP Port assignment	Sundaresan

Engineering Change for CM-SP-GCP-I03-170524

ECN	Date Accepted	Summary	Author
GCP-N-17.1697-1	03/09/2017	Editorial EC to remove Requirements from table	Barringer