Video Streaming and Device Control Over Ethernet Standard

version 2.0

(includes errata up to March 13, 2013)

900 Victors Way, Suite 140 • Ann Arbor, Michigan 48108 USA • www.visiononline.org

GigE Vision Licensing and Logo Usage

GigE Vision is a widely adopted standard and is used on hundreds of products on the market today.

The standard was designed so that users of the technology can quickly and easily identify GigE Vision compliant products that will interoperate and "plug and play" with each other. All commercial products developed using the GigE vision standard must license the standard and qualify for the right to use the name and logo. To qualify, each product must have the proper paperwork submitted to the AIA and must pass GigE Vision compliance testing. More information on licensing GigE Vision can be found at

www.visiononline.org/standards

The GigE Vision logo may be used only in conjunction with licensed products which have passed GigE Vision compliance testing.

Special Note

This version of the GigE Vision standard text includes all errata that have been identified in the main text since the release 2.0 of the specification. These errata are highlighted using a light green background color. A comprehensive list of the changes can be found in the *Document History* section at the end of the document.

Table of Content

List of Figures

List of Tables

1 Introduction

1.1 Purpose

GigE Vision is a communication interface for vision applications based on the ubiquitous Ethernet technology. It allows for easy interfacing between the GigE Vision device and a network card using standard CAT-5e/6 cable or any other physical medium supported by Ethernet. GigE Vision supports any speed grade of Ethernet.

The aim of this specification is to define the protocols used by GigE Vision compliant products to communicate with each other. The specification builds upon the Ethernet technology. As such, it does not cover the physical description of the transport media. Nor is any specific implementation of specialized high-performance Internet Protocol (IP) stack or network driver part of the specification.

GigE Vision systems cover a wide spectrum of different network topologies. The simplest one is a point-to-point connection between a personal computer (PC) and a GigE video streaming device using a single cable; a more complex one is a video distribution system comprised of hardware and software video sources, sinks and processing units on an IP network using routers. It is crucial that GigE Vision products are fully compatible with other IP devices on the network. Therefore, it is mandatory to follow the guidelines provided by the Request for Comments (RFC) of the Internet Engineering Task Force (IETF). But at the same time, the number of mandatory requirements should be limited to allow for economical realization of GigE Vision devices.

Even though the name of this specification specifically refers to Gigabit Ethernet, GigE Vision can be implemented for any Ethernet speed grade. Any physical medium specified by IEEE 802.3 can be used to implement GigE Vision.

1.2 Technical Committee

The GigE Vision Standard Committee was formed in June 2003 to define an open transport platform based on gigabit Ethernet for high-performance vision applications. The committee is sponsored by the AIA: http://www.visiononline.org/.

1.2.1 Version 2.0

Work on version 2.0 of the standard was initiated in October 2009. The following members have provided technical content introduced by this revision of the standard.

Eric Carey	Teledyne DALSA inc.
Eric Gross	National Instruments
Stéphane Maurice	Matrox Electronic Systems
Thies Möller	Basler Vision Technologies
Vincent Rowley	Pleora Technologies Inc.
Rupert Stelz	STEMMER IMAGING

The following individuals have participated in the technical meetings leading to version 2.0.

Jeff Fryman, Director of Standards

Eric Carey, Chair

Vincent Rowley, Vice-Chair

Osamu Aimono	Mark Jones	Thomas Olschläger
Chris Barrett	Kohei Kageshita	Ivonne Puhlmann
Stefan Battmer	Seiji Kajiwara	Gordon Rice
Jan Becvar	Kenji Kakinoki	Geoff Roddick
Stéphane Bouchard	Eckhard Kantz	Manuel Romero
Eric Bourbonnais	Matthias Karl	Björn Rudde
Karsten Ingeman Christensen	Tim Kennedy	Matthias Schaffland
Liviu Dan	Sebastian Kinzler	Markus Schatzl
Friedrich Dierks	Andreas Koch	Toshiya Shiba
Sascha Dorenbeck	Kazunari Kudo	Arnold Sodder
Ralf Ebeling	Stéphane Laplante	Malcom Steenburgh
Holger Eddelbuettel	John Le	Rupert Stelz
Werner Feith	Masahide Matsubara	Akiyoshi Sugawara
Ryan Friedman	Stéphane Maurice	Dave Tarkowski
Martin Garon	Michael Miethig	Glen Varsava

Alain Gauthier	Thies Möller	Shinichi Wakabayashi
Bob Grochmal	Ronald Mueller	Xuemei Wang
Eric Gross	Marcel Naggatz	Manfred Wuetschuer
Alexander Happe	Damian Nesbitt	Peifang Zhou
Thomas Hopfner	Hiroyuki Nishikawa	Christoph Zierl

The following technical member companies voted on this revision of the standard.

Basler Vision Technologies	National Instruments
Baumer Optronic	Pleora Technologies
JAI A/S	Point Grey Research
Kappa Optronics GmbH	Sensor To Image
Leutron Vision AG	STEMMER IMAGING
Matrox Electronic Systems	SVS-Vistek
MathWorks, Inc.	Teledyne DALSA
MVTec Software	

The following individuals provided independent review of some parts of the standard.

Niranjan Avadhanam	Ambarish Natu	Sassan Pejhan
Michael Isnardi	Thierry Pauwels	Shevach Riabtsev
Arkady Kopansky		

1.3 Definitions and Acronyms

1.3.1 Definitions

Aggregator	In IEEE 802.1AX, the entity performing the operations required to have multiple physical links function as an aggregated link.
Application	GigE Vision control application software running on a host. Typically a software application running on a PC but can also be of another nature such as microcode running on a field programmable gate array (FPGA).
Control protocol	GigE Vision control protocol (GVCP) defining commands supported by GigE Vision compliant devices.
Conversation	As defined by IEEE 802.1AX, a set of packets transmitted from one device to another, where all of the packets form an ordered sequence, and where the communicating devices require the ordering to be maintained among the set of packets exchanged.

Device	GigE Vision compliant controllable device. Typically a camera but can also be of another nature such as a software video server running on a PC.
Event Generator	Entity generating event messages according the GigE Vision specification.
Event Receiver	Entity receiving and capable of interpreting event messages according the GigE Vision specification.
Gratuitous ARP	An ARP request or reply sent by the device where the source and destination IP addresses are both set to the IP of the device issuing the packet and the destination MAC is the broadcast address. This can be used to check if another device is using the same IP address.
GVSP Receiver	Entity receiving and capable of de-encapsulating a stream of data according to the GigE Vision Streaming Protocol.
GVSP Transmitter	Entity producing a stream of data according to the GigE Vision Streaming Protocol.
Link	The physical connection that transports packets across a source and a destination.
Load Balancing	The practice of allocating traffic across multiple links to evenly distribute load and maximize overall throughput.
Management Entity	Application responsible for the configuration and control of devices deployed in a system.
Multihomed	A host is said to be multihomed if it has multiple IP addresses.
Network Interface	A subsystem providing the means for a device to attach to a communication link.
Partner	The device at the other end of a point-to-point link. This could be an end station (such as a network interface card) or an Ethernet switch port among others.
Persistent IP	A persistent IP address is hard-coded in non-volatile memory of the device. It is re-used by the device on power-cycle when Persistent IP is enabled.
Primary application	Application having exclusive or control access (read/write) to the device. This includes control access with switchover enabled.
Primary application Switchover request	The condition under which another application is trying to take control over a device that is under the control of a primary application that was granted control access with switchover enabled.
Secondary application	Application having monitoring access (read-only) to the device.
Server	Server that provides IP server functionality (ex: DHCP) used to assign IP addresses. The server can optionally be included in the application, but most of the time it is separated.
Standard GVCP port	UDP port used on a device to receive GVCP commands. The IETF has attributed port number 3956 as the standard GVCP port. For multi-service device, this is the port attached to the first service.
Static IP	A static IP address is set by the application into the device volatile memory. It is lost on power-cycle. Static IP is mainly used by an application to “force” an IP address into a device (for instance, when an invalid Persistent IP is memorized by the device).

1.3.2 Requirements Terminology

This specification uses the following convention to list requirements.

Table 1-1: Requirements Terminology

Term	Description	Representation
Absolute Requirement	Feature that MUST be supported by the product. It is mandatory to support the feature to ensure interoperability.	[R-<sn><suffix>]
Conditional Requirement	Feature that MUST be supported IF another feature is present. It is mandatory to support the feature when another feature is supported.	[CR-<sn><suffix>]
Absolute Objective	Feature that SHOULD be supported by the product. It is recommended, but not essential.	[O-<sn><suffix>]
Conditional Objective	Feature that SHOULD be supported IF another feature is present. It is recommended, but not essential.	[CO-<sn><suffix>]

Each requirement is represented by a unique number in brackets. Each number is composed of up to 3 elements:

1. Requirement Type: Absolute **R**equirement, **C**onditional **R**equirement, Absolute **O**bjective, **C**onditional **O**bjective
2. Sequence number (sn): Unique number identifying the requirement or objective. The sequence numbers are attributed sequentially as new requirements and objectives are added to the specification.
3. Suffix: Identifies if the requirement or objective is applicable to application software, devices, GVSP transmitters, GVSP receivers or some combination of these according to the terminology presented in Table 1-2.

Table 1-2: Suffix Terminology

Suffix	Meaning	Description
ca	**c**ontrol **a**pplication	Represents a requirement or objective exclusive to application software.
cd	**c**ontrol **d**evice	Represents a requirement or objective exclusive to devices.
c	**c**ontrol	Represents a requirement or objective applicable to both applications and devices.
st	**s**tream **t**ransmitter	Represents a requirement or objective exclusive to GVSP transmitters.
sr	**s**tream **r**eceiver	Represents a requirement or objective exclusive to GVSP receivers.
s	**s**tream	Represents a requirement or objective applicable to GVSP transmitters and receivers.
No suffix	-	Represents a requirement or objective applicable to applications, devices, GVSP transmitters and receivers.

For instance, [R-61cd] is requirement number 61 that only applies to GigE Vision devices.

A cross-reference table is provided in Appendix 1 to link the requirements and objectives with the ones from GigE Vision version 1.x.

Note: GigE Vision 1.x used a different requirement numbering scheme where the section number was part of the requirement number. This created issues when sections were added and led to the introduction of this new numbering style.

To facilitate the correspondence between the requirement numbers of this older style (pre-GigE Vision 2.0) and the ones introduce by GigE Vision 2.0, the older style requirement number is appended at the end of the text of the requirement when applicable.

1.3.3 Acronyms

AOI	Area of Interest
ARB	Arbitrary (as defined in IEEE 1588)
BOOTP	Bootstrap Protocol
CCP	Control Channel Privilege (a bootstrap register)
CFA	Color Filter Array
CID	Chunk Identifier
DHCP	Dynamic Host Configuration Protocol
dLAG	Dynamic Link Aggregation Group
DNS	Domain Name System
DNS-SD	Domain Name System - Service Discovery
EOC	End Of Codestream

EOI	End Of Image
FMO	Flexible Macroblock Ordering
GEV	GigE Vision
GIF	Graphic Interchange Format
IANA	Internet Assigned Numbers Authority
ID	Identifier
IDR	Instantaneous Decoding Refresh
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IP	Internet Protocol
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
ISO	International Organization for Standardization
ITU	International Telecommunication Union
GigE	Gigabit Ethernet
GMT	Greenwich Mean Time
GPS	Global Positioning System
GVCP	GigE Vision Control Protocol
GVSP	GigE Vision Streaming Protocol
JFIF	JPEG File Interchange Format
JPEG	Joint Photographic Experts Group
LAG	Link Aggregation Group
LFSR	Linear Feedback Shift Register
LLA	Link-Local Address
lsb	Least Significant Bit
LSB	Least Significant Byte
MAC	Media Access Control
mDNS	Multicast Domain Name System
ML	Multiple Links
MPEG	Moving Picture Experts Group
msb	Most Significant Bit
MSB	Most Significant Byte
MTU	Maximum Transmission Unit
NAL	Network Abstraction Layer
NAT	Network Address Translation

NIC	Network Interface Card
OSI	Open Systems Interconnection
OUI	Organizationally Unique Identifier
PC	Personal Computer
PFNC	Pixel Format Naming Convention
PTP	Precision Time Protocol
RFC	Request For Comments
ROI	Region of Interest
RTP	Real-time Transport Protocol
SDP	Session Description Protocol
SL	Single Link
sLAG	Static Link Aggregation Group
SOC	Start Of Codestream
SOD	Start Of Data
SOI	Start Of Image
SOT	Start Of Tile
SPI	Stateful Packet Inspection
TAI	International Atomic Time
TCP	Transmission Control Protocol
UCS	Universal Character Set
UDP	User Datagram Protocol
UTC	Coordinated Universal Time
UTF	UCS Transformation Format
UTP	Unshielded Twisted Pair
VCL	Video Coding Layer

1.4 Reference Documents

IEEE Standards	
IEEE 1588-2008	IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems. Also known as Precision Time Protocol (PTP).
IEEE 802-2001	IEEE Standard for Local and Metropolitan Area Networks: Overview and Architecture
IEEE 802.1AX-2008	IEEE Standard for Local and Metropolitan Area Networks – Link Aggregation
IEEE 802.3-2002	IEEE Standard for Information technology--Telecommunications and information exchange between systems--Local and metropolitan area networks--Specific requirements--Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications

Internet Standard	
STD3 (RFC1122 + RFC1123)	Requirements for Internet Hosts - Communication Layers + Requirements for Internet Hosts - Application and Support
STD5 (RFC791 + RFC792 + RFC919 + RFC922 + RFC950 + RFC1112)	INTERNET PROTOCOL + Internet Control Message Protocol + Broadcasting Internet Datagrams + Broadcasting Internet datagrams in the presence of subnets + Internet Standard Subnetting Procedure + Host extensions for IP multicasting
STD6 (RFC768)	User Datagram Protocol
STD13 (RFC1034 + RFC1035)	Domain Concepts and Facilities Domain Implementation and Specification
STD33 (RFC1350)	THE TFTP PROTOCOL (REVISION 2)
STD37 (RFC826)	Ethernet Address Resolution Protocol: Or Converting Network Protocol Addresses To 48.bit Ethernet Address For Transmission On Ethernet Hardware
STD41 (RFC894)	Standard For The Transmission Of IP Datagrams Over Ethernet Networks
STD43 (RFC1042)	Standard For The Transmission Of IP Datagrams Over IEEE 802 Networks
Internet RFC	
RFC951	Bootstrap Protocol
RFC1035	Domain Names – Implementation and Specification
RFC1534	Interoperation Between DHCP and BOOTP
RFC1542	Clarifications and Extensions for the Bootstrap Protocol
RFC1951	DEFLATE Compressed Data Format Specification version 1.3
RFC2131	Dynamic Host Configuration Protocol
RFC2132	DHCP Options and BOOTP Vendor Extensions
RFC2435	RTP Payload Format for JPEG-compressed Video
RFC3171	IANA Guidelines for IPv4 Multicast Address Assignments
RFC3376	Internet Group Management Protocol, Version 3
RFC3629	UTF-8, a transformation format of ISO 10646
RFC3927	Dynamic Configuration of IPv4 Link-Local Addresses
RFC3984	RTP Payload Format for H.264 Video
RFC3986	Uniform Resource Identifier (URI): Generic Syntax
RFC5227	IPv4 Address Conflict Detection
RFC5371	RTP Payload format for JPEG 2000 Video Streams
Internet Draft	
Multicast DNS	Multicast DNS Internet Draft, S. Cheshire, M. Krochmal
DNS-Based Service Discovery	DNS-Based Service Discovery Internet Draft, S. Cheshire, M. Krochmal

Others	
AIA Pixel Format Naming Convention	AIA Pixel Format Naming Convention version 1.0.
GenICam™	Generic Interface for Cameras, EMVA, version 2.2.0. GenICam™ is a trademark of the European Machine Vision Association.
ITU-T Rec. T.81	JPEG – Digital compression and coding of continuous-tone still images
ITU-T Rec. T.800	JPEG 2000 image coding system
ITU-T Rec. H.264	Advanced video coding for generic audiovisual services

1.5 Document Typographic Convention

This standard document uses the following typographic conventions:

Bootstrap registers

Bootstrap registers are highlighted using Arial font.

Ex: Link Speed

Field names in packets and bootstrap registers

The names of the fields in a packet are expressed using lowercase *italic* letters, except for acronyms. When a field name contains an acronym or a reserved word, then the acronym can also use uppercase *italic* letters. When multiple words are used, they are separated with an underscore ('_').

Ex: *packet_size*

Feature names

Feature names are expressed by concatenating all the words forming the feature name and using uppercase for the first letter of each word.

Ex: AcquisitionStart

GVCP Commands

All GVCP command names are expressed in uppercase with no space. Underscore ('_') are used to separate the CMD (command) or ACK (acknowledge) suffix.

Ex: READMEM_CMD

GigE Vision Status Codes

All GigE Vision status codes are expressed using uppercase with underscore ('_') to separate multiple words. They start with "GEV_STATUS".

Ex: GEV_STATUS_SUCCESS

GigE Vision Events

All GigE Vision events are expressed using uppercase with underscore (‘_’) to separate multiple words. They start with “GEV_EVENT”.

Ex: GEV_EVENT_LINK_SPEED_CHANGE

1.6 Liability Disclaimer

This specification is provided "as is" and without any warranty of any kind, expressed or implied. Without limitation, there is no warranty of non-infringement, no warranty of merchantability, and no warranty of fitness for a particular purpose. All warranties are expressly disclaimed.

User assumes the full risk of using this specification. In no event shall the AIA, members of the technical committee, or their companies, be liable for any actual, direct, indirect, punitive, or consequential damages arising from such use, even if advised of the possibility of such damages.

1.7 System Overview

GigE Vision systems cover a wide spectrum of different network topologies. The simplest one is a point-to-point connection between a PC and a GigE video streaming device using a crossover cable or even over an Ethernet network while a more complex one is a video distribution system comprised of hardware and software video sources, sinks, processing units and peripherals on an IP network.

Figure 1-1 demonstrates the traditional point-to-point camera to host control and receiving application use case while Figure 1-2 demonstrates a more complex use case.

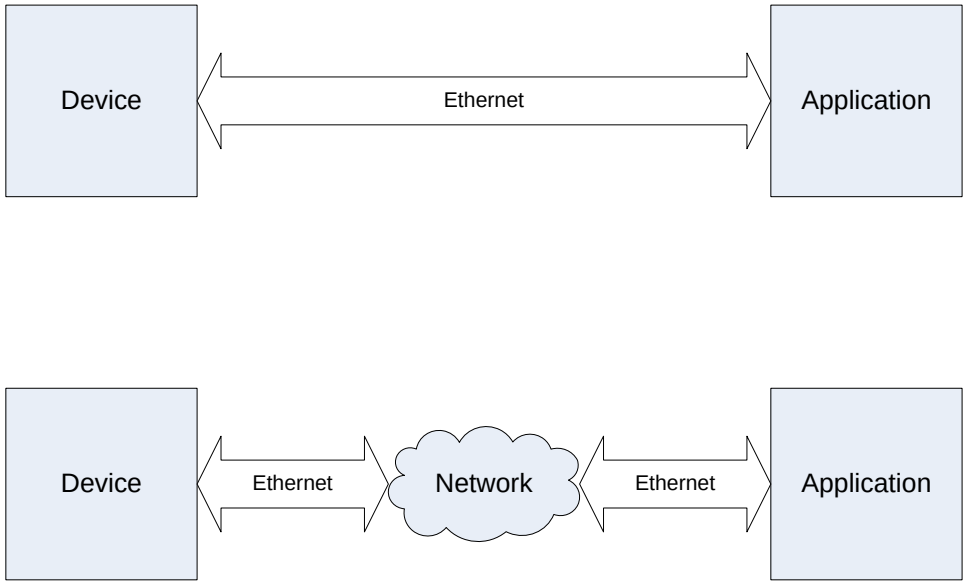


Figure 1-1: Traditional Streaming Device to Control and Receiving Application

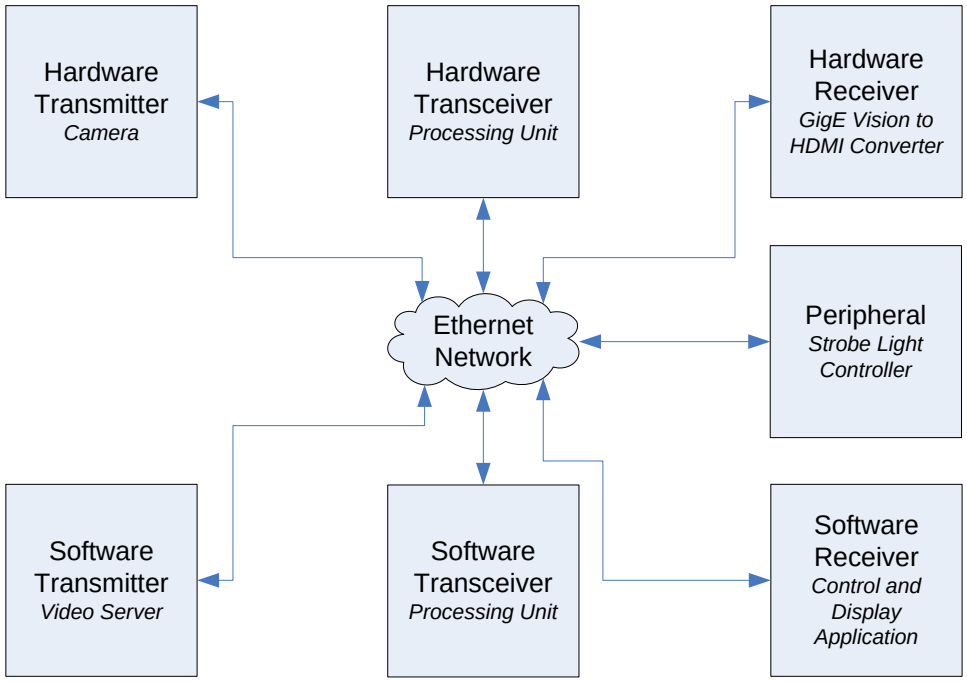


Figure 1-2: Advanced Video over Ethernet Distribution System

Although, the GigE Vision protocol could be leveraged effectively and successfully in more complex systems, versions prior to release 1.2 of this specification limited the class of products that could claim GigE Vision compliance. This is because the requirements terminology being used in these versions focused more or less on the typical camera use case where a device would be a streaming device transmitting video data to a control and receiving application.

Version 1.2 of this specification removed this limitation by simply decoupling control and streaming in the requirements terminology. In other words, a device can be transmitting and/or receiving data while the opposite holds true for an application.

Therefore, version 1.2 defined new classes of products and generalized the concepts. The main concepts are:

1. A product must be an application, a device or both.
2. A product doesn't need to have a stream channel.

In addition to this, the following device classes are defined.

1. ***Transmitter***: A transmitter is a device capable of streaming data. It includes one or more GVSP transmitters. For instance, this can be a camera.
2. ***Receiver***: A receiver is a device capable of receiving data. It includes one or more GVSP receivers. For instance, this can be a GigE Vision to HDMI converter.
3. ***Transceiver***: A transceiver is a device capable of receiving and transmitting data. It includes one or more GVSP receivers and one or more GVSP transmitters. For instance, this can be a GigE Vision image processor.

4. ***Peripheral***: A peripheral is a device that cannot transmit or receive data. However, it can execute tasks controlled using the GigE Vision control protocol. For instance, this can be a GigE Vision strobe light controller.

This specification also dictates that if a GigE Vision product is to be controlled over Ethernet, it must be possible to do so using the GigE Vision control protocol. This is in order to insure interoperability amongst GigE Vision products.

It should be noted that a product can act as the control application of more than one device. However, the parameters of a product can only be controlled by a single control application at a given point in time. Please refer to section 10.1 for more details.

Version 1.2 of the specification also maintained the "streaming devices" to "control and receiving applications" backward compliance (see second paragraph of this section as well as Figure 1-1). However, given the fact that new type of products were introduced, one could not expect forward compatibility if using a legacy product with a newly defined one. For instance, an application created to handle streaming devices in the past may have some difficulties to cope with a newly defined peripheral device if the application tries to instantiate a stream channel on the device. Nevertheless, as mentioned above, backward compliance for the traditional point-to-point camera to host control and receiving application was maintained.

PART 1 – Device Discovery

2 Device Discovery Summary

2.1 Overview

Device Discovery covers the sequence of events required for a controllable device to get a connection through its network interface and obtain a valid IP address using standard IP protocols, and for a control application to enumerate devices on the network. Once Device Discovery is completed, an application is ready to send control requests to a device.

Note: By definition, a **device** is a GigE Vision compliant **controllable device** and an application is a GigE Vision compliant **control application** software running on a host. Therefore, the terms **device** and **application** are used in the remainder of this document is order to simplify the standard text.

Device Discovery is separated in 3 sequential steps:

1. Link negotiation
2. IP configuration
3. Device enumeration

A device has one or more network interfaces connecting it to the network. The first step is for the device to determine which links are active and which ones are not. The Ethernet auto-negotiation procedure, as defined by IEEE 802.3, then deals with link speed negotiation between two partners.

The second step, IP configuration, uses standard IP protocols. It is initiated by the device. Its goal is to assign a unique IP address to the device. GigE Vision devices must support Dynamic Host Configuration Protocol (DHCP) and Link-Local Address (LLA). GigE Vision devices may support Persistent IP. Persistent IP is defined as static IP address that persists across power cycle or reset. It is kept in non-volatile storage in the device.

The third step, device enumeration, is initiated by the application to gather information about device presence on the network. If the application does not know the device IP address, this can be achieved on the application's subnet using a UDP broadcast command. If the client's IP address is known, this can be accomplished using unicast UDP. This step is realized using GVCP message exchange between the device and the application. This answer incorporates various pieces of information about the device, such as the manufacturer, device model, etc.

Warning: This specification does not identify how an application can retrieve an IP address of a device residing on a different subnet. This could be achieved using information provided by the DHCP server, by multi-casting a device enumeration request or using DNS. A future version of the specification may address this topic.

2.2 Goals

The following design goals are set as a target for IP Configuration and Device Discovery.

1. It must permit support for multiple network interfaces.
2. It should have distinctive steps for IP configuration and for device enumeration.
3. It must assign a valid IP address to the GigE Vision device.
4. It must enumerate all GigE Vision devices on the same subnet as the GigE Vision application.
5. It should offer provisions to find GigE Vision devices on a different subnet than the application.
6. Where possible, it should provide a plug-and-play solution that does not require any networking configuration.
7. It should ensure devices are reachable within 20 seconds after being powered-up or reset.
8. It should provide device specific information to GigE Vision application to uniquely identify devices (including model and manufacturer).
9. It should leverage on existing IP protocols whenever possible to ensure interoperability with IP networks and minimize development efforts.
10. It should optionally allow the user to force a persistent IP address to a device. This address would be used on all subsequent re-initialization of the GigE Vision device.
11. It should minimize IP stack complexity in the GigE Vision device. It should try to push the complexity on the application side where CPU power and memory are more readily available.

At the end of the Device Discovery process, the application must have identified all GigE Vision compliant devices on its subnet and each device must be ready to receive requests.

2.3 Scope

This part covers GigE Vision Device Discovery. This includes link negotiation, IP configuration and device enumeration.

Other protocols used to communicate between a device and application software (such as GVCP) are covered in other sections of this specification.

3 Physical Link Configuration

Before IP configuration can take place, a physical connection must be established between the device and the application. GigE Vision devices can use multiple physical links to increase the overall available bandwidth for data transmission. When multiple links are used, they can be combined together and data packets can be distributed among them to optimize bandwidth utilization. For instance, by combining two 1 gigabit Ethernet links, a device can offer up to 2 gigabit/sec of bandwidth.

This specification provides 4 different physical link configurations for a device:

1. **Single Link** configuration (SL configuration)
2. **Multiple Links** configuration (ML configuration)
3. **Static Link Aggregation Group** configuration (static LAG configuration, sLAG)
4. **Dynamic Link Aggregation Group** configuration (dynamic LAG configuration, dLAG)

The first 2 (SL and ML configurations) are controlled manually and each network interface is configured individually. Each network interface goes through IP configuration at start-up. So the device ends up with one different IP address for each physical network interface.

The last 2 (static LAG and dynamic LAG) use the IEEE 802.1AX specification to create a link aggregation group. In these configurations, a single virtual link is created. As such, only one IP address is seen by the application for the regrouped physical interfaces. The grouping of the physical links is performed at the MAC layer and is transparent to the application software.

The following table compares those configurations for GVSP transmitters and receivers:

Characteristic	SL	ML	Static LAG	Dynamic LAG
Number of network interfaces	1	2, 3 or 4	2, 3 or 4	2, 3 or 4
Number of IP address	1	One per network interface	1	1
Minimal number of stream channels	1	One per link	1	1
Load balancing	Not applicable	Manual using stream channels.	Round-robin distribution algorithm	Round-robin distribution algorithm
Physical link down recovery	Device is lost. Manual recovery.	Stream channels on broken link go down. Possibility to lose control channel. Manual recovery.	Packets redistributed on remaining physical links	Packets redistributed on remaining physical links
Grouping configuration	Not necessary	Each link is configured independently (not seen as a group).	Some (or all) links are automatically (or sometime manually) grouped on the device. Manual grouping must be performed on the PC (often called *teaming*).	Automatic on the device and PC through link aggregation control protocol (LACP) after configuration.

Note: It is allowed for a device to use a combination of LAG and ML.

3.1 Single Link Configuration

The Single Link configuration (SL configuration) is the simplest configuration since only one physical network interface is necessary. Hence all control, stream and message channels are attached to this interface.

No special steps are needed to indicate how links are grouped or how stream channels must be distributed. By necessity, all channels are attached to the only available physical link. When multiple physical links are available, one can service the SL configuration.

[R-001cd] A device MUST have at least one network interface. [R12-1cd]

[R-002cd] All devices MUST support SL configuration.

3.2 Multiple Links Configuration

When multiple network interfaces are available, one option is to configure them as independent links where each interface as a distinct IP address.

If a streaming device has multiple network interfaces, then it can optionally support the Multiple Link (ML) configuration.

[R-003cd] A device MUST support at most 4 different network interfaces. [R12-2cd]

[CR-004cd] If a device supports the ML Configuration, then each network interfaces MUST execute the IP configuration procedure independently when using the ML configuration. [CR3-3cd]

The number of supported network interface is reported by the Number of Network Interfaces register at address 0x0600 in the bootstrap registers memory space. Other bootstrap registers in the 0x0600 to 0x07FF address range are used to store link negotiation and IP configuration information of each network interface.

Each network interface has an index associated to it (from 0 to 3). Interface #0 is the main interface to control the device (control and message channels). It is the only interface supporting GVCP. For instance, Device Discovery is always performed on interface #0 to ensure the device is not seen multiple times with different IP addresses. Other network interfaces are only used to support additional stream channels. This also means that these other interfaces only output or receive GVSP packets; they are not supporting GVCP.

Note that interface #0 does not need to be the first physical interface on the casing of the device. A manufacturer might decide to clearly label interface #0, but he can also allow interface #0 to be the first one that goes active. If the cable on interface #0 is pulled out, then the control channel goes down and communication with the device is lost. LAG configurations provide more robustness in this situation.

Note: It is acceptable for a LAG to be considered as one link in the ML configuration.

[CR-005cd] When a device supports the ML Configuration, then it MUST provide independent storage for bootstrap registers related to IP configuration on each interface. [CR12-3cd]

[R-006cd] It is only possible to use FORCEIP message with interface #0. Other interfaces MUST use Persistent IP, DHCP or LLA to get their IP configuration. [R12-5cd]

For the ML Configuration, it is recommended to allow any supported stream channel to be associated to any network interface through bit 12 to 15 of the SCPx register (*network_interface_index* field of SCPx). However, it is acceptable to hard-code a stream channel to a specific network interface. In the latter case, the *network_interface_index* field is read-only.

[CR-007s] When the Multiple Link Configuration is supported, the specific interface to use for the stream channel MUST be specified in the SCPx register. [CR10-7s]

Note: A product is also allowed to present each network interface as a different GigE Vision device. In this case, suitable resources (such as bootstrap registers) must be provided on each interface, and each interface is considered independent from the others. This is equivalent to grouping multiple devices together into a single unit.

3.2.1 Load Balancing Considerations

Since different IP addresses are assigned to each network interface of the device, load balancing can be implemented at the transmitter by using a proper distribution algorithm without having to worry about intermediate network equipment (such as Ethernet switches) having to deal with a distribution algorithm. Internet Protocol will take care of routing traffic to the correct interface at the receiver side.

In the ML configuration, load balancing can be achieved by having multiple stream channels and by separately associating each of them to one of the different links to balance the overall bandwidth required from each of these links.

For instance, on a dual-port streaming device, one would need 2 stream channels that consume the same bandwidth in order to perform load balancing. Each stream channel would be associated to a different link. Hence the same amount of bandwidth would be consumed by the streaming on each interface.

3.3 Link Aggregation Group Configuration

IEEE 802.1AX formally defines link aggregation. This allows regrouping multiple links to form a link aggregation group (LAG), enabling the device to treat the LAG as if it was a single link. Contrary to the ML configuration, this is done in a transparent way from the application perspective.

To facilitate communications, IEEE 802.1AX defines a conversation as a set of packets among which ordering must be maintained upon reception. It is also important to note that IEEE 802.1AX does **not** define the distribution algorithm to be used at the transmission end of a link aggregation group.

GigE Vision builds on the IEEE 802.1AX standard to enable link aggregation. This section presents additional clarifications about using IEEE 802.1AX on a GigE Vision compliant device to facilitate inter-operability.

Note: It is important to understand that IEEE802.1AX capable Ethernet switches directly comply with the IEEE specification and thus might not be able to load balance the GEV stream traffic onto multiple outgoing ports as they typically use information in the Ethernet frame header (and sometime IP packet header) as input to their distribution algorithm. Those Ethernet switches don't have visibility at the GVSP level. Since LAG shows a single MAC/IP, then those switches cannot figure out how to distribute the GVSP traffic: the traffic might end-up on one outgoing port of the switch.

A GigE Vision streaming device might elect to support static Link Aggregation, dynamic Link Aggregation, or both.

3.3.1 Network Interface

Only one aggregator is permitted. Therefore, all active links associated to the LAG are bound to the aggregator.

[CR-008cd] If link aggregation configuration is supported, then the device MUST permit only a single aggregator in both static and dynamic LAG configurations.

[CO-009cd] If link aggregation configuration is supported, then the aggregator SHOULD use the MAC address of the lowest-numbered physical network interface associated to the LAG.

As a consequence, for a LAG configuration, only one "virtual" network interface is visible from the bootstrap registers. If a device only supports LAG and has no additional ML physical interface, then this LAG interface is shown as Network Interface #0. Otherwise it might be any of the available interfaces. The Link Speed register can be used to determine the actual bandwidth available out of any interface (LAG or not).

3.3.2 GVCP Impacts

[CR-010c] If link aggregation is supported, then each GVCP channel (control or message) is considered to be one conversation.

As such, a GVCP channel must always be sent on the same physical link of the LAG.

3.3.3 GVSP Impacts

In order to allow load balancing, GigE Vision loosely defines a conversation for a stream channel.

[CO-011st] If link aggregation is supported, then the distribution algorithm for stream channels SHOULD be round-robin across all the links composing the aggregation group.

A round-robin distribution algorithm allows for a uniform distribution of the bandwidth associated to a stream channel since all GVSP packets (except possibly the last one of each block) have the same size. So it adequately balances the bandwidth across available links on the LAG. A suitable packet size must be selected to ensure all physical links can handle it.

Because of this loose definition of conversation and the selected distribution algorithm, it is necessary for the receiver of a GVSP stream channel to be tolerant to out of order packets and accommodate longer timeouts than seen with Single Link configuration when looking for retransmission of missing packets.

Special provision must be taken for the inter-packet delay: it represents the minimal delay between packets of a stream channel traveling on a given physical link. Alternatively, it is preferable for a device to consider using the optional IEEE 802.3 PAUSE mechanism. Note that the full duplex flow control function resides architecturally below that of link aggregation. Therefore, each physical link deals with PAUSE independently.

Note: For link aggregation, all physical MAC part of the same aggregator use the same PAUSE configuration since they are presented as a single logical link.

The first physical link to use in the round-robin cycle is left as a quality of implementation; therefore, the first stream packet can be transmitted on any of the interface of the aggregator. It is acceptable to restart a new round-robin cycle at a new data block boundary.

3.3.4 Static LAG vs Dynamic LAG

The only difference between static LAG and dynamic LAG is the use of the IEEE 802.1AX Link Aggregation Control Protocol (LACP). In static LAG, any active link bound to the aggregator is considered part of the LAG (only one aggregation group per device). This assumes that all cables for these links are going to the same destination (typically a multi-port NIC). This is the intention since LAG is used to augment the available bandwidth for streaming (i.e. have a bigger pipe between the GigE Vision transmitter device and its destination).

But if the cabling is not correctly setup, then this creates a misconfiguration and prevents the device from operating correctly. The LACP is designed to ensure the system correctly binds the physical links that belong to the same interconnection. For dynamic LAG, this guaranties that all connections in an aggregator are between the same 2 partners, as defined in IEEE 802.1AX. There is no such guaranty for static LAG and thus additional burden is put on the person setting-up the system to correctly connect the cabling.

The method to configure the LACP on a GigE Vision device supporting dynamic LAG is left as a quality of implementation.

3.3.5 Events for LAG

[CO-012cd] If link aggregation is supported, then the device SHOULD offer a GEV_EVENT_LINK_SPEED_CHANGE event triggered by a change of the aggregated speed of the links participating in the aggregation group.

This event can be fired by the device when the aggregated speed changes due to a physical link going up or down in the aggregation group.

Note: The actual method to enable GEV_EVENT_LINK_SPEED_CHANGE event is defined through the Standard Feature Naming Convention.

The actual speed value is reported by the corresponding Link Speed register.

4 IP Configuration

This section lists the requirements a device has to follow to obtain a valid IP address.

[R-013cd] All devices MUST execute IP configuration upon power-up, device reset or after successful Ethernet link negotiation. [R3-1cd]

[R-014cd] A device MUST support the following IP configuration protocols:

- Dynamic Host Configuration Protocol (DHCP)
- Link-Local Address (LLA) [R3-2cd]

Optionally, a device may support a user configurable Persistent IP address. This address is stored in the device non-volatile memory (bootstrap registers) to be used on device power-up or reset. When a persistent IP address is used, it is up to the user to ensure the selected IP address will not cause any conflict on the network.

In this section, a valid IP address represents the combination of:

1. An IP address
2. A corresponding subnet mask
3. A default gateway

This specification asks for the IP address to be assigned in a reasonable amount of time (ideally in less than 20 seconds in the worst case). The user expects the device to be up and running within seconds after powering the unit. This could be difficult to achieve if hundreds of devices are powered-up simultaneously, thus overloading the DHCP server. In order to limit the IP configuration time, each configuration scheme, except link-local address, can optionally be disabled. Note that the application might have its own restriction to obtain an IP address: some operating systems might take up to 1 min. to assign a LLA IP address to a network interface.

This section explains how the device selects which IP configuration protocol to use and how specific options of these protocols can be enabled.

Note: A GigE Vision product may host multiple devices. In this case, each device is considered independent and needs to follow the requirements and objectives pertaining to devices.

4.1 Protocol Selection

Each device has to iterate through a selection of IP configuration scheme.

[R-015cd] On a device, the sequence of execution of each IP configuration protocol MUST be

1. Persistent IP (if supported and enabled)
2. DHCP (if enabled)
3. Link-Local Address [R3-5cd]

Note: The above requirement can be interpreted in 2 ways which are both acceptable.

The first interpretation has the DHCP client stopped when the state machine moves from DHCP stage to Link-Local Address. This is a strict sequential approach which better matches Figure 4-1, but contradicts a recommendation of RFC3927.

The second interpretation keeps the DHCP client running according the recommendation of RFC3927. Therefore, even if a LLA IP address was assigned, the DHCP client can still obtain an IP address from a DHCP server. This interpretation is in-line with the behavior of typical operating systems, such as Windows and Linux. Note that when this situation happens, the device will replace its LLA IP address with the newly obtained DHCP IP address. Using that interpretation, recommendations from RFC3927 should be followed for general DHCP client operation, including address renewal (the device could loose its IP address if it cannot renew it) and retransmission delay after the initial 2 retransmissions have failed.

[R-016cd] Factory defaults of a device MUST have Persistent IP disabled and DHCP enabled. LLA is always enabled. [R3-6cd]

This ensures a standardized way to configure the IP address of a new device, even when devices from different manufacturers are put on the same network segment.

Note that GVCP also offers a command, FORCEIP_CMD, to force a static IP address into interface #0 of a device. This address overrides the IP address obtained using the normal IP configuration selection described in this section. But this address is lost on power cycle or device reset.

The procedure to select the IP assignation protocol is illustrated below.

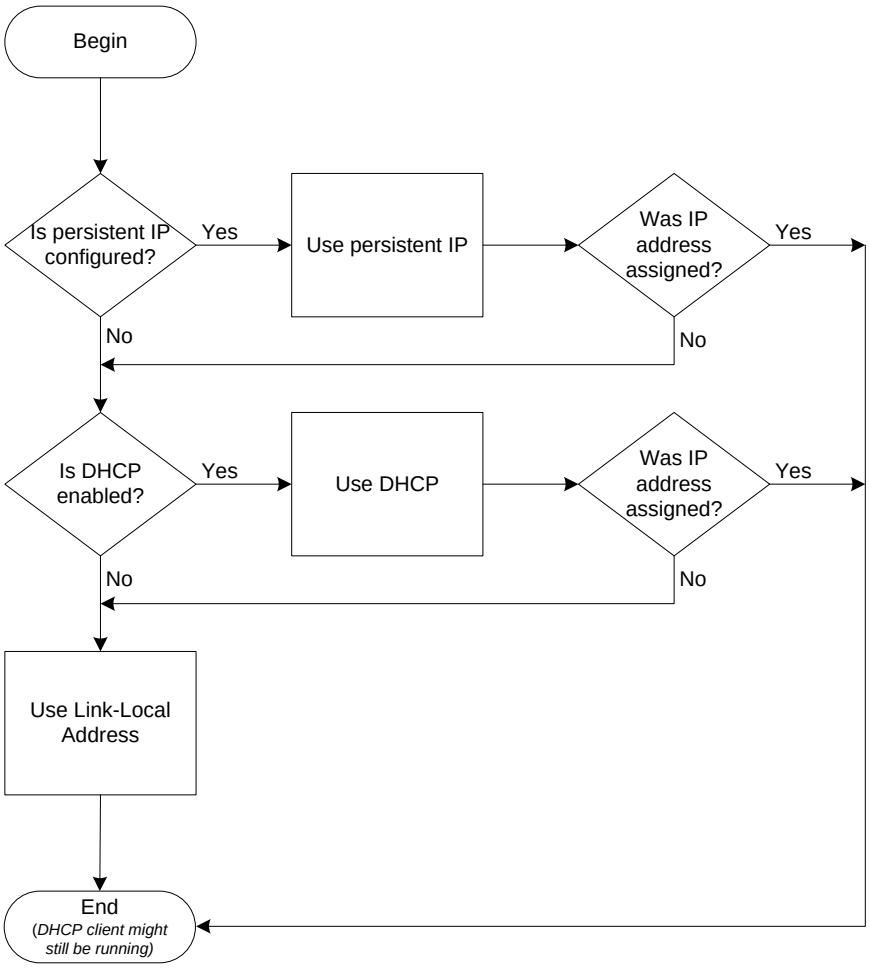


Figure 4-1: Protocol Selection Flowchart

[R-017cd] Once a valid IP address is assigned to the device, its associated information MUST be copied into the following bootstrap registers:

1. Current IP address (address 0x0024 for network interface #0)
2. Current Subnet Mask (address 0x0034 for network interface #0)
3. Current Default Gateway (address 0x0044 for network interface #0) [R3-7cd]

4.2 Persistent IP

A device may elect to support Persistent IP on any of its network interface.

[CR-018cd] If a device supports Persistent IP, then it MUST do so using the appropriate bootstrap registers and it MUST provide some non-volatile memory to store those settings. [CR3-8cd]

The following bootstrap registers are used to support Persistent IP:

1. Network Interface Capability (address 0x0010 for network interface #0): bit 31 indicates if Persistent IP is supported by this device (bit set) or not (bit cleared).
2. Network Interface Configuration (address 0x0014 for network interface #0): bit 31 indicates if Persistent IP has been activated (bit set) or not (bit cleared) by the user.
3. Persistent IP Address (address 0x064C for network interface #0): this is the Persistent IP address assigned by the user. It is up to the user to ensure this is a valid IP address.
4. Persistent Subnet Mask (address 0x065C for network interface #0): this is the subnet mask associated with the Persistent IP address.
5. Persistent Default Gateway (address 0x066C for network interface #0): this is the default gateway when Persistent IP is activated.

[CR-019cd] If a device supports Persistent IP, then the Persistent IP address, its associated subnet mask and its default gateway MUST be stored in non-volatile memory of the device. This is true for all network interfaces supporting Persistent IP. The state of a Persistent IP register (Persistent IP Address, Persistent Subnet Mask and Persistent Default Gateway) is stored in non-volatile memory as soon as the corresponding Persistent IP register is set by an application. [CR3-9cd]

If a device does not have non-volatile memory, then it cannot support Persistent IP.

It is up to the end-user to correctly assign the persistent IP address of the device and to ensure it does not create a conflict on the network. RFCs relevant to IP address assignment should be respected to avoid IP configuration problems.

Note: It is not permitted to have a Persistent IP address of 0.0.0.0 or 255.255.255.255 when the interface is configured. If such a configuration is set, then the device moves to the next valid IP configuration method.

[CO-020cd] If a device supports Persistent IP, then it SHOULD probe the Persistent IP address with ARP before using it in order to detect a potential address conflict, as indicated in RFC5227. If such a conflict is detected, the device MUST NOT use this IP address and SHOULD signal this problem to the user (for example, by flashing a status LED). The way the device signals this problem is left as a quality of implementation. The device MUST then move to the next IP configuration scheme. [CO3-11cd]

When an unidentified IP address is assigned through Persistent IP, it is possible for an application to gain control of the device using the FORCEIP_CMD message of GVCP. The application can then modify the Persistent IP information of the bootstrap registers to a valid state or simply disable Persistent IP.

4.3 DHCP

DHCP is a very well known scheme to obtain an IP address. All network interfaces of a device must support DHCP.

[R-021cd] A device MUST support DHCP. Refer to RFC2131 (Dynamic Host Configuration Protocol) and RFC2132 (DHCP Options and BOOTP Vendor Extensions) for more information on DHCP. [R3-12cd]

[O-022cd] A device SHOULD implement a DHCP enable flag in non-volatile storage to indicate if DHCP should be enabled on next device reset. This flag is part of the Network Interface Configuration bootstrap register (one per network interface). [O3-13cd]

[CR-023cd] If this enable flag is supported and is set to TRUE, then the device MUST try to obtain an IP address using DHCP. [CR3-14cd]

[CR-024cd] If this enable flag is supported and is set to FALSE, then the device MUST moves to the next IP configuration protocol given by the protocol selection flowchart without using DHCP. [CR3-15cd]

[CR-025cd] If this non-volatile storage is not present, then the device MUST react as if DHCP was enabled (the DHCP enable flag is hard-coded to TRUE and is not user configurable). [CR3-16cd]

This DHCP enable flag is located in the bootstrap registers. The following bootstrap registers are used to support DHCP:

1. Network Interface Capability (address 0x0010 for network interface #0): bit 30 indicates if DHCP is supported by this device (bit set) or not (bit cleared). This bit is always 1.
2. Network Interface Configuration (address 0x0014 for network interface #0): bit 30 indicates if DHCP has been activated (bit set) or not (bit cleared) by the user.

The following figure illustrates a DHCP message. The way this message is constructed is given by RFC2131. This specification provides additional information on how specific fields can be used by GigE Vision devices.

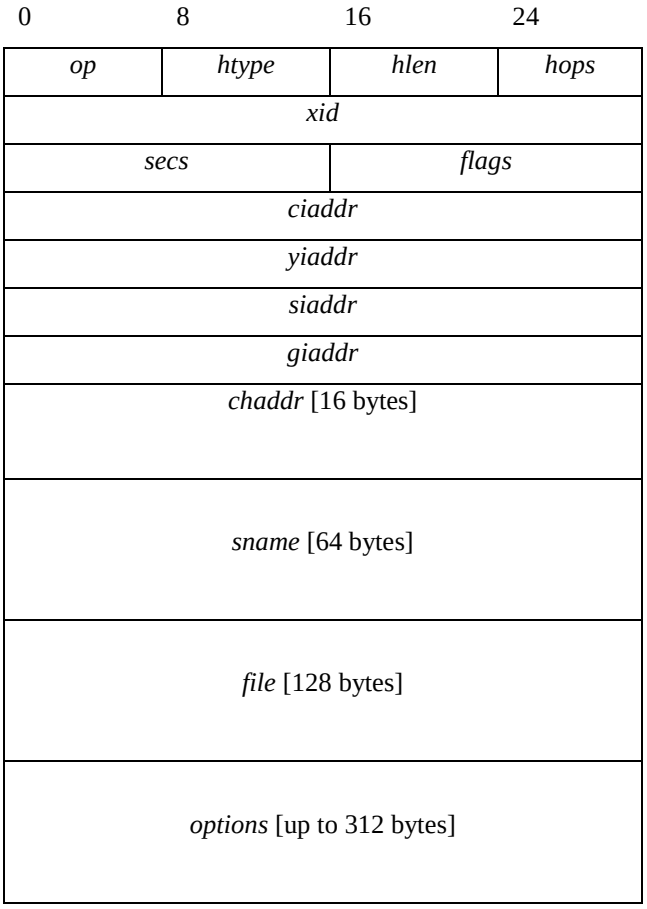


Figure 4-2: DHCP Message

[O-026cd] A device SHOULD support the following DHCP options (see RFC2132):

- **Subnet mask option** (code 1). The subnet mask is useful to identify to which subnet a device belongs. This is particularly true for multi-homed configurations where the IP address may not be sufficient to identify to which network card the device is connected.
- **Router option** (code 3). The router option is used to identify a default gateway. This is useful when the application does not reside on the same subnet as the device. [O3-18cd]

4.3.1 DHCP Retransmission Strategy

RFC2131 provides general guidelines for DHCP retransmission. Because GigE Vision iterates through various IP configuration mechanism and the user expects the device to be accessible quickly upon power-up (within 20 seconds for this specification), this specification requires a different strategy. This ensures the device obtains an IP address in a reasonable amount of time.

When using DHCP, a device may time out for two different messages:

1. DHCPDISCOVER: When the device sends a DHCPDISCOVER message, the server is expected to answer with a DHCPOFFER message.
2. DHCPREQUEST: When the device sends a DHCPREQUEST message, the server is expected to answer with a DHCPACK or DHCPNAK message.

The device times out and retransmits the message if it does not receive any valid answer message from the DHCP server. A maximum of 2 retransmissions are allowed (for a total of 3 DHCPDISCOVER messages followed by 3 DHCPREQUEST messages in the worst case).

[O-027cd] The DHCP retransmission strategy of a device SHOULD follow the following algorithm:

1. On the first transmission, if the device does not receive an answer message from a server within **2 seconds**, optionally randomized by the value of a uniform random number chosen from the range –1 to +1, the device performs a first retransmission.
2. On the first retransmission, if the device does not receive an answer message from a server within **4 seconds**, optionally randomized by the value of a uniform random number chosen from the range –1 to +1, the device performs a second retransmission.
3. On the second retransmission, if the device does not receive an answer message from a server within **6 seconds**, optionally randomized by the value of a uniform random number chosen from the range –1 to +1, the device moves to the next IP configuration protocol given by the protocol selection flowchart. [O3-19cd]

Using this algorithm, a device will typically pass 12 seconds in the DHCP stage when no DHCP server is present before moving to the next IP configuration scheme (15 seconds in worst case).

4.3.2 DHCP Lease Expiration

When the device is operated in DHCP mode, it must follow RFC2131. Of particular importance is the expiration of the lease for the IP address provided by the DHCP server. When this happens, the device must stop using this IP address and restart the IP configuration cycle. When no DHCP server is present on the network, then the device will fall back to LLA (assuming PersistentIP has not been enabled).

4.4 Link-Local Address

Link-Local Address provides a simple scheme for a device to select a valid IP address. Essentially, the device randomly claims an IP address in a specific range and then verifies if another device on the subnet is using this address. If the address is used, the device restarts this process until it finds an available IP address.

[R-028cd] The device MUST support Link-Local Address (LLA). Refer to RFC3927 (Dynamic Configuration of IPv4 Link-Local Addresses) for a complete description of Link-Local Address. [R3-20cd]

[R-029cd] On a device, LLA MUST always be activated (it cannot be disabled). [R3-21cd]

LLA is sometimes called “Auto IP” and is mapped to IP address range 169.254.xxx.xxx. More precisely, IP addresses ranging from 169.254.1.0 to 169.254.254.255 are available for LLA.

The following bootstrap registers are used to support Link-local address configuration:

1. Network Interface Capability (address 0x0010 for network interface #0): bit 29 indicates if Link-local address is supported by this device (bit set) or not (bit cleared). This bit is always 1.
2. Network Interface Configuration (address 0x0014 for network interface #0): bit 29 is always set since LLA MUST always be supported and activated (default IP configuration scheme).

5 Device Enumeration

GigE Vision provides 2 mechanisms to enumerate devices:

1. GVCP Device Discovery (mandatory)
2. Multicast DNS / DNS Service Discovery (optional)

5.1 GVCP Device Discovery

[R-030cd] Once the device has completed IP configuration, it MUST answer device discovery requests coming from any application if it has a valid IP address. [R4-1cd]

[R-031cd] A device MUST NOT answer device discovery requests before it has completed IP configuration. [R4-2cd]

Device discovery messages are an integral part of GVCP.

An application may perform Device Enumeration to find devices, but it is not required to do so. This could be helpful to minimize bandwidth usage when the IP address and other information from the device are known ahead of time.

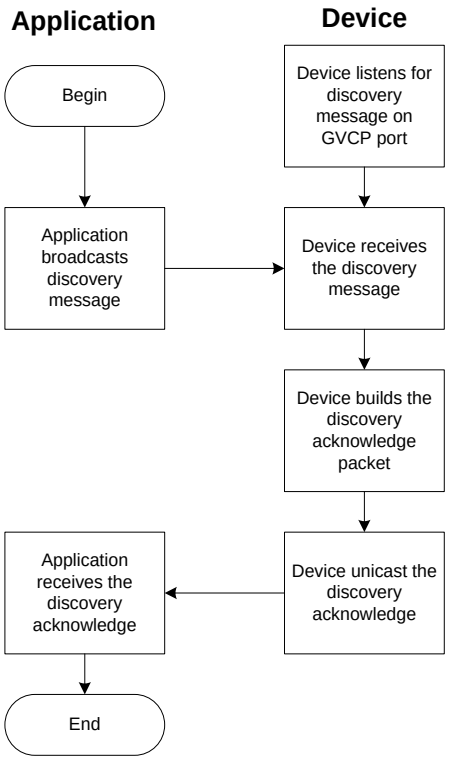


Figure 5-1: Device Discovery Flowchart

5.1.1 Broadcast Device Discovery

Broadcast Device Discovery messages can be used by an application to look for devices residing on the same subnet. It can be realized using a UDP broadcast message with a destination IP address of 255.255.255.255. This is defined as a "limited broadcast" by RFC1122 (Requirements for Internet Hosts -- Communication Layers). Note this message will not cross routers. This is why only devices on the same subnet will receive it. Using broadcast device discovery, it is not possible to enumerate devices on a different subnet than the network card(s) of the application.

This specification does not define to which network card(s) the limited broadcast message should be sent when the application resides on a multihomed host. Note that RFC1122 takes no stand on this issue either. To overcome this ambiguity, an application can use a "subnet directed" broadcast.

Definitions of broadcast types are provided in section 3.3.6 of RFC1122. Any time the GigE Vision specification refers to broadcast, this implicitly covers any type of broadcast.

[R-032cd] In the answer message, the device MUST set the source IP address, subnet mask and default gateway equal to the IP information obtained during IP configuration. [R4-5cd]

5.1.2 Unicast Device Discovery

Unicast Device Discovery messages can only be used when the device IP address is known beforehand by the application. It is realized by sending a UDP packet directly to the device IP address.

This specification does not state how the application knows the IP address of a device ahead of time. Forcing a fixed IP address using Persistent IP scheme or using a DNS are possible ways to do this.

[R-033cd] A device MUST answer a unicast device discovery message using a unicast answer message to the application. [R4-6cd]

5.1.3 Associating the Device to the Enumeration List

Some criteria are recommended to allow easy identification of the device.

[O-034cd] In order to easily associate a device to its entry in the discovery list, the device SHOULD have a serial number and MAC address label on the device casing. [O4-7cd]

This could be matched against the information returned by the device discovery message.

5.2 Zeroconf Discovery

In GigE Vision 1.x, GVCP is only defined to listen on UDP port 3956. This limitation means that to have multiple independently controlled devices inside a single physical box requires using multiple IP addresses and GVCP stacks, at a minimum. A GigE Vision product can have several virtual MAC addresses behind a single Ethernet port (as if there is a switch in front) but this approach does not scale well.

GigE Vision version 2.0 defines how devices can advertise and discover each other using a combination of multicast DNS and DNS-SD, called Zeroconf Discovery. In addition to providing a more standard and robust discovery mechanism, one feature is that it natively provides support for various types of multi-service configurations.

To start, this mechanism separates the concept of "hosts" (typically a single piece of hardware that plugs into a network) from "services" (a single instance of some application that provides some service over the network, such as an HTTP server). A single host may have more than one Ethernet interface as well as more than one IP address (along with both IPv4 and IPv6). A service has three main components: type (fixed for GVCP), name (identifies the particular instance), and the UDP/TCP port number the service is running on. Each service can also have a unique list of TXT records that can contain details about the particular instance. This allows a single host to have N number of IP interfaces and M number of services with no explicit relationship between the two. In the context of GigE Vision, each service instance corresponds to a GVCP control channel.

The following use cases can thus be realized through this mechanism.

Standard device

This is the SL configuration. It advertises a single host with a single service.

Configuration:

- 1 Ethernet port
- 1 IP address
- 1 GVCP channel (on GVCP port 3956)

Standard device with link-aggregation

This advertises a single host with a single service (the multiple links are treated as one logical link).

Configuration:

- Multiple Ethernet ports (bound through link-aggregation, sLAG or mLAG)
- 1 IP address
- 1 GVCP channel (on GVCP port 3956)

Standard device with multiple links with no link aggregation

This could advertise a single host with multiple IP addresses and a single service and maps to the ML configuration. It would then be up to the application to decide which one to connect to (it could always use the first one listed) and it would be implementation-defined which links are used for any particular streams. This is equivalent to the behavior in GigE Vision 1.x.

Configuration:

- N Ethernet ports
- N IP addresses
- 1 GVCP channel (on GVCP port 3956)

Multi-controller device with single link

This advertises a single host with multiple GVCP services. Each service is seen as a SL configuration sharing the same physical interface.

Configuration:

- 1 Ethernet port,
- 1 IP address,
- multiple GVCP channels (using different UDP ports, first channel must use GVCP port 3956)

Note: In this case, all the « independant » GVCP stacks behind the same Ethernet port and IP address share the same IP address. Therefore, if an application changes the IP address for one device, then the IP address of the others will follow that change.

Multi-controller device with multiple independent links

Each link would advertise as a different host (allowed by mDNS specification) and each of those hosts would advertise a single service. Generally each link would only respond to queries matching the unique host/service names corresponding to that link, so hosts on the other end of the links would see only the interfaces to which they are connected to as well as be able to deduce which services are reachable via which specific links (most mDNS responder clients are able to indicate which local link a service is seen through). This maps to the SL configuration, but with a different physical interface for each service.

Configuration:

- N Ethernet ports
- N IP addresses
- N GVCP channels (all channels must use GVCP port 3956)

5.2.1 Multicast DNS (mDNS)

Multicast DNS (mDNS) is an implementation of the traditional DNS protocol but decentralized and based upon issuing queries through multicast to other mDNS clients that are running. These queries are simply standard DNS queries that can be for types such as A/AAAA records (for IPv4/IPv6 name resolution) and SRV records for services. The protocol implementation makes use of many techniques to reduce the overhead of sending and responding to queries when there are many hosts on the network. Some of these techniques consist of reducing redundant traffic through record caching, host announcement, known answer suppression, duplicate answer suppression, and response aggregation.

DNS is used to assign domain name (i.e. text string) and providing the association between those domain names and the IP address of the device. This is quite similar to a phone book where the name of a person (equivalent to a domain name) is associated to its phone number (equivalent to the IP address).

[O-035cd] A device SHOULD implement mDNS as specified in the Multicast DNS Internet Draft.

As quick reference, this Internet Draft provides the following information:

- The IPv4 multicast address allocated for mDNS is 224.0.0.251.
- The IPv6 link-local multicast address allocated for mDNS is FF02::FB.

- mDNS uses UDP port 5353.
- mDNS only uses UTF-8 to encode resource record names.
- mDNS uses the DNS top-level domain ".local.".

The device host name would take the form of "<*host_name*>.local.", where <*host_name*> is a unique label provided by the device manufacturer. It is important for this label to be unique to avoid name conflict on the subnet. The host name should be constructed by concatenating the name of the device manufacturer to the name of the device followed by the device MAC address in upper-case hexadecimal (ex: "Manufacturer-Camera-01020300A2F3.local.").

5.2.2 DNS Service Discovery (DNS-SD)

DNS Service Discovery (DNS-SD) is a mechanism to use DNS to look up particular services identified by name. Hosts can effectively "browse" for a service they are interested in by making a query for a particular SRV record. This query can be directed to either a traditional DNS server or by using mDNS to search the local network. Hosts can register services to advertise either locally, or dynamically update them on a centralized DNS server to allow services to be discovered on a wider basis (even globally).

A service would be defined as a GVCP control channel. A multi-stream device with a single GVCP control channel would have a single service. A multi-stream device presented as independent devices (today implemented by some vendors as a multi-port device with multiple MAC addresses and IP addresses) could have multiple services advertised on a single device, or multiple single-service devices.

The DNS Service Discovery has 2 main tasks:

1. Enumerate the list of service name
2. Translate from the service name to the associated IP address

[O-036cd] A device SHOULD implement DNS-SD as specified in the DNS-Based Service Discovery Internet Draft.

[CR-037cd] If DNS-SD is supported, then the full-qualified service name referred to via DNS MUST be "_gvcp._udp".

Note that the first instance of the GVCP service must advertise on the IANA-reserved port of 3956 used for GVCP. This is required for backwards compatibility with older software based on specification 1.x. However, devices with more than one control port (such as for multiple, independent camera interfaces) can advertise additional service instances on any desired port number.

The content of the TXT records for GigE Vision mostly match the existing items in the current DISCOVERY_ACK packet.

Note that the TXT records are allowed to contain user-defined keys. In that situation, size constraint should be validated against the DNS-SD RFC has this might limit the maximum size of a key.

[CR-038cd] if DNS-SD is supported, then its TXT record MUST support at least the following keys:

SpecVer=<decimal major>.<decimal minor>	Matches the Version bootstrap register (required)
DevMode=<uppercase hex value string MSB first>	Matches the Device Mode bootstrap register (required)
MacAddr=<12-character uppercase hex MAC string>	MAC address provided to give a unique identifier between devices from all camera vendors (required). ex: '001F3B67C2C9'
Vendor=<string>	Matches the Manufacturer Name bootstrap register (required)
Model=<string>	Matches the Model Name bootstrap register (required)
DevVer=<string>	Manufacturer-specific versioning information (optional)
VenSpecific =<string>	Manufacturer-specific string (optional)
SerialNum=<string>	Different vendors may use different serial number schemes so this is provided as a string to match the bootstrap implementation (if supported by the device)
UserDefName=<string>	Matches User-defined Name bootstrap register (if supported by the device)
instance=<8-bit unsigned decimal>	Unique identifier to clearly identify individual instances on a multiple instance device. The application can use this to represent it as a multi-instance device rather than multiple single-instance devices as well as differentiate multiple services with the same MAC address (optional – should not be present on a single-instance device). Typically, the instance number starts at 0 and increments with each new individual instance residing on the device.

Note: One item to note from the above keys is the absence of the IP configuration settings present in the current DISCOVERY_ACK mechanism. This is for several reasons: Firstly, it contradicts the guidelines of DNS-SD in that it presents duplicate information that should instead be queried by resolving the service instance and retrieving an A (IPv4) or AAAA (IPv6) record. Additionally, it takes a lot of extra space in the packet. The main advantage to having these parameters in the packet appears to be that they make it easier for an "IP configurator" type program to scan for devices and modify their IP settings. In the normal case where the device is not misconfigured for the current network, it can simply retrieve these parameters by opening a GVCP control session and reading the bootstraps. If the device is misconfigured and a control session cannot be opened, such an application can still send a broadcast DISCOVERY_CMD packet and discover that information. It would need to be able to establish a GVCP control session first to change any of those settings anyways (potentially using FORCEIP_CMD). Since the times an application is expected to need this extended IP configuration info besides just the current IP address is very limited, it does not need to be a part of the normal DNS discovery mechanism.

Camera-side Example

An example camera would define its automatic hostname (if not provided by DHCP or other mechanism) to a unique id such as "CameraVendor-0123456".

It could then advertise two camera instances (say it is a 2-port analog to GigE Vision converter): "CameraVendor 0123456 (Port 1)._gvcp._udp" on port 3956 and "CameraVendor 0123456 (Port 2)._gvcp._udp" on port 15000 (camera-defined). The text records registered for each instance are below.

Instance 1

```
SpecVer=2.0
DevMode=80000001
MacAddr=000102030405
Vendor=My Vendor Name
Model=My Model Name
DevVer=1201.2345beta22
VenSpecific=My Device Info
SerialNum=30405
UserDefName=MyConverter
instance=0
```

Instance 2

```
SpecVer=2.0
DevMode=80000001
MacAddr=000102030405
Vendor=My Vendor Name
Model=My Model Name
DevVer=1201.2345beta22
VenSpecific=My Device Info
SerialNum=30405
UserDefName=MyConverter
instance=1
```

Note: Host name and service name have different set of rules. A host name is normally not visible. The goal is to discover services, not the name of the station hosting the service. The service name does not have to be built from the host name. There might be some commonality, but it is not necessary

Host-side Example

A client application would typically create a "browse request" with the mDNS stack for the service "._gvcp._udp" on the "local." domain. Alternatively it could query it on a non-local domain for cameras that are registered with a centralized DNS server.

The client application would then typically receive callbacks indicating when matching service instances are added or removed. In this case we would see additions of our two camera instances "CameraVendor 0123456 (Port 1)._gvcp._udp" and "CameraVendor 0123456 (Port 2)._gvcp._udp".

For each instance, the client application would then submit a request for SRV and TXT records on the instance names that it desires more information on. For the first port instance, this would result in returning

the hostname “CameraVendor-0123456.local.” and the port 3956 while the second port would return the same hostname but port 15000.

Lastly, if the application wants to connect to the device or display its IP addressing information, it can then submit a request for A (IPv4) or AAAA (IPv6) records for that hostname.

6 Device Attachment and Removal

An integral part of Device Discovery is live attachment or removal of a device from the network. The 'live' designation indicates the application is started and it has already enumerated the devices when the topology of the network is changed.

[O-039ca] An application SHOULD be able to dynamically react to a change of device network topology (adding or removing a device from the network). [O5-1ca]

6.1.1 Removal

Live removal is mainly handled by the control protocol. The application is going to timeout on the message commands it has sent (no acknowledge is received from the device when the application sends a heartbeat). Or alternatively, a control and receiving application can timeout on the video stream not coming anymore from the GVSP transmitter.

The way the application signals a device removal to the user, when this feature is supported, is left as a quality of implementation.

6.1.2 Attachment

On live attachment, usage of DHCP lets the server recognize a new device has been added to the network when it receives the DHCP request. This way, the server can nicely react and inform the application of the additional device. This unfortunately requires a close relationship between the server and the application. This might be difficult to achieve, especially if the server and the application do not reside on the same host.

Alternatively, another way for the application to know a new device is added to the network is to periodically send a DISCOVERY command. But sending periodical broadcast message consumes bandwidth out of the network, especially if many devices are to answer each time. One way to minimize bandwidth usage is to offer the user with a control to refresh the list of devices.

Finally, the multicast DNS and DNS Service Discovery is yet another mean to find new devices.

Because devices are not notified when other devices are added to a network, they are unaffected by the attachment of a new device (except for the amount of network bandwidth required to configure the new device).

The way the application signals a device attachment to the user, when this feature is supported, is left as a quality of implementation.

PART 2 – GVCP

7 GVCP Summary

7.1 Overview

GVCP is an application layer protocol relying on the UDP transport layer protocol. It basically allows an application to configure a device (typically a camera) and to instantiate stream channels (GVSP transmitters or receivers, when applicable) on the device, and the device to notify an application when specific events occur.

GVCP provides necessary support for only one application (the primary application) to control a device (to write to a device). Nevertheless, it is possible for many applications to monitor a device (to read from a device) if this is allowed by the primary application. It is also possible for an application to request control of a device already under the control of a primary application provided that this is supported by the device and allowed by the primary application. Under GVCP, the application is the master and the device is the slave. Command requests are always initiated by the application.

Command and acknowledge messages must each be contained in a single packet. The application must wait for the acknowledge message (when requested) before sending the next command message. This creates a very basic handshake ensuring minimal flow control. The acknowledge message provides feedback that a command was actually received by the device and it also provides any response data requested by the command.

The current version of the specification uses UDP IPv4 as the transport layer protocol. Because UDP is unreliable, GVCP defines mechanisms to guarantee the reliability of packet transmission and to ensure minimal flow control.

7.2 Goals

The goals for GigE Vision Control Protocol (GVCP) are:

- Allow command and acknowledge messages to be exchanged between a GigE Vision device and a GigE Vision application.
- Provide a way for the application to instantiate a stream channel from the device, when applicable.
- Define a mechanism for a GigE Vision device to send asynchronous event messages to a GigE Vision application (using an event generator).
- Provide a uniqueness access scheme so only one application can control the device.
- Minimize IP stack complexity in the GigE Vision device.

7.3 Scope

This part provides:

- Specification of the control protocol.
- Description of GVCP headers.
- List of all command and acknowledge messages.

- List of the different status codes representing error on the device.
- List of the different asynchronous event codes.

This control protocol does not cover GigE Vision Streaming Protocol (GVSP). This is discussed in a different part of the standard.

8 GVCP Transport Protocol Considerations

For the GigE Vision specification,

[R-040c] GVCP MUST use UDP with IPv4 as the transport protocol. [R7-1c]

[R-041c] Device and application MUST NOT use any IP options in the IP datagram for GVCP. [R7-2c]

This way, the IP header size is fixed at 20 bytes. This is to allow efficient hardware implementation of UDP/IP offload engines.

This section defines various mechanisms implemented by GVCP to guarantee the reliability of the transmission between the application and the device.

8.1 UDP

UDP is a connectionless protocol that adds no reliability, flow-control or error recovery to IP. Because of this lack of functionality, GVCP imposes restrictions on how connections are handled.

[R-042c] The first GVCP port of the device MUST use UDP port 3956. [R7-3c]

This standard GVCP port has been registered with IANA (http://www.iana.org/assignments/port-numbers). The application can use any available dynamic port number. When an application sends a GVCP packet, it uses the device GVCP port (3956) as the destination and its dynamically allocated UDP port number as the source. The device answers GVCP requests by using the application dynamic UDP port as the destination and its GVCP port (3956) as the source. For compatibility reason, it is necessary that a device always present its first instance using the standard GVCP port number. This allows non-mDNS enabled application to enumerate the device using the GVCP DISCOVERY command.

Note: The situation where the device uses mDNS/DNS-SD to advertise its GVCP services is described in section 5.2.

8.1.1 Fragmentation

Fragmentation defines the way a large message is broken into smaller segments suitable to be transmitted over the IP protocol.

[R-043c] In GVCP, command and acknowledge messages MUST each be contained in a single packet. [R7-4c]

This is to guarantee the packet will never necessitate IP fragmentation. The communication stack of the device can thus be simplified as it does not need to handle IP de-fragmentation. A device may reject fragmented IP datagram.

For IP, the maximum transmission unit (MTU) to avoid IP fragmentation is 576 bytes (i.e. the maximum size datagram that all hosts are required to accept or reassemble from fragments is 576 bytes).

[R-044c] Applications and devices MUST send GVCP packets containing at most 576 bytes. [R7-6c]

These 576 bytes include the IP header, UDP header, GVCP header and payload (Ethernet header is not part of the 576 bytes). The following table lists the number of bytes required by each header. Note that GVSP might use packets larger than 576 bytes.

Table 8-1: GVCP Packet Header Size

Layer	Size (in bytes)
IP header (options not allowed)	20
UDP header	8
GVCP header	8
Max. GVCP payload	540
Total	**576**

Note that even though the application and the device might handle packets larger than 576 bytes without IP fragmentation, other network components (such as switches and routers) might inflict this restriction.

This limit of 576 bytes maximum packet size imposes a restriction on the GVCP message themselves. This is incorporated in the message definition provided by this specification.

8.1.2 Packet Size Requirements

To simplify packet handling, the following restriction is established.

[R-045c] GVCP payload MUST have a size multiple of 32 bits. [R7-7c]

The GVCP header is itself a multiple of 32 bits.

8.1.3 Reliability and Error Recovery

Reliability ensures any given packet has been correctly received by the destination. In GVCP, this is realized by the application optionally requesting an acknowledge from each command message. This is illustrated in the following figure.

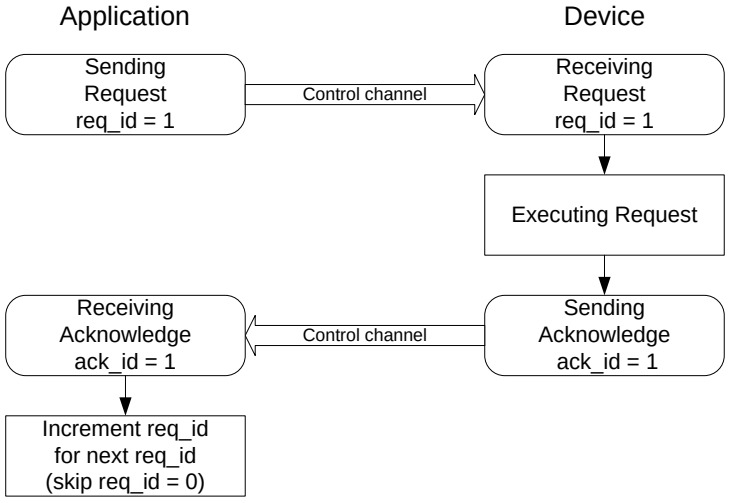


Figure 8-1: Use of Acknowledge

After sending a command, the application waits for the acknowledge (when requested). In order to detect if a message was not received by the device, the application implements a counter to signal timeout conditions. Note that an application is not required to request an acknowledge. This is left as a quality of implementation but can be useful to manage flow control.

[O-046ca] When an acknowledge is requested, if after a user-configurable timeout the application did not receive the acknowledge, it SHOULD send the command again. [O7-8ca]

[R-047ca] When resending the command, the application MUST leave the *req_id* field unchanged. [R7-9ca]

This provides a mechanism for the device to determine if the command is a retransmission.

If the application reaches the maximum number of retries, it reports the error to the upper software layer. The way this error is reported by the application is left as a quality of implementation.

[O-048ca] For a control channel, the application SHOULD provide a user programmable timeout value for GVCP message. [O7-10ca]

This acknowledge timeout may have a default value of 200 ms. The Pending Timeout register (at address 0x0958) provides the device's guideline for the worst-case timeout to expect from it.

Note: This specification does not provide a requirement on the default acknowledge timeout value since this is considered system dependent.

[O-049ca] The application SHOULD provide a user programmable number of retries value for GVCP message. [O7-11ca]

The number of retries may have a default value of 3.

Because of the symmetry between control and message channels, the device's bootstrap registers provide transmission timeout and retry count registers for the message channel (when this channel is available).

[R-050ca] The *req_id* field of GVCP messages MUST be incremented from one message to the next by the application. [R7-12ca]

Note: Because of broadcast commands, a device cannot assume that the *req_id* field is going to increment by one from one command to the next on its control channel. It can increment by more than 1. The *req_id* field is essentially present to facilitate the association of the acknowledge to a command by the application and for a device to determine if the same command was retransmitted from the originating application.

The initial value of *req_id* is not specified, but it cannot be 0. The *req_id* may be reset to the initial value when the control channel is closed.

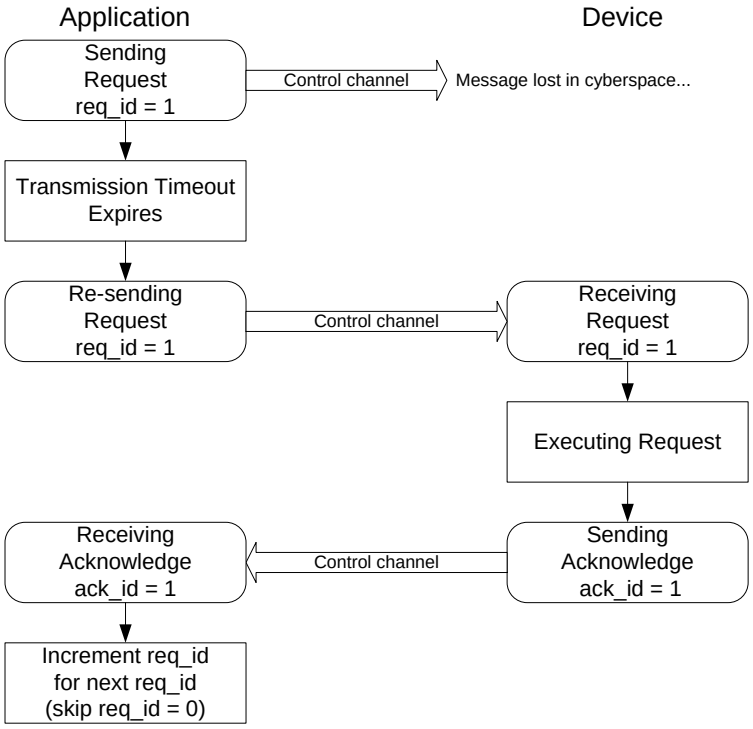


Figure 8-2: Timeout on Request

On the other end, if the device receives a command with the same *req_id* field as the previously received command from this application, it knows the same command has been received twice. If running the command twice makes no difference to the device state, it is allowed to do that. Otherwise the command is only executed once.

[R-051cd] When receiving a command with the same *req_id* from the same application, if running the command twice makes a difference, the device MUST return the corresponding acknowledge message again without executing the command the second time. [R7-13cd]

[R-052cd] In all cases where an acknowledge is requested, the device MUST return an acknowledge for each command it receives (the original and the repeated commands). [R7-14cd]

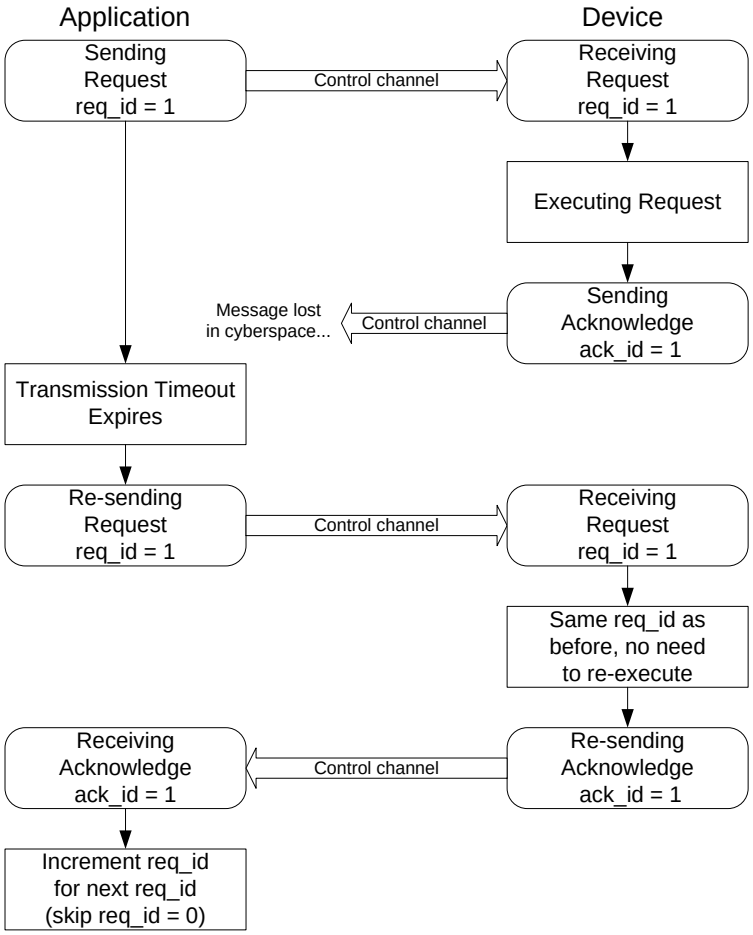


Figure 8-3: Timeout on Acknowledge

Ethernet provides a CRC to guarantee data integrity. If the data does not match the CRC, then the packet is silently discarded. UDP checksums are not required by GVCP, but may be used.

The GVCP header provides an 8-bit field containing the hexadecimal key value 0x42. This enables devices and applications to identify GVCP packets.

The timeout mechanism will ensure retransmission of a corrupted GVCP message. Watchout for Sorcerer's Apprentice Syndrome (explained in RFC1123 – Requirements for Internet Hosts – Application and Support): when the application receives an ACK message that does not match the current ack_id, it must silently discard it.

8.1.4 Flow Control

Flow control ensures the device does not overflow its internal communication buffers when receiving messages. A simple scheme is required by GVCP:

An application sends one packet and then it must wait for the acknowledge message when an acknowledge is requested. It is not permitted to send a second packet before the first acknowledge message has been received unless the transmission timeout expires or no acknowledge had been requested by the application.

In the case that no acknowledge is requested, flow control management is left as a quality of implementation. In this case, the application does not have to wait for acknowledge from the device. It is up to the application to ensure it does not overflow the device.

One possible way to manage flow control is using the IEEE 802.3 PAUSE mechanism. The Network Interface Capability register (at address 0x0010 for interface #0) and the Network Interface Configuration register (at address 0x0014 for interface #0) provide a way to query the device for PAUSE support and enable it.

[CR-053cd] If a device supports the PAUSE mechanism, then the PAUSE configuration MUST persist across device reset. This configuration is visible from the Network Interface Configuration register.

8.1.5 End-to-End Connection

UDP is a connectionless protocol. As such, it does not inherently support a scheme to guarantee the remote device is still connected.

GVCP requires the device to support a heartbeat counter. Any valid command or GVCP packet coming from the primary application resets the heartbeat counter.

~~[R-054ca] An application MUST provide a user programmable heartbeat timeout parameter. [R7-18ca]~~

[R-055ca] If no GVCP message is sent by the application, then the application MUST ensure an artificial GVCP message is sent to the device, so the device's heartbeat counter is reset. [R7-19ca]

Only the primary application has to send heartbeat messages.

The device offers a Heartbeat Timeout bootstrap register. It is recommended for the application to send heartbeat message three times within that device heartbeat timeout period. This ensures at least two heartbeat UDP packets can be lost without the connection being automatically closed by the device because of heartbeat timeout. Depending on the quality of the network connection, it is possible to adjust these numbers.

9 The Channel Concept

In order to handle multiple connections each carrying different types of information, GigE Vision introduces the concept of “channels”.

A channel is a virtual link used to transfer information between GigE Vision entities. GigE Vision supports three types of channels:

1. ***Control channel*** (there is always 1 control channel for the primary application)
2. ***Stream channel*** (from 0 to 512 stream channels)
3. ***Message channel*** (0 or 1 message channel)

Different channels can use the same network interface. In this case, they are each assigned a different UDP port. The following figure represents those channels and highlights the direction of data flow for a basic use case while Figure 9-2 presents an advanced one.

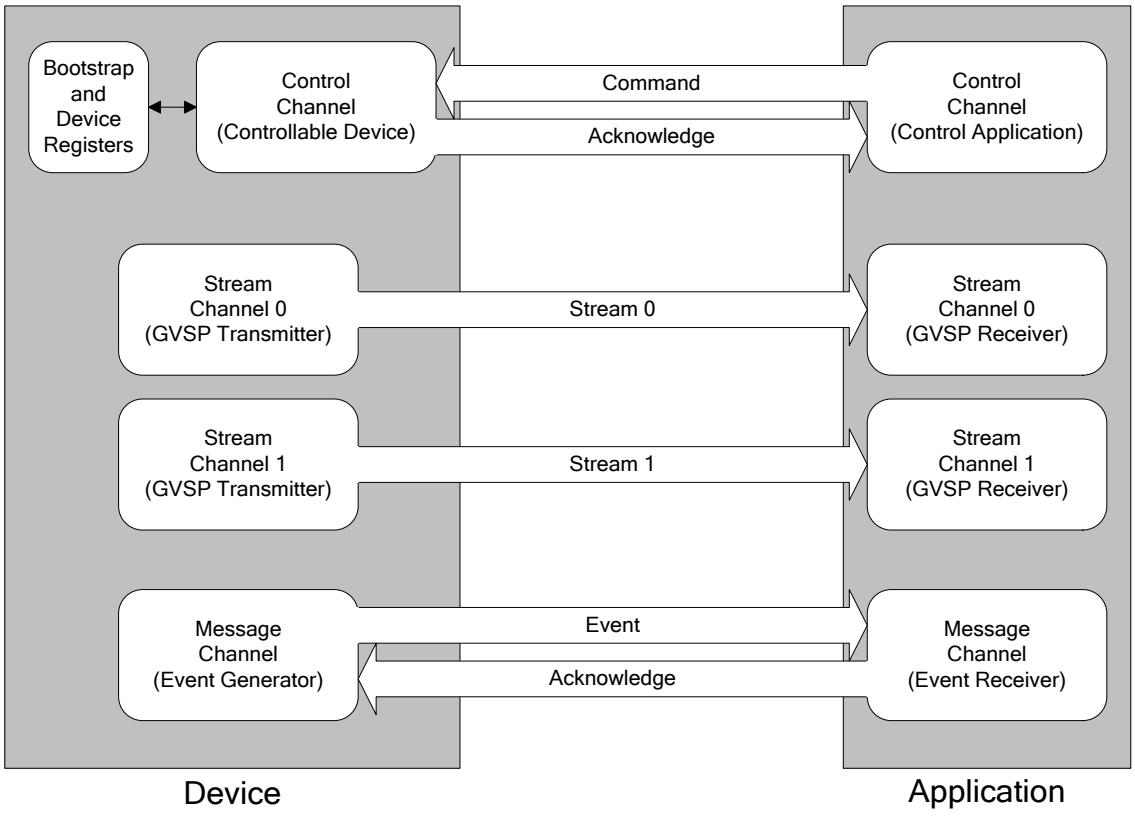


Figure 9-1: Basic Channels Example

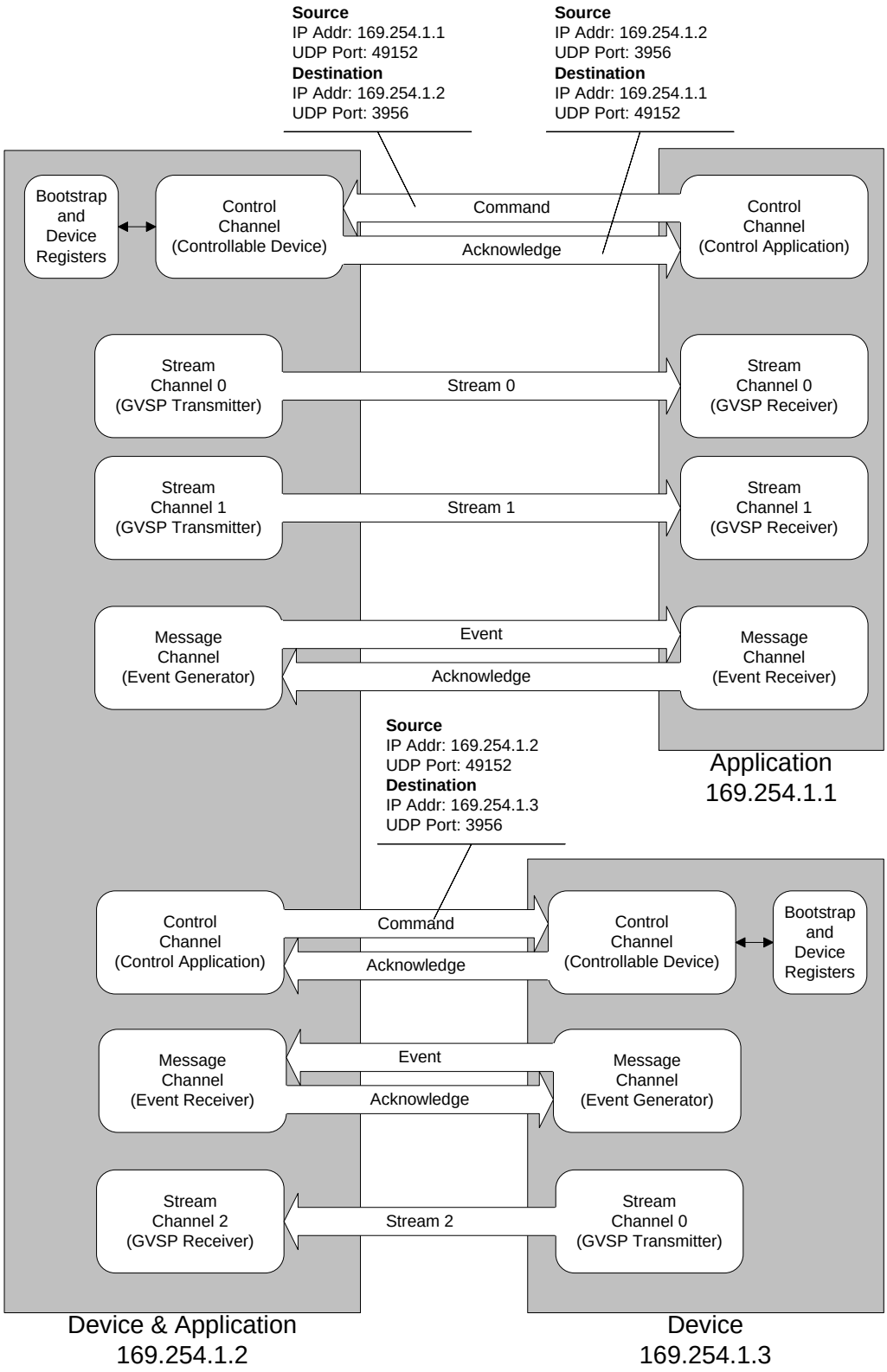


Figure 9-2: Advanced Channels Example

Channels are created dynamically by the application and the device. Except for the standard GVCP port used by the device for the control channel, channels can use any available UDP port.

The application can open or close these channels as needed.

10 Control Channel

A control channel is used by an application to communicate with a device. GVCP defines two types of control channels:

1. ***Primary control channel***: A primary control channel is created by the primary application. The primary application is the one that is allowed to write to the device registers. Only one application is allowed to be the primary application for a device.
2. ***Secondary control channel***: A secondary control channel is created by any secondary applications. Secondary applications can only read from the device registers (they cannot write into them). This can be used for monitoring or debugging. A device may support many secondary applications. A device is allowed to support no secondary application.

Note: Even though only one primary application is supported, it is possible to send/receive image/data streams to/from multiple recipients using UDP broadcast or multicast. But all streams are under the control of the primary application (except for PACKETRESEND requests).

[R-056cd] A device MUST answer device discovery requests from any application after IP configuration is completed, even if the request is not coming from the primary application.

A control channel is required to send commands from the application to the device and to receive the corresponding acknowledge (when one is requested).

[R-057cd] A device MUST support one primary control channel. [R9-1cd]

A device may support none or any number of secondary control channels.

A control channel must be instantiated by the primary application before stream or message channels can be created.

[R-058ca] An application MUST instantiate a control channel to access the device registers. [R8-3ca]

The device bootstrap registers allow the primary application to create the message and stream channels. It is not possible for an application to instantiate message or stream channels if it does not first have a control channel with exclusive or control access (without or with switchover enabled) to the device.

The packet flow sequence on a control channel is:

1. Application sends a command packet
2. Device receives the command packet
3. Device executes the command

4. Device sends the acknowledge packet (if acknowledge is requested)
5. Application receives the acknowledge packet (if acknowledge is requested)

[O-059cd] The device SHOULD send back the acknowledge (if one is requested) as soon as the immediate action requested by the command has been executed. [O9-2cd]

For instance, if the application has requested to write to a register to initiate an image capture, the device should send the acknowledge when the register has been written to (the immediate action), not when the image capture is completed (the indirect reaction of writing into the register).

[R-060ca] An application MUST NOT send a second command before it has received the first acknowledge (the only exceptions being a DISCOVERY_CMD, a PACKETRESEND_CMD, a retry packet when the transmission timeout expires or when no acknowledge is requested). [R9-3ca]

It is recommended that a device executes all command requests and sends back the acknowledge message very efficiently so an application can send the next request.

10.1 Control Channel Privileges

GVCP defines four levels of privileges:

1. ***Exclusive access***: An application with exclusive access is the primary application for this device. It can read or write to the device. No other application can read or write to this device.

[R-061cd] A device MUST support exclusive access. [R9-4cd]

[R-062cd] When an application has exclusive access, the device MUST not allow a secondary application to create a secondary control channel. [R9-5cd]

A device can grant exclusive access if there is no application registered with exclusive access or with control access. If an application has control access with switchover enabled, then the device can grant exclusive access to another application provided that the latter application has the right credentials.

[R-063cd] When a device has granted exclusive access, it MUST return an error (GEV_STATUS_ACCESS_DENIED) in the acknowledge message to any other application sending command message, with the exception of ACTION_CMD, DISCOVERY_CMD, PACKETRESEND_CMD and FORCEIP_CMD. FORCEIP_CMD must respect [R-150cd]. [R9-6cd]

Exclusive access provides a minimal level of protection since other application cannot access the device. Exclusive access is granted by writing into the CCP register. A device has to answer any DISCOVERY_CMD message at all time after successful IP configuration. This is true even if a primary application has exclusive control of the device.

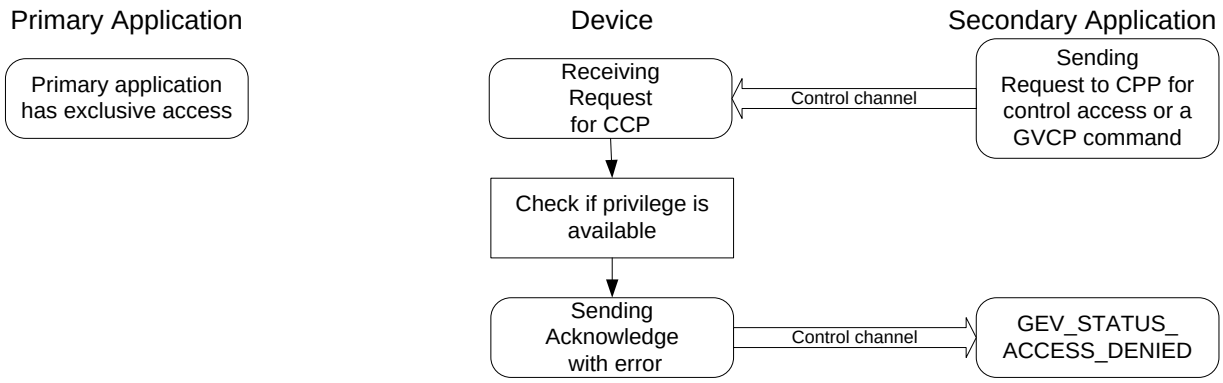


Figure 10-1: Exclusive Access

2. ***Control access***: An application with control access is the primary application for the device. It can read or write to the device. But other applications are allowed to read from the device. Note that a device IS NOT required to support control access.

[CO-064cd] When a device supports control access and when an application has control access, the device SHOULD allow a secondary application to create a control channel supporting the READREG and READMEM messages. [CO9-7cd]

[CR-065cd] When a device supports control access, it can grant control access if there is no application registered with exclusive access or with control access. If an application has control access with switchover enabled, then the device can grant access to another application if the latter has the right credentials. An application has the right credentials if the *control_switchover_key* field of the CCP register is set to the value held by the Control Switchover Key bootstrap register when access is requested by this application. Otherwise, it MUST return an error (GEV_STATUS_ACCESS_DENIED). [CR9-8cd]

Control access is granted by writing into the CCP register.

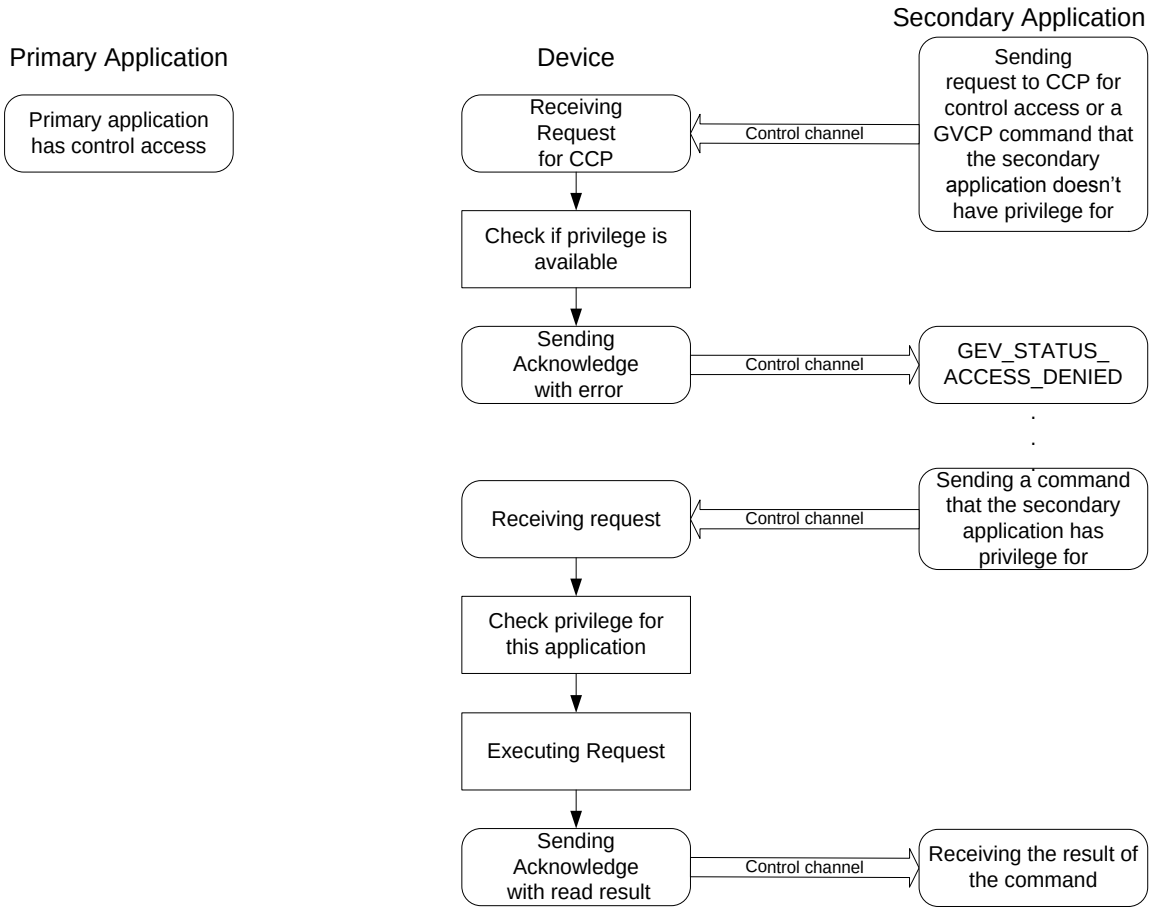


Figure 10-2: Control Access

3. ***Control access with switchover enabled***: Just like in the control access mode, an application with control access with switchover enabled is the primary application for the device. It can read or write to the device. Other applications are allowed to read from the device. However, this mode enables another application to take control over the device provided that it has the right credentials. Note that a device IS NOT required to support control access with switchover enabled.

[CO-066cd] When a device supports control access with switchover enabled and when an application has control access with switchover enabled, the device SHOULD allow a secondary application to create a control channel supporting the READREG and READMEM messages. [CO9-17cd]

[CR-067cd] When a device supports control access with switchover enabled, it can grant control access if there is no application registered with exclusive access or with control access. If an application has control access with switchover enabled, then the device can grant access to another application if the latter has the right credentials. An application has the right credentials if the *control_switchover_key* field of the CCP register is set to the value held by the Control Switchover Key bootstrap register when access is requested by this application. Otherwise, it MUST return an error (GEV_STATUS_ACCESS_DENIED). [CR9-18cd]

Control access with switchover enabled is granted by writing into the CCP register.

4. ***Monitor access***: An application with monitor access has no particular privilege over the device. It is a secondary application that can only read from the device when no application with exclusive control access is linked with the device. A secondary application is typically used to help debugging. For instance, it can be used to dump the registers of the device. A device can allow monitor access only if there is no application with exclusive access. If this privilege is available, the secondary application can directly send a READREG_CMD or READMEM_CMD message to the device. An application does not write to the CCP register to get monitor access. Note that a device is not required to support monitor access. A device is not required to track the *req_id* of a secondary application: each command from a secondary application, even a retried command, gets executed.

It is perfectly legal for a device to only support exclusive access. In this case, only the primary application can access this device.

[R-068cd] The device MUST memorize the context associated with the primary application. This context is at least composed of:

1. The IP address of the application
2. The source UDP port for the application
3. The granted privilege (exclusive, control access or control access with switchover enabled) [R9-9cd]

The device checks this context to determine if it can grant privilege to other applications and if received command messages are valid.

Once a device has been discovered, an application can take command of it by opening a primary control channel. This ensures only a single application can control the device.

Note: A primary control channel is not a security scheme. It only ensures one application is bound to control the device.

Control channels are created using dynamic port number (chosen arbitrarily) on the application side and the standard GVCP port on the device end (unless a different GVCP port number is advertised through mDNS).

The application must select a port number for the life period of the channel. The application must also indicate if it wants to take exclusive control of the device or allow other application to monitor the device. If supported by the device, the application can also enable another application to take control over the device provided that the later application has the right credentials.

Note: Having non-exclusive access to the device may be useful during application development to allow other debugging tools monitor the device.

Note: Enabling another application to take control over a device may be required in systems were redundancy and fault recovery is necessary. Please refer to Section 14.4 for details.

By using the standard GVCP port for the channel on the device side, the connection procedure is made simpler through the GVCP DISCOVERY command. But mDNS can be used to advertise a different port number for GVCP.

10.2 Control Channel Registers

The control channel provides the following registers:

1. Control Channel Privilege register (CCP)
2. Control Switchover Key register
3. Heartbeat Timeout register
4. Pending Timeout register

Refer to the bootstrap registers section for a complete description of the fields in these registers.

Note: The list above is not an exhaustive list of all the registers associated with the control channel. It lists the main registers related to the concepts presented in this section.

10.3 Opening a Control Channel

An application opens a control channel to take exclusive or control access (without or with switchover enabled). There is no need to open a control channel for monitor access.

To open a channel, an application writes the requested privileges to the CCP register and checks for status returned by the acknowledge message of device. If successful, application now has the privilege and the device returns GEV_STATUS_SUCCESS. Otherwise, device returns status GEV_STATUS_ACCESS_DENIED. The application knows from the status code if it has been granted the requested privilege.

When the device receives a command to write to the CCP register it tries to create a channel context if the application is asking for exclusive or control access (without or with switchover enabled). Any secondary application does not have to write to the CCP register: it can directly read from the device. The device verifies whether it can give the desired privilege and it updates the context if necessary.

[R-069cd] A primary application is allowed to ask for the same privilege without closing the control channel. In this case, the device MUST accept the request and return GEV_STATUS_SUCCESS. [R9-10cd]

The primary application is allowed to directly switch to another control privilege when writing the CCP bootstrap register.

[R-070cd] A device MUST allow an application to directly switch between Exclusive, Control and Control with Switchover Enabled (when supported) privilege without having to first disconnect the control channel. [R9-11cd]

Another application that has the right credentials can request, and get granted, device control when the current primary application has been granted control access with switchover enabled. Control access with switchover enabled is detailed in Section 14.4.

10.4 Closing a Control Channel

A primary application closes its opened control channel by writing 0 to the CCP register. Another application cannot write into the CCP register unless the primary application has been granted control access with switchover enabled. In this case, the CCP request can be granted if the other application has the right credentials. Otherwise, it gets a GEV_STATUS_ACCESS_DENIED from the device.

[R-071ca] The primary application MUST write 0 into CCP to release its privilege. [R27-30ca]

[R-072cd] When the primary control channel is closed, the device MUST stop streaming, unless it has been configured otherwise via the unconditional streaming enabled bit of the Stream Channel Configuration registers, reset all connection states (including message and GVSP transmitter stream channels) by at least clearing the MCP register and the SCPx registers of the GVSP transmitters (if applicable), resetting the Primary Application Port and Primary Application IP Address registers (if supported) and make itself available for another connection upon control channel closure. Device's registers (including bootstrap registers) MUST represent the new state after the operation is completed. [R9-12cd]

Note: When closing a primary control channel, the application might leave the device with an incoherent set of registers. It is always up to the primary application to ensure it provides a complete coherent set of registers when it takes exclusive or control access of a device (without or with switchover enabled). For instance, a device might have a default pixel size of 10 bits. The first primary application might set pixel size to 8 bits (non-default value) then close the control channel leaving 8 bits programmed into the device. A second application might then program the device for acquisition, but leave the pixel size register unchanged, assuming the device still uses its default value. Obviously the set of device registers does not match what the second application expects.

Note: Resetting the GVSP receiver stream channels when the control channel is closed is left as an implementation detail on devices including GVSP receivers but it is recommended to leave them open. This is because a simple receiver product may want to keep displaying incoming video feeds even if a primary control channel is not open on the device.

Note: In some video distribution systems over Ethernet, it is desirable for video sources to keep transmitting video even if the primary application crashes or is closed. This specification enables GVSP transmitters to continue streaming independently of the state of the control channel or of any ICMP messages received by their associated device (such as destination unreachable messages). When this feature is supported by GVSP transmitters, it is enabled with configuration bits. Please refer to Section 11.10 for more details on the topic.

10.5 Control Channel Heartbeat

Because UDP is a connectionless protocol, the GVCP must implement a recovery mechanism when one of the two participants is abruptly detached without a proper channel close procedure. This is achieved using a heartbeat sequence. The application must periodically run this sequence if there is no activity on the control channel. The rate at which the sequence must be run is programmable in the device.

[O-073ca] The application SHOULD provide a parameter to configure the heartbeat rate. [O9-13ca]

[R-074cd] The device MUST provide a bootstrap register to control the heartbeat timeout. [R9-14cd]

[R-075cd] Any valid GVCP command from the primary application MUST reset the heartbeat counter on the corresponding device, with the exception of the PACKETRESEND_CMD, the DISCOVERY_CMD, the ACTION_CMD and the FORCEIP_CMD. FORCEIP_CMD must respect [R-150cd]. [R9-15cd]

Note: ACTION_CMD can be coming from a different UDP port than the Control Channel. Hence it cannot reset the heartbeat.

It is recommended for the application to read the CCP register as the method to reset heartbeat counter.

Note: The heartbeat counter can only be reset by the primary application. Read access by secondary application have no effect on the heartbeat counter.

[R-076cd] If the device does not receive a control message for more than the user-programmable value in bootstrap registers (three seconds is the default value) and the heartbeat capability has not been disabled (bit 31 of the GVCP Configuration register) on the device, then it MUST close the control channel as described in the previous section. [R9-16cd]

The application could detect a device malfunction when it cannot read the CCP register or if it reads an unexpected value. If the application reads an unexpected value, it assumes the connection has been lost. The application must then establish new connection with the device by instantiating the control channel.

10.6 Controlling the Device

Once an application has opened a primary control channel with a device, it can send any command supported by GVCP. Refer to the dictionary sections of this document (Control Channel Dictionary and Message Channel Dictionary).

Secondary applications can send READREG and READMEM commands to read device state.

DISCOVERY command may be sent at any time by any application and be correctly answered by the device. A device has to always answer DISCOVERY commands from any application after it has completed its IP configuration cycle.

If the device and the application don't reside on the same subnet, then the device can broadcast the DISCOVERY_ACK message. Otherwise, it can use unicast transfer to the source of the discovery request.

ACTION and PACKETRESEND commands can also be sent at any time by any application and be processed properly by the device.

Finally, a FORCEIP command received from a secondary application must be handled according to [R-150cd].

The following table provides a summary of the characteristics of the GVCP commands. This table is informative only. Please refer to the standard text for the actual requirements and objectives that each of these commands has to support.

Table 10-1 : GVCP Commands Characteristics

	Mandatory	*Application Type*	*Unicast/ Broadcast of CMD*	*Unicast/ Broadcast of ACK*	*Reset Heartbeat*	*can generate PENDING _ACK*
DISCOVERY	Yes	Primary and secondary	Unicast or broadcast	Unicast or broadcast	No	No
FORCEIP	Yes	Secondary	Broadcast	Unicast or broadcast	No	No
READREG	Yes	Primary and secondary	Unicast preferred, broadcast possible	Unicast	Yes	Yes
WRITEREG	Yes	Primary	Unicast preferred, broadcast not recommended	Unicast	Yes	Yes
READMEM	Yes	Primary and secondary	Unicast preferred, broadcast possible	Unicast	Yes	Yes
WRITEMEM	Yes	Primary	Unicast preferred, broadcast not recommended	Unicast	Yes	Yes
PACKETRESEND	No	Primary and secondary	Unicast	N/A. Sent on stream channel	No	No
PENDING	No	Primary and secondary	N/A	Unicast	N/A	N/A
ACTION	No	Primary and secondary	Unicast or broadcast	Unicast	No	Yes

10.7 Use of Pending Acknowledge

GVCP provides a very simple handshake between the device and the application through the acknowledge message. But sometimes, command execution can take longer to complete than what is expected by the application. It is thus useful to have a mechanism to communicate:

1) The maximum amount of time to execute a typical request;
2) Have a way to notify the application when request execution time is going to exceed this value so the application can correctly wait the extra amount of time necessary to complete the request.

Version 1.0 of this specification did not provide any mechanism for the device to indicate to the application that the request execution time will exceed the timeout. Therefore, when a request takes too long to execute, the application is likely to consider the packet was lost and try to retransmit it using the same *req_id*, resulting in inefficient handling of the command.

Version 1.1 of this specification introduced a special ACK message called PENDING_ACK that is sent by the device to tell the application the current command execution time is going to take longer than the value reported by the Pending Timeout bootstrap register.

[O-059cd] provides a guideline to quickly complete the execution of the immediate action. In this situation, no PENDING_ACK needs to be issued. The following diagram illustrates this scenario.

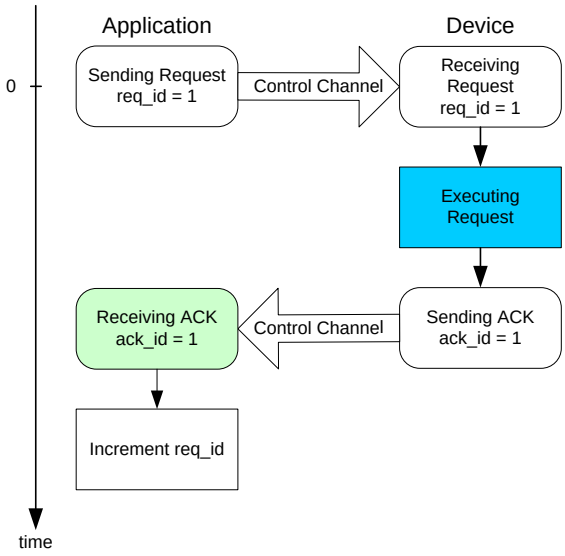


Figure 10-3: No PENDING_ACK

The following diagram presents an example of the retransmission of a command as a result of the expiration of the application ACK timeout before the end of the execution of the command on the device.

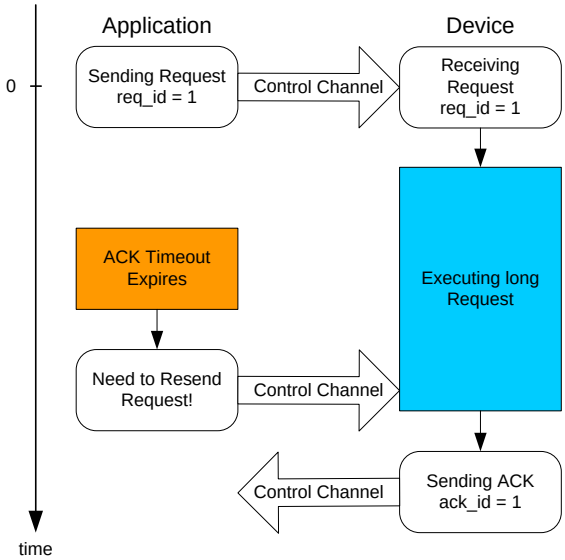


Figure 10-4: Request Retransmission (PENDING_ACK not Enabled)

Note that in a situation like this, it is very plausible the ACK of the initial request will be interpreted by the application as the ACK for the retransmission. Typically, a device will not re-execute a request with the same *req_id*. Nevertheless, the device might be re-executing the long command a second time (see [R-051cd] and the paragraph before). This might lead to another timeout on the next request coming from the application! Moreover, as per [R-052cd], a device will return 2 ACKs: one for the initial command and one for the retransmitted one.

Starting with the previous timeout scenario, if the application enables the PENDING_ACK, the device has a way to notify the application that a command may take longer to execute. The device returns a PENDING_ACK to reset the application ACK timeout (to 2000 ms in the following example). Therefore, the application has a clear indication of the amount of time required to complete the original request (if it doesn't like it, application can decide to timeout anyway!). Note the ack_id of PENDING_ACK must stay the same as for the initial request. This clearly indicates the PENDING_ACK belongs to this request.

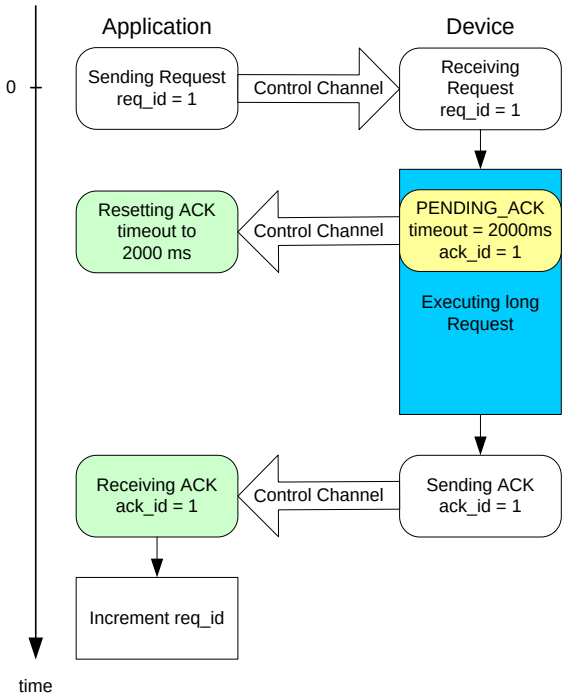


Figure 10-5: Device using PENDING_ACK

An application receiving a PENDING_ACK has to wait for the indicated timeout to complete or for the actual request ACK to arrive before sending the next command. Also note that if the PENDING_ACK message is lost, then the Application will behave in same way as with version 1.0 of this specification.

[CR-077cd] If a device supports PENDING_ACK, then it MUST answer DISCOVERY_CMD message between the time a PENDING_ACK is issued and the command ACK is sent.

[CR-078cd] If a device supports PENDING_ACK, then the device cannot use PENDING_ACK to answer a DISCOVERY_CMD.

Without the previous two conditional requirements, other applications on the network might not correctly enumerate devices on the network.

Note: How a device should react when it receives a GVCP command (other than DISCOVERY_CMD) after a PENDING_ACK has been issued, but before the final acknowledge is sent, is left as a quality of implementation.

For instance, the device may have the capacity to process requests from different applications simultaneously. Alternatively, it could respond with a PENDING_ACK to the new request with a time to completion value equal to the remaining time of the previous command plus the time required to execute the new one (even if the new command would not normally return PENDING_ACK). This would indicate that the new command was queued for processing and should not be re-issued. Another option would be for the device to respond with a GEV_STATUS_BUSY status and let the application resend the command at a later time. However, a device can simply queue the new command as any other packets which would increase the likelihood for this command to timeout since the application issuing the new request would not have received the PENDING_ACK of the previous command.

If a device responds with a PENDING_ACK to a request from a secondary application, then it is assumed that the primary and all secondary applications of the system can support PENDING_ACK. However, how a primary application can determine if all the secondary applications of a system can support PENDING_ACK is beyond the scope of this specification and is left as system level consideration.

11 Stream Channel

A stream channel enables data transfer from a GVSP transmitter to a GVSP receiver. This transfer uses the GigE Vision Streaming Protocol (GVSP).

[R-079s] A product supporting GVSP streaming MUST offer at least one and at most 512 stream channels. [R8-1s]

Note: A product is not required to support streaming. In this case, it doesn't include any stream channel.

[R-080s] A product supporting GVSP streaming MUST support stream channel sequentially starting from index 0. It is not allowed to leave a gap in the index of supported stream channels. [R10-2s]

The stream can transport any payload type defined by this specification. Each stream channel can be a transmitter (data source) or a receiver (data sink) but cannot be both. In other words, the direction of the stream channel cannot be programmed. Bootstrap register bits capture the direction of each stream channel.

Note: In order to promote compatibility with applications compliant with the previous versions of this specification, it is recommended, for transceiver devices, that stream channels transmitting data are defined in the register map before receiving stream channels. In other words, stream channels transmitting data use a lower index in the bootstrap registers list than the receiving stream channels.

11.1 Stream Channel Registers

A given stream channel can be a transmitter or a receiver. The following registers are associated to transmitter and receiver stream channels:

1. Stream Channel Port register (SCPx)
2. Stream Channel Packet Size register (SCPSx)
3. Stream Channel Destination Address register (SCDAx)
4. Stream Channel Configuration register (SCCFGx)

In addition to this, the following registers are associated to transmitter stream channels.

5. Stream Channel Packet Delay register (SCPDx)
6. Stream Channel Source Port register (SCSPx)

Note: While the Stream Channel Packet Size register (SCPSx) for transmitters is used to configure the size of the packet to use when streaming data, it is used to report the maximum packet size supported by a receiver.

Refer to the bootstrap registers section for a complete description of the fields in these registers.

11.2 Tagging Data Block

Information passed on a stream channel is divided into blocks. For instance, a GigE Vision camera fits each captured image into a data block. A GVSP receiver would thus be able to track images by looking at the block ID associated with each data block. This block ID could be 16-bit (*block_id*, introduced by GigE Vision version 1.0) or 64-bit (*block_id64*, introduced by GigE Vision version 2.0) wide.

[R-081s] When 64-bit *block_id64* are used, then 32-bit *packet_id32* MUST be used.

[R-082s] When 16-bit *block_id* are used, then 24-bit *packet_id* MUST be used.

[CR-083s] If a GVSP transmitter or GVSP receiver is certified to be compliant to GigE Vision version 2.0, then it MUST support 64-bit *block_id64* and 32-bit *packet_id32*.

[CO-084s] If a GVSP transmitter or GVSP receiver is certified to be compliant to GigE Vision version 2.0, then it SHOULD support 16-bit *block_id* and 24-bit *packet_id*.

[CO-085st] If a GVSP transmitter supports 16-bit *block_id* and 24-bit *packet_id*, then it SHOULD have 16-bit *block_id* and 24-bit *packet_id* enabled as its factory settings.

Note: A device that does not have 16-bit *block_id*/24-bit *packet_id* enabled as its factory settings cannot advertise to be a GigE Vision 1.x device as it is not directly interoperable with a GigE Vision 1.x application out of the box.

The introduction of 64-bit *block_id64* and 32-bit *packet_id32* in GigE Vision 2.0 generates a number of backward compatibility challenges that must be clearly understood to ease interoperability.

A GigE Vision 2.0 compliant GVSP transmitter or receiver can either be:

1. **pure GigE Vision 2.0** if it only supports 64-bit *block_id64* and 32-bit *packet_id32*.
2. **bi-mode GigE Vision 2.0** if it supports both 16-bit *block_id* and 64-bit *block_id64* (and 24-bit *packet_id*/32-bit *packet_id32*).

A bi-mode GigE Vision 2.0 product does not suffer from any interoperability problem. But a pure GigE Vision 2.0 transmitter or receiver cannot work with the corresponding GigE Vision 1.x receiver or transmitter.

The following table lists the 9 possible scenarios and interoperability conclusions.

Table 11-1 : Interoperability Analysis for Block ID Size

GVSP Transmitter block ID Support	GVSP Receiver block ID Support	Explanation
16-bit only	16-bit only	GVSP Transmitter is GigE Vision 1.x GVSP Receiver is GigE Vision 1.x *Conclusion*: no interoperability issue
16-bit only	64-bit only	GVSP transmitter is GigE Vision 1.x GVSP receiver is a pure GigE Vision 2.0 *Conclusion*: since the GVSP receiver does not offer support for 16-bit *block_id*, this pair **cannot** work together.
16-bit only	16-bit and 64-bit	GVSP transmitter is GigE Vision 1.x GVSP receiver is a bi-mode GigE Vision 2.0 *Conclusion*: since the receiver is bi-mode, it can accommodate 16-bit *block_id*.
64-bit only	16-bit only	GVSP transmitter is pure GigE Vision 2.0 GVSP Receiver is GigE Vision 1.x *Conclusion*: since the GVSP transmitter does not offer support for 16-bit *block_id*, this pair **cannot** work together.
64-bit only	64-bit only	GVSP transmitter is pure GigE Vision 2.0 GVSP receiver is a pure GigE Vision 2.0 *Conclusion*: no interoperability issue.
64-bit only	16-bit and 64-bit	GVSP transmitter is pure GigE Vision 2.0 GVSP receiver is a bi-mode GigE Vision 2.0 *Conclusion*: this pair can operate with 64-bit *block_id64*.
16-bit and 64-bit	16-bit only	GVSP transmitter is a bi-mode GigE Vision 2.0 GVSP Receiver is GigE Vision 1.x *Conclusion*: this pair can operate with 16-bit *block_id*.
16-bit and 64-bit	64-bit only	GVSP transmitter is a bi-mode GigE Vision 2.0 GVSP receiver is a pure GigE Vision 2.0 *Conclusion*: this pair can operate with 64-bit *block_id64*.
16-bit and 64-bit	16-bit and 64-bit	GVSP transmitter is a bi-mode GigE Vision 2.0 GVSP receiver is a bi-mode GigE Vision 2.0 *Conclusion*: no interoperability issue.

To facilitate interoperability, it is recommended for GVSP receivers to support 16-bit *block_id* and 64-bit *block_id64*.

11.3 Opening a Stream Channel

A primary application opens a stream channel by writing the host port to the SCPx register and the destination IP address to the SCDAx register. Only the primary application is allowed to configure stream channels.

[O-086st] A GVSP transmitter SHOULD use any dynamic port number as the UDP source port for a given stream channel. [O10-3st]

The IANA has established that dynamic ports (also known as ephemeral ports) are those from 49152 through 65535.

[R-087ca] The stream channel MUST be activated by the application by writing the *host_port* field of the SCPx register with a value different from 0. Otherwise, the channel is closed. [R27-36ca]

[R-088st] The *block_id/block_id64* representing the index of the current data block in the stream MUST be reset to 1 when the stream channel is opened.

The above ensures that if a primary application connects to a device, the first data block transmitted will necessarily have a *block_id/block_id64* of 1 after the stream channel is configured and opened.

11.4 Operation of the Stream Channel

Once a stream channel is opened, it must respect the following rules:

[CR-089cd] If supported by the device, SCSPx MUST return a non-zero value corresponding to the source UDP port of the GVSP transmitter stream channel when SCPx is non-zero. It is valid for it to be non-zero at any point in time prior as well as long as it then remains constant for the duration of the control session. [CR27-45cd]

[CR-090cd] If supported by the device, when SCPx is non-zero the device MUST silently ignore any UDP traffic coming from the address and port combination listed in SCDAx and SCPx (if stream channel has been opened) targeted at the device SCSPx port. [CR27-46cd]

11.5 Closing a Stream Channel

[R-091ca] A primary application MUST close a stream channel by writing a 0 to the SCPx register. [R10-4ca]

Since SCPx is used to open and close the stream channel, it has to be the last stream channel register accessed by the application upon opening the channel and the first register accessed when closing the stream.

A GVSP transmitter should stop streaming data as soon as possible after the stream channel is closed. When closing a stream channel, it is possible that the last block of data is not totally sent on the link. But any packet has to be sent in its entirety. It is up to the GVSP receivers to correctly handle this situation.

[R-092st] A GVSP transmitter MUST NOT send an incomplete streaming packet. [R10-5st]

The proper way to stop the stream on a nice block boundary is by using manufacturer-specific registers. For instance, a camera offers AcquisitionStart and AcquisitionStop features.

A GVSP transmitter is not required to send the data trailer when the stream channel is closed during transmission of a given *block_id/block_id64* packets. It behaves like an abort where the current packet is the last one sent.

11.6 Packet Size

GVSP does not impose any restrictions on packet size. It is up to the management entity controlling the global system and GVSP transmitters to negotiate a suitable packet size. One way to determine the maximum packet size that does not lead to IP fragmentation is by sending stream test packets (the *do_not_fragment* bit should be set by the application for test packets).

This can be achieved through the SCPSx bootstrap register of GVSP transmitters: bit 16 to 31 provides the packet size while bit 0 indicates to fire a test packet. Bit 1 must be set to indicate IP fragmentation is not allowed by the application. All these bits are written at once to SCPSx. This results in the GVSP transmitter sending a single test packet with the total size (including IP and UDP headers) equal to the requested maximum packet size.

To discover the optimal packet size, the management entity sets up the stream channel by writing into SCPx, SCPDx and SCDAx registers. It can then perform a simple iterative binary search (or any other search) by writing into the SCPSx register with various packet sizes. If the test packet does not reach the GVSP receivers, then the management entity can reduce the packet size. If the test packet reaches the GVSP receivers, then the management entity can increase the packet size. This way, the management entity can converge to the optimal packet size. In most scenarios, the larger the packet size, the lower the overhead of packet transmission.

Note: The management entity of the system can easily determine if the test packet reached the destination in a point-to-point configuration when the management entity is the application and the GVSP receiver. However, it could be more complicated for the management entity to determine if the test packet reached its destination(s) in other situations. It is beyond the scope of this specification to define how a management entity determines that a test packet is received by a GVSP receiver. It is left as a quality of implementation.

[R-093st] If the packet size requested in SCPSx is not supported by the GVSP transmitter, then it MUST NOT send a test packet when it is requested to do so. [R10-6st]

The management entity needs to find a packet size supported by the GVSP transmitter and by all network elements between the GVSP transmitter and the GVSP receivers.

Note: This is not a bullet-proof method of finding the maximum packet size when multiple routes are available from the GVSP transmitter to the GVSP receivers. Different routes might very well support different MTU. One cannot guarantee all data packets will travel using the same route as the test packet.

11.7 Multicasting

A stream channel might use multicasting if the data stream must be sent to multiple locations. Multicasting is activated by setting the destination IP address of the GVSP transmitter to a multicast group in the SCDAx register. When multicasting, any destination is allowed to send a PACKETRESEND command on the control channel.

Note: When multicasting, the destination IP address might range from 239.0.0.0 to 239.255.255.255 for Administratively-scoped (local) multicast addresses and from 224.0.1.0 to 238.255.255.255 for Globally-scoped (Internet-wide) multicast addresses. Refer to RFC3171 (IANA Guidelines for IPv4 Multicast Address Assignments) for more information.

11.8 Impact of Multiple Network Interfaces

GVSP allows multiple network interfaces for stream channels to increase the bandwidth available for streaming. Refer to section 3 for additional information about multiple interface devices.

It is up to the primary application to correctly handle the network interfaces available on the device using the bootstrap registers.

Bootstrap registers might optionally be available to report the current link speed of each network interface.

11.9 Traversing Firewalls or Network Address Translation Devices

Due to the unidirectional nature of GVSP traffic, it is common for the streaming channel data to be blocked by firewalls or other types of network since the incoming traffic cannot be matched against an outgoing packet. To address this, the SCSPx registers are designed to allow the application associated to a GVSP receiver to create a simulated bidirectional traffic conversation between the GVSP receiver and the GVSP transmitter streaming channel by informing the application of the remote UDP port. This information would otherwise be unknown until the first streaming packet arrives, which may never happen if it was blocked. Additionally, this information may be used in other ways to help configure software and/or networking equipment.

The GVSP Capability register (at address 0x092C) can be used to query if SCSPx registers are supported by this device.

[CR-094st] If a product supports SCSPx for one GVSP transmitter, then it MUST support it for all its GVCP transmitter stream channels. [CR10-8st]

The suggested usage of SCSPx is to query it after opening the image stream via SCDAx and SCPx but before starting the camera's flow of image data. If desired, the application can query it earlier and cache it off (only if it is non-zero) so that it does not need to query it each time the streaming channel is opened. Refer to [CR-089cd] for proper behavior of SCSPx.

Next, the application can send a UDP packet from the same source port it programmed into SCPx to the port specified by SCSPx. The content of this packet is ignored, but the suggested payload should be smaller than the maximum size of a GVCP command packet. This operation should be realized on all the open stream channels.

At this point the application can continue starting the acquisition on the camera. Depending on the configuration of the firewall or other impediment, the application may choose to continue periodically sending packets similar to the first one for the duration of the streaming session. The suggested interval is 30 seconds and could be configurable. Different firewalls might require adjustment to this interval.

[CR-095ca] When SCSPx is supported by the application, the application MUST not send packets to the port specified by SCSPx unless it has the stream channel opened and successfully reads a non-zero value back from SCSPx. [CR10-9ca]

The application should not expect the device to answer the packet it sends to the GVSP transmitter stream channel. But this mechanism provides a way to open-up the UDP port in the firewall. The implementation is left as a quality of implementation and is likely to vary with different firewall types and operating systems.

Note that the same mechanism is also available for the message channel.

11.10 Unconditional Streaming

In a number of video distribution systems over Ethernet, especially the ones involving simultaneous GVSP receivers, it is often mandatory to make sure that GVSP transmitters can continue streaming no matter what happens with their primary application or with the state of the network. For instance, it might be required that a device continues streaming even if its primary application crashes or is closed and moved to a different host computer.

Starting with version 1.2, the specification enables such scenarios by allowing GVSP transmitters, supporting this capability, to be configured to continue streaming independently of the state of the control channel or any ICMP messages received by the device associated to a GVSP transmitter.

It is then possible for an application to configure GVSP transmitters for unconditional streaming by setting the *unconditional_streaming_enabled* bit in the Stream Channel Configuration registers (SCCFGx) provided that the capability is supported by the GVSP transmitters in question. When configured in this mode, a GVSP transmitter will not stop if the control channel of the device is closed. Moreover, it will not stop streaming if its associated device receives ICMP destination unreachable messages. In other words, the GVSP transmitter continues streaming as long as dictated by its primary application. The Stream Channel Capability registers (SCCx) indicate if GVSP transmitters can support unconditional streaming.

12 Message Channel

A message channel allows a device to send asynchronous messages to the application. For instance, a device may want to signal that a camera trigger has been detected.

[R-096cd] A device MUST offer either one or no message channel (event generator). [R8-2cd]

A message channel is very similar to a control channel, but requests are emitted in the opposite direction. The device always initiates transactions on the message channel. Therefore, the message channel headers are identical to the control channel headers. Support of message channel by a device is optional.

The packet flow sequence on a message channel is:

1. Device sends a message packet
2. Application receives the message packet
3. Application process the message
4. Application sends the acknowledge packet (if acknowledge is requested)
5. Device receives the acknowledge packet (if acknowledge is requested)

[CR-097cd] When a device supports a message channel, the *req_id* field on the message channel MUST increment from one message to the next (except for retransmission). When *req_id* wraps-around to 0, then its value must be set to 1 (0 is invalid for *req_id*). Therefore, *req_id* 65535 gets incremented to 1. [CR11-1cd]

This allows the application to detect if a UDP message has been lost, even when no acknowledge is requested.

12.1 Message Channel Registers

When supported, the message channel provides the following registers:

- Message Channel Port register (MCP)
- Message Channel Destination Address register (MCDA)
- Message Channel Transmission Timeout register (MCTT)
- Message Channel Retry Count register (MCRC)
- Message Channel Source Port register (MCSP)

Refer to the bootstrap registers section for a complete description of the fields in these registers.

12.2 Opening the Message Channel

A primary application opens the message channel by writing the destination IP address to the MCDA register and then the host port to the MCP register. Only the primary application is allowed to open the message channel since it must have write access privilege.

[CO-098cd] When a device supports a message channel, it SHOULD use any dynamic port number as the UDP source port for the message channel. [CO11-2cd]

[CR-099ca] When supported, an application MUST activate the message channel by writing the *host_port* field of the MCP register to a value different from 0. Otherwise, the channel is closed. [CR27-31ca]

[CR-100ca] When an application supports the message channel, it MUST acknowledge messages on the message channel using the UDP source port of the incoming packet as the UDP destination port for the acknowledge packet when such an acknowledge is requested. [CR11-7ca]

12.3 Operation of the Message Channel

Once a message channel is opened, it must respect the following rules:

[CR-101cd] When a message channel is supported, if the timeout expires, then the device MUST try to resend the same message. The number of retries is indicated by MCRC register. [CR27-33cd]

[CR-102cd] When a message channel is supported and when MCRC is set to 0, then the device MUST not retransmit a message. [CR27-35cd]

[CR-103cd] When a message channel is supported and when MCTT is different from 0, then the ACKNOWLEDGE bit of the command header MUST be set. Acknowledge generation is disabled by setting MCTT to 0. In this case, ACKNOWLEDGE bit of the command header MUST be cleared by the device. [CR27-34cd]

[CR-104cd] If supported by the device, MCSP MUST return a non-zero value corresponding to the source UDP port of the message channel when MCP is non-zero. It is valid for it to be non-zero at any point in time prior as well as long as it then remains constant for the duration of the control session. [CR27-43cd]

[CR-105cd] If supported by the device, when MCSP is non-zero the device MUST silently ignore any UDP traffic coming from the address and port combination listed in MCDA and MCP (if message channel has been opened) targeted at the device MCSP port if they don't match an EVENT_ACK or EVENTDATA_ACK. [CR27-44cd]

12.4 Closing the Message Channel

[CR-106ca] When a message channel is supported, a primary application MUST close it by writing a 0 to the MCP register. [CR11-3ca]

Since MCP is used to open and close the message channel, it has to be the last message channel register accessed by the application upon opening the channel and the first register accessed when closing the channel.

When closing a message channel, it is not allowed for a message packet to be sent incomplete.

[CR-107cd] When a message channel is supported, if a packet is being transmitted by the device while its message channel is being closed, then this message MUST be completely transmitted. [CR11-4cd]

[CR-108ca] When a message channel is supported, if a message is received by the application when its message channel is closed, then it MUST be silently discarded. [CR11-5ca]

12.5 Asynchronous Events

A device can send asynchronous event messages on the message channel when the channel is opened. Each event is represented by a 16-bit ID. This document defines two categories of events:

- GigE Vision standard events (value 0 to 36863)
- Device-specific events (value 36864 to 65535)

Manufacturer-specific registers are used to enable/disable those events. XML device description files describe the events, its index and the register to enable/disable. Note that even the standard events are enabled/disabled using manufacturer-specific registers.

12.6 Multicasting

A message channel might use multicasting if the events must be sent to multiple locations.

[CR-109cd] When a message channel is supported, in the case of event message multicast, acknowledge packets are not permitted (the acknowledge bit of the message MUST be cleared by the device). [CR11-6cd]

Multicasting is activated by setting the destination IP address of the device to a multicast group in the MCDA register. When an application sets a multicast address in MCDA, then it has to set MCTT to 0 to disable acknowledge generation, as indicated by [CR-103cd], to respect the previous conditional requirement.

Note: When multicasting, the destination IP address might range from 239.0.0.0 to 239.255.255.255 for Administratively-scoped (local) multicast addresses and from 224.0.1.0 to 238.255.255.255 for Globally-scoped (Internet-wide) multicast addresses. Refer to RFC3171 (IANA Guidelines for IPv4 Multicast Address Assignments) for more information.

12.7 Traversing Firewalls or Network Address Translation Device

The Message Channel suffers from the same issue related to firewalls than the stream channels. The same mechanism available to help open-up UDP ports for stream channels is available to the message channel.

The Message Channel Capability register (at address 0x0930) can be used to query if the MCSP register is supported by this device.

The application's suggested usage of MCSP is to query it after opening the message channel via MCDA and MCP.

Next, the application can send a UDP packet from the same source port it programmed into MCP to the port specified by MCSP. The content of this packet is ignored, but the suggested payload should be smaller than the maximum size of a GVCP command packet.

Depending on the configuration of the firewall or other impediment, the application may choose to continue periodically sending packets similar to the first one for the duration of the session. The suggested interval is 30 seconds and could be configurable. Different firewalls might require adjustment to this interval.

[CR-110ca] When MCSP is supported by the application, the application MUST not send packets to the port specified by MCSP unless it has the message channel opened and successfully reads a non-zero value back from MCSP. [CR11-8ca]

The application should not expect the device to answer the packet it sends to the device message channel. But this mechanism provides a way to open-up the UDP port in the firewall. The implementation is left as a quality of implementation and is likely to vary with different firewall types and operating systems.

Note that the same mechanism is also available for stream channels.

13 Device with Multiple Network Interfaces

Additional rules are necessary for control, stream and message channels to correctly work when more than one interface is available.

13.1 Impact on Control Channel

[R-111c] The control channel MUST always be instantiated on interface #0. The application MUST take control of the device using that device interface. [R12-6c]

[R-112cd] A device MUST NOT answer GVCP requests coming on an interface different than interface #0. In this case, the message is silently discarded. [R12-7cd]

13.2 Impact on Stream Channels

[R-113st] When streaming data on a given stream channel, the GVSP transmitter MUST use the network interface specified in the *network_interface_index* field of the corresponding SCPx register. [R12-8st]

[CR-114sr] If the GVSP receiver is a device, then when receiving data for a given stream channel, it MUST use the network interface specified in the *network_interface_index* field of the corresponding SCPx register. [CR12-11sr]

Note: A GVSP software receiver that is not a device does not need to implement bootstrap registers.

[R-115ca] An application MUST send the PACKETRESEND_CMD on interface #0 as it is the only one supporting GVCP. In this case, the *stream_channel_index* field will tell the device on which network interface to resend the packet since it maps to SCPx. [R12-9ca]

13.3 Impact on Message Channel

[CR-116c] If supported, the message channel MUST be instantiated on interface #0. [CR12-10c]

14 Additional Concepts

This chapter describes additional concepts for GigE Vision devices.

14.1 Retrieving the XML Device Description File

The GigE Vision specification uses the GenICam™ specification to advertise the device capabilities. The bootstrap registers provide sufficient information for an application to establish communication and to retrieve the XML device description file.

[R-117cd] A GigE Vision device MUST have an XML device description file with the syntax described in the GenApi module of the GenICam standard. [R17-1cd]

For a camera device, the names and types of the device's mandatory features must respect the standard feature list given in Section 29.

Because of the large size of the XML file (which can be a few hundred kilobytes), this specification supports file compression. The file extension indicates the type of compression used by the device. It is up to the application to decompress the file after the download. Note the device only stores a copy of the XML file. As such, it does not need to have specific knowledge about file compression (the XML file might have been compressed prior to be stored in device non-volatile memory).

[R-118ca] Application MUST support uncompressed and compressed (ZIP) XML file.

1. Uncompressed (xml): In this case, the file MUST be stored with the standard .XML extension. The file is an uncompressed text file.
2. Zipped (zip): The file is compressed using the standard .ZIP file format. The file extension MUST be ".zip". Only the DEFLATE and STORE compression methods of the .ZIP file format specification are permitted by this specification. It is up to the application to uncompressed the file after the download. [R17-3ca]

Compression might be useful to minimize the amount of non-volatile memory used to store the file and to minimize the file download time.

Note: The .ZIP algorithm was developed by Phil Katz for PKZIP in January 1989. DEFLATE is a lossless data compression algorithm that uses a combination of the LZ77 algorithm and Huffman coding. It was originally defined by Phil Katz for version 2 of his PKZIP archiving tool.

Note: The specification of the .ZIP is available from PKWARE at http://www.pkware.com/documents/casestudies/APPNOTE.TXT. Free versions of the ZIP compressor and decompressor utilities are available at http://www.info-zip.org.

The XML device description file can be stored in 3 different locations:

1. Device non-volatile memory
2. Vendor web site
3. A local directory on the application machine

A Manifest Table (see section 14.1.4) can be used to indicate the location of the XML device description file. When the Manifest Table is not present, then two bootstrap registers are used to indicate the location of the file:

1. First URL (address 0x0200)
2. Second URL (address 0x0400)

These 2 registers store a string of up to 512 bytes respecting the URI syntax defined in RFC3986. These URLs can be read each using a single call to READMEM.

[R-119cd] A device MUST align the start of the XML file to a 32-bit boundary to ease the retrieve process using READMEM. [R17-5cd]

14.1.1 Device Non-Volatile Memory

When the URL is in the form "local:<*filename*>.<*extension*>;<*address in hex*>;<*length in hex*>", this indicates the XML device description file is available from on-board non-volatile memory (entries in italic must be replaced with actual character values).

Field	Description
local	Indicates the XML device description file is available from on-board non-volatile memory. Case-insensitive.
filename	Name of the file. It is recommended to put the vendor, device and revision information in the filename separated by underscore. For instance: acme_titan_rev1 is suited to revision 1 of the Titan device produced by the Acme company. Case-sensitive.
extension	Indicates the type of file. Case-sensitive. xml: uncompressed text file zip: compressed zip file
address	Starting address of the file in device memory map. It must be expressed in hexadecimal form. Case-insensitive.
length	Length in byte of the file. It must be expressed in hexadecimal form. Case-insensitive.

For example “local:acme_titan_rev1.zip;1C400;A000” indicates “acme_titan_rev1” is a .ZIP file located at address 0x1C400 in device memory map with a length of 0xA000 bytes. Having the filename might be handy to compare with another version of the file that might be available locally on the application machine.

[O-120ca] When the XML device description file is stored locally on the device, the application SHOULD use the READMEM message of GVCP to retrieve it. [O17-6ca]

Because READMEM message is a multiple of 32-bit, the read request might ask for slightly more data than really required to allow for alignment and length restriction.

The way READMEM operates is very similar to TFTP (Trivial file transfer protocol, RFC1350): the file is divided into small blocks of one GVCP packet, and an ACK is required for each block before the application sends a request for the next block.

READMEM is a mandatory command of GVCP. This way, an application can always use this command to get a local XML device description file.

14.1.2 Vendor Web Site

When the URL is in the form “http://www.<*manufacturer.com*>/<*path*>/<*filename.extension*>, this indicates the XML device description file is available from the vendor web site on the Internet. This is a standard URL that can be used directly in a web browser.

Field	Description
http	Indicates the file is available on the Internet. Case-insensitive.
www.*manufacturer.com*	This is the vendor web site. Case-insensitive.
path	Location of the file on the vendor web site. Case-sensitive.
filename	Name of the file. It is recommended to put the vendor, device and revision information in the filename separated by underscore. For instance: acme_titan_rev1 is suited to revision 1 of the Titan device produced by the Acme company. Case-sensitive.
extension	Indicates the type of file. Case-sensitive. xml: uncompressed text file zip: compressed zip file

For example “http://www.acme.com/camera/acme_titan_rev1.xml” indicates the XML device description file is available on Acme web site. The XML file extension indicates the file is an uncompressed text file. Note that URL with extensions different than “.com” (ex: “.org”) are also acceptable.

14.1.3 Local Directory

When the URL is in the form "file:<*filename*>.<*extension*>", this indicates the XML device description file is available on the machine running the application. The file is typically shipped on a CD coming with the device. In this case, the user must put this file in a pre-defined directory where the application stores device description files. It is up to the application software to correctly locate the file that matches the filename.

Field	Description
file	Indicates the XML device description file must be retrieved from a directory on the application machine. Case-insensitive.
filename	Name of the file. It is recommended to put the vendor, device and revision information in the filename separated by underscore. For instance: acme_titan_rev1 is suited to revision 1 of the Titan device produced by the Acme company. Case-sensitive.
extension	Indicates the type of file. Case-sensitive. xml: uncompressed text file zip: compressed zip file

For example, "file:acme_titan_rev1.zip" indicates the XML device description file must be available in the device description folder of the application under the filename "acme_titan_rev1.zip". Because the file extension is ".zip", the file is in ZIP compressed format.

14.1.4 Manifest Table

A GigE Vision device may support multiple versions of the GenApi schema and thus supply multiple XML files.

A manifest table register block shows a list of entries which provide information about the referred XML files. Each entry in the Manifest Table contains the version number of the XML file itself as well as the version number of the GenApi schema the XML file is based on. In addition the entry contains the address of a pair of URL registers which names and locates the document.

The Manifest Table only supports GenICam XML files. However, a GenICam file may provide a mechanism to reference other documents (such as datasheets).

[R-121ca] When the Manifest Table feature is not supported by a device, then the application MUST revert back to the First URL and Second URL registers for the location of the XML description files. The application MUST try to use the first URL choice, and then move to the second choice of URL if unsuccessful in using the first choice. [R17-4ca]

A capability bit (*manifest_table*) is provided in the GVCP Capability register (at address 0x0934) to indicate if the Manifest Table feature is supported.

[R-122ca] If a device provides a Manifest Table, then the application MUST prefer the Manifest Table to retrieve the XML device description file over the First URL and Second URL bootstrap registers.

14.2 Device Synchronization

The only notion of time in the GigE Vision 1.x model is a free running counter clocked at a device-specific rate. This counter is used to timestamp images and events. For single device setup, this allows for sorting images in time. For multi-device setup, since all counters start independently and are running from quartzes with independent drift and precision characteristics, the timestamp cannot be used to order multiple images and events properly.

To address these shortcomings, GigE Vision 2.0 introduces optional support for IEEE 1588-2008. This allows for replacing the undisciplined free running counter of GigE Vision 1.x devices with a network wide synchronized IEEE 1588 clock.

14.2.1 IEEE 1588-2008 Principles

With respect to the UDP/IP protocol, the IEEE 1588 standard defines a network protocol that allows to precisely synchronize clocks in networked measurement and control systems: the Precision Time Protocol (PTP). This protocol enables systems, which could even differ in the precision and stability of their local clocks, to syntonize and synchronize to a grandmaster clock. A network of IEEE 1588 enabled devices selects the device which has the highest precision clock as the grandmaster clock by using a best master clock selection algorithm.

1. To syntonize a local clock to a grandmaster clock is to adjust the frequency of the local clock to the frequency of the grandmaster clock. The duration of a second is identical on both devices.
2. To synchronize a local clock to a grandmaster clock is to set the local clock to the time of the grandmaster.

With moderate effort, the reachable synchronization precision is in the sub-microsecond range.

The basic concept of IEEE 1588 is the exchange of timestamp messages, for which the sender records the precise transmission time and for which the receiver records the precise reception time. To record the timestamps in a very precise manner, specialized hardware must inspect IEEE 1588 packets at the MAC-PHY interface (MII) or in the PHY itself to record the ingress and egress time of those packets as close as possible to the network medium.

This precise measurement of the network transmission times allows a receiver to calculate the exact time on the transmitter side. The drift, delay and offset of the local clock are calculated from the timestamp information and used to syntonize and to synchronize the local clock to the grandmaster clock.

An IEEE 1588 protocol engine contains a control loop that balances out fluctuations in the measurements. The IEEE 1588 standard describes a management protocol that is running in parallel with the GigE Vision protocols, to control and observe the state of the devices regarding the time synchronization.

14.2.2 Timestamp Synchronization

IEEE 1588 is integrated into GigE Vision in such a way to allow IEEE 1588 enabled devices to interoperate with software applications that do not support IEEE 1588.

In legacy network setups two special operation modes, which are defined by IEEE 1588, guarantee the compatibility:

1. An IEEE 1588 enabled device that operates in a network with no other enabled devices will not discipline its local clock. The drift and precision of the local clock is identical to a non-IEEE 1588 enabled device.
2. A network of IEEE 1588 enabled devices, but without any device that has a time traceable to UTC (e.g. GPS receiver), will operate in the ARB timescale. This timescale is relative, i.e. is only valid in the network. The best master clock algorithm will select the clock which has the highest stability and precision as the grandmaster clock of the network.

The difference between the time representation of IEEE 1588 and GigE Vision is solved by converting the IEEE 1588 time structure into a format that is compatible with the existing GigE Vision time structure.

Mapping IEEE 1588 time to GigE Vision timestamp counter

All IEEE 1588 timestamps are expressed in cumulative seconds and nanoseconds since the epoch (1 January 1970 00:00:00 TAI).

The IEEE 1588 defines a Timestamp data structure to store and transport time information. It consists of 2 counters: *secondsFields* and *nanosecondsFields*.

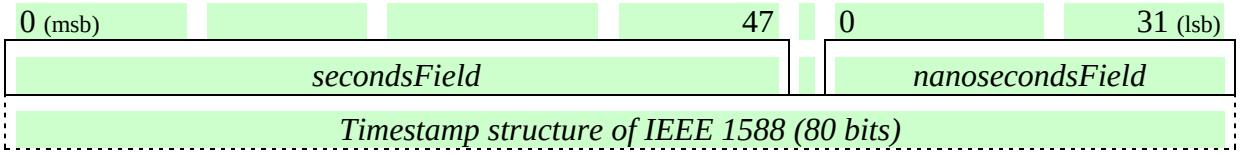


Figure 14-1 : IEEE 1588 Timestamp structure (big-endian representation)

secondsField	unsigned integer	The *secondsField* is the integer portion of the timestamp in units of seconds. It is incremented when the *nanosecondsField* equals 10^9 nanoseconds.
nanosecondsField	integer	The *nanosecondsField* is the fractional portion of the timestamp in units of nanoseconds. The *nanosecondsField* is always less than 10^9.

This 80-bit wide structure is capable of holding nanosecond precise time since the IEEE 1588 epoch. This is called the PTP timescale. The structure can also hold a time in ARB timescale. In this mode the epoch is arbitrary, as it is not bound to an absolute time.

In contrast with this definition of time, GigE Vision has the notion of a 64-bit timestamp counter that is incremented at a device specific rate. This rate can be obtained from the Timestamp Tick Frequency bootstrap register.

To allow a mapping of the IEEE 1588 time into the GigE Vision timestamp a number of restrictions apply.

[CR-123cd] If a device supports IEEE 1588-2008, then its Timestamp Tick Frequency bootstrap register MUST be fixed to 1 000 000 000 (i.e. 1 GHz).

[CR-124cd] If a device supports IEEE 1588-2008, then when it is operating with a disciplined clock, the Timestamp Control register reset feature MUST be disabled. A device MUST return the GEV_STATUS_WRITE_PROTECT status code if an application tries to reset the timestamp by writing to the Timestamp Control register.

In this situation, the value of the GigE Vision Timestamp Value bootstrap register is controlled by the IEEE 1588 protocol. As it is not possible to trigger the GigE Vision Timestamp reset in multiple devices at the same time, using this feature contradicts the synchronization of the clocks and must thus be disabled.

The resolution of the timestamp is reduced from an IEEE 1588 80-bit unsigned value to a 64-bit signed integer (two's complement notation) to accommodate the resolution of the GigE Vision timestamp counter.

Note: This 64-bit signed value directly matches the GenICam integer definition.

For positive value of the timestamp, the 63-bit magnitude (excluding the sign bit) is a continuous representation of the time since epoch, which represents the origin of the timescale, in nanosecond. This reduced timescale of 63 bits will overflow on Nov 4th, 2262 at 23:47:16 GMT. Negative values of the timestamp are not supported by IEEE 1588-2008.

Note: Version 1 of the IEEE 1588 (IEEE 1588-2002) allows negative timestamp values. But this has been removed in version 2 (IEEE 1588-2008).

[CR-125c] All implementations that transfer time information between IEEE 1588-2008 and GigE Vision (applications and devices) MUST handle the conversions between the time formats using *Equation* 14-1 and *Equation* 14-2.

$$t_{GEV} = t_{nanoseconds} + 10^9 \times t_{seconds}$$

Note: If necessary t_{GEV} must be saturated to 0x7FFF FFFF FFFF FFFF

Equation 14-1 : Equation to Convert from IEEE 1588-2008 to GigE Vision

$$t_{nanoseconds} = t_{GEV} \; mod \; 10^9$$

$$t_{seconds} = \left\lfloor \frac{t_{GEV}}{10^9} \right\rfloor$$

Equation 14-2 : Equations to Convert from GigE Vision to IEEE 1588-2008

14.2.3 IEEE 1588 Configuration

IEEE 1588 defines a number of options that can be configured at runtime, to adjust the behavior of the system (e.g. the frequency in which the drift measurements takes place can be adjusted depending on the precision requirements). As IEEE 1588 defines a management protocol on its own it is not necessary to expose those options into GigE Vision bootstrap registers.

The IEEE 1588 Status register (at address 0x096C) contains information about the state of the IEEE 1588 clock. Most IEEE 1588 capable embedded devices signal this information via a LED. As no LED is required for GigE Vision devices, the information has to be stated in the register interface.

14.2.4 IEEE 1588 Profile

The IEEE 1588-2008 standard defines different mechanism to synchronize devices in a network. For a GigE Vision device that is using IEEE 1588-2008 a set of options has to be selected.

This is called an IEEE 1588 profile. A profile should define:

- Configuration and node management options
- Path delay measurement option
- Range and default values of all configurable attributes
- Required, permitted, or prohibited transport mechanisms
- Required, permitted, or prohibited options
- Required, permitted, or prohibited node types
- Procedure to verify conformance

[CR-126c] If a GigE Vision device or application supports IEEE 1588, then it must support the default PTP profile for use with the delay-response mechanism, as per IEEE 1588-2008.

> Default PTP profile for use with the delay request-response mechanism.
>
> Version 1.0
>
> Profile identifier: 00-1B-19-00-01-00
>
> This profile is specified by the Precise Networked Clock Synchronization Working Group of the IM/ST Committee.
>
> A copy may be obtained by ordering IEEE Std 1588-2008 from the IEEE Standards Organization.
>
> http://standards.ieee.org/

14.3 Action Commands

Triggering an action in multiple devices at roughly the same time can be accomplished through the optional ACTION_CMD message. Using a distinct command (instead of a WRITEREG_CMD) has the advantage that it does not rely on a specific register map. Therefore, different type of devices can be targeted with the same message. Note that due to the nature of Ethernet, this is not as synchronous as a hardware trigger since different network segments can have different latencies. Nevertheless in a switched network, the resulting jitter is acceptable for a broad range of applications and this scheme provides a convenient way to synchronize devices by means of a software command.

ACTION_CMD message can be unicasted or broadcasted by the primary or any secondary applications to the GVCP port of the device. The acknowledge (ACTION_ACK) sent back by the device is always a unicast message.

The most typical scenario is when an application desires to trigger a simultaneous action on multiple devices. This case is showed by the figure below for a primary application. The application fires a broadcast ACTION_CMD that will reach all the devices on the subnet.

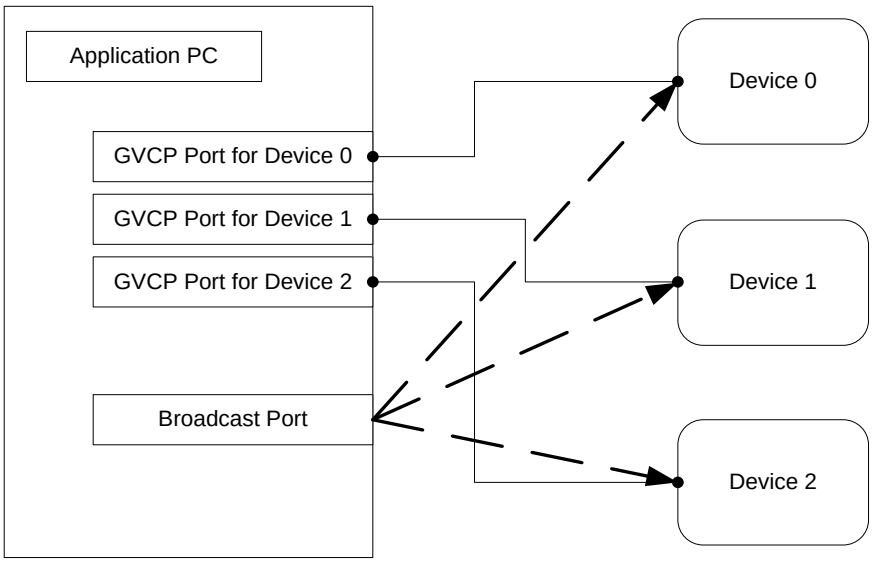


Figure 14-2: ACTION_CMD Broadcast from Primary Application

But ACTION_CMD can also be used by secondary applications. This can even be another device on the same subnet. This is depicted in the following figure.

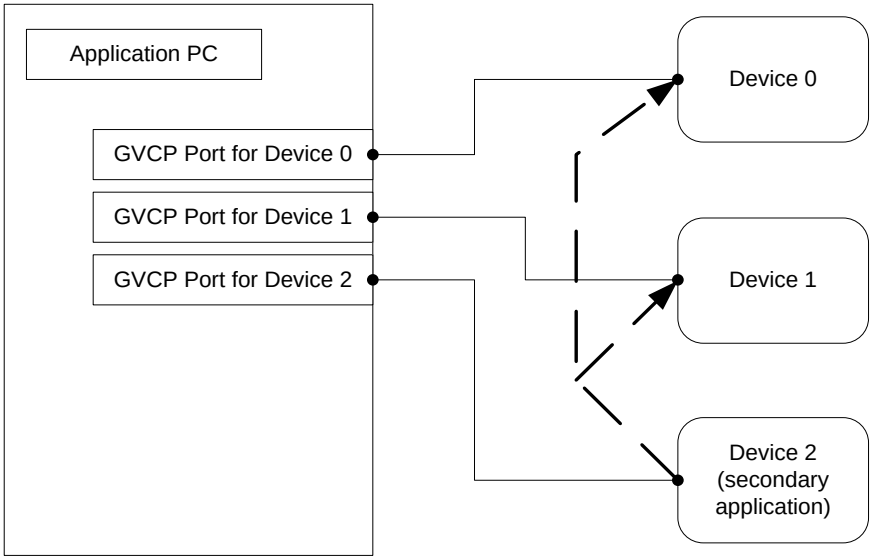


Figure 14-3: ACTION_CMD Broadcast from Another Source (Secondary Application)

Upon reception of an ACTION_CMD message, the device will decode the information to identify which internal action signal is requested. An action signal is a device internal signal that can be used as a trigger for functional units inside the device (ex: a frame trigger). It can be routed to all signal sinks of the device.

Each ACTION_CMD message contains information for the device to validate the requested operation:

1. *device_key* to authorize the action on this device.
2. *group_key* to define a group of devices on which actions have to be executed.
3. *group_mask* to be used to filter out some of these devices from the group.

Action signals can only be asserted if the device has an open primary control channel or when unconditional action mode is enabled.

Table 14-1: ACTION Command Four Conditions

The conditions for an action signal ACTION_x to be asserted by the device are:

1. the device has an open primary control channel or unconditional action mode is enabled.
2. the *device_key* in the ACTION_CMD packet and the Action Device Key configured in the boostrap register (at address 0x090C) must be equal.
3. the *group_key* in the ACTION_CMD packet and the ACTION_GROUP_KEYx of the corresponding device action bootstrap register for ACTION_x must be equal.
4. the logical AND-wise operation of *group_mask* in the ACTION_CMD packet against the ACTION_GROUP_MASKx of the corresponding device action bootstrap register for ACTION_x must be non-zero. Therefore, they must have at least one common bit set at the same position in the 32-bit register.

When these 4 conditions are met for at least one action signal ACTION_x, then the device internally asserts the requested action and returns an ACTION_ACK packet. As these conditions could be met on more than one action, the device could assert more than one action signal in parallel. When one of the 4 conditions is not met for any supported action signal, then the device ignores the ACTION_CMD request and does **not** even return an ACTION_ACK.

This first condition asks for the primary control channel to be opened unless the unconditional action bit is set in the GVCP Configuration register at address 0x0954. When this bit is set, the device can assert the requested action even if the primary control channel is closed as long as the 3 other conditions are met. It is left as a quality of implementation for the system to ensure that the device is properly configured to process those action command requests even though the primary control channel is closed.

14.3.1 Scheduled Action Commands

Scheduled Action Commands provide a way to trigger actions in a device at a specific time in the future. The typical use case is depicted in the following diagram:

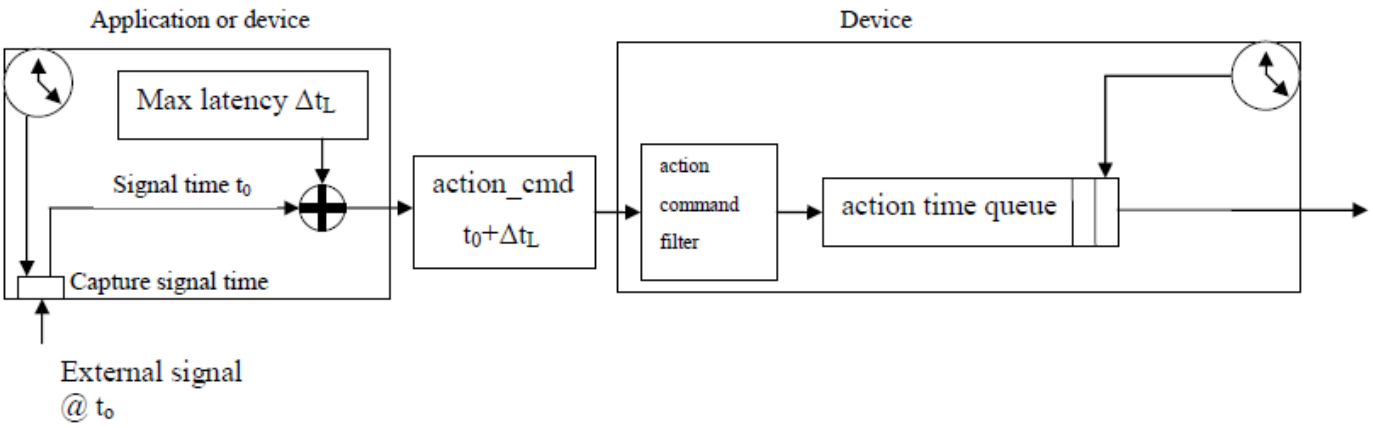


Figure 14-4 : Principle of Scheduled Action Commands

The transmitter of an action command records the exact time, when the source signal is asserted (External signal). This time t_0 is incremented by a delta time Δt_L and transmitted in an ACTION_CMD packet to the receivers. The delta time Δt_L has to be larger than the longest possible transmission and processing latency of the ACTION_CMD packet in the network.

If the packet passes the action command filters in the receiver, then the action signal is put into a time queue (the depth of this queue is indicated by the Scheduled Action Command Queue Size register) and the device sends the ACTION_ACK immediately. It does not wait for the action to be executed as this might take some time and the application is waiting for the acknowledge on the control channel.

[CR-127cd] If the device supports Scheduled Action Commands, then if the Scheduled Action Commands queue is full and the device receives an additional ACTION_CMD, then the device MUST return an ACTION_ACK with a GEV_STATUS_OVERFLOW status code (if the appropriate extended status code are enabled in the GVCP Capability register) and it MUST NOT en-queue this request.

[CR-128cd] If the device supports Scheduled Action Commands, then if the reference time is not synchronized to any master clock, then the device MUST return an ACTION_ACK with a GEV_STATUS_NO_REF_TIME status code (if the appropriate extended status code are enabled in the GVCP Capability register) and it MUST NOT queue this request.

When the time of the local clock is greater or equal to the time of an action signal in the queue, the signal is removed from the queue and asserted.

Combined with the timestamp precision of IEEE 1588 which can be sub-microseconds, a Scheduled Action Command provides a way to allow low-jitter software trigger.

If the sender of an action command is not capable to set a future time into the packet, the action command has a flag to fall back to legacy mode (bit 0 of the *flag* field). In this mode the signal is asserted in the moment the packet passes the action command filters.

14.3.2 ACTION_CMD examples

The following examples illustrate the behavior of the ACTION_CMD message in various scenarios. The figure below shows 4 different ACTION_CMD requests issued by a primary or secondary application. Content of the ACTION_CMD packet is illustrated on the left side of the figure.

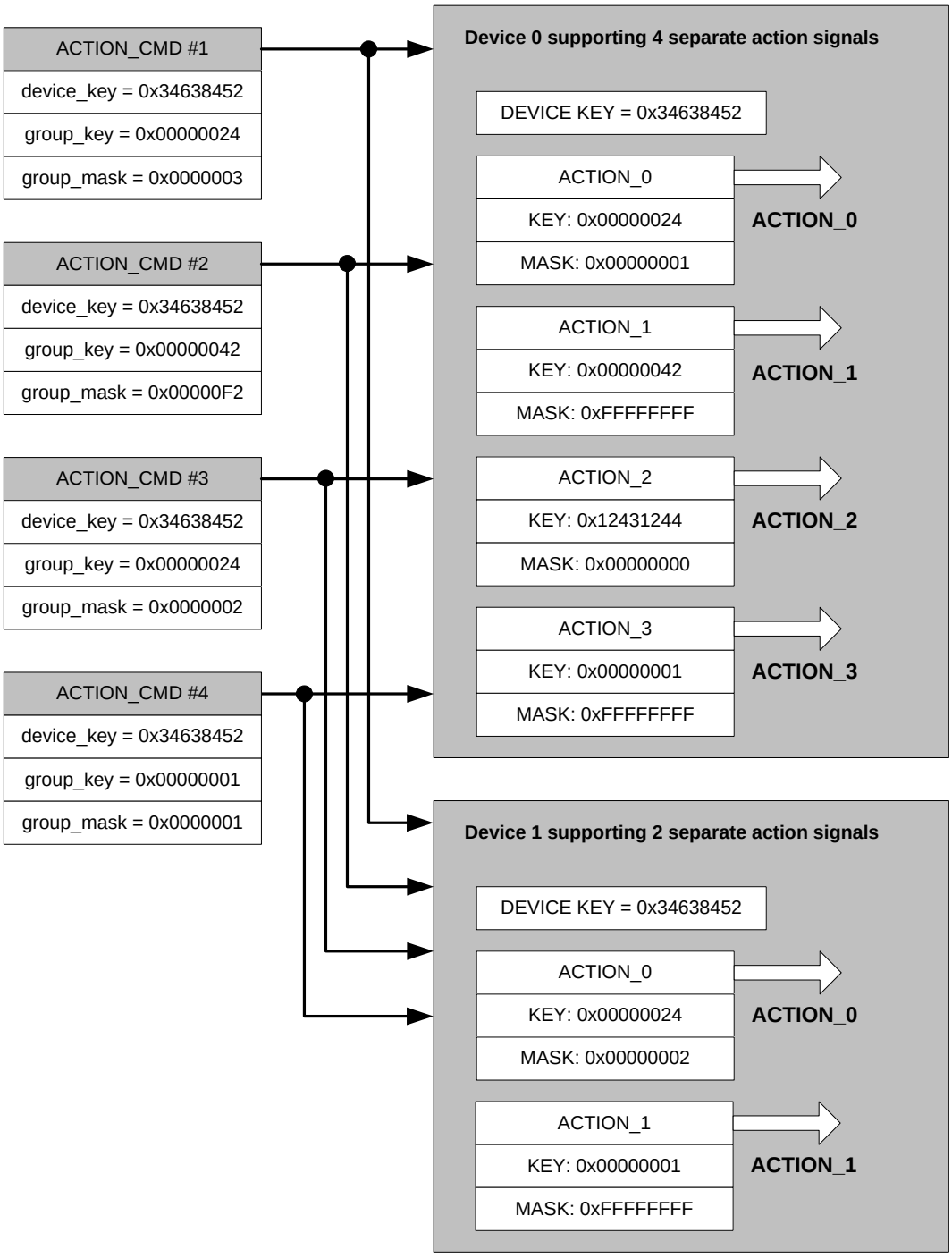


Figure 14-5: ACTION_CMD examples

The content of the ACTION_CMD request must be examined against the 4 conditions listed in Table 14-1 for each supported action signal.

For the first request (ACTION_CMD #1):

	ACTION_CMD #1	Device 0			Device 1
		ACTION_0	**ACTION_1**	**ACTION_2**	**ACTION_0**
device_key	0x34638452	0x34638452	0x34638452	0x34638452	0x34638452
group_key	0x00000024	0x00000024	**0x00000042**	**0x12341244**	0x00000024
group_mask	0x00000003	0x00000001	0xFFFFFFFF	**0x00000000**	0x00000002

Device 0 receives the request and looks for the 4 conditions.

1. a primary control channel is open (or unconditional action is enabled)
2. device_key matches
3. group_key matches
4. Logical AND-wise comparison of requested group_mask is non-zero

All 4 conditions are met only for ACTION_0, thus the device immediately sends back an ACTION_ACK with GEV_STATUS_OK and asserts the internal signal represented by ACTION_0.

The same steps are followed by Device 1. Only the *group_mask* is different, but nevertheless the logical bitwise AND operation produces a non-zero value, leading to the assertion of ACTION_0 by Device 1 and the transmission of an acknowledge.

For the second request (ACTION_CMD #2):

	ACTION_CMD #2	Device 0			Device 1
		ACTION_0	**ACTION_1**	**ACTION_2**	**ACTION_0**
device_key	0x34638452	0x34638452	0x34638452	0x34638452	0x34638452
group_key	0x00000042	**0x00000024**	0x00000042	**0x12341244**	**0x00000024**
group_mask	0x000000F2	**0x00000001**	0xFFFFFFFF	**0x00000000**	0x00000002

Looking for the 4 conditions, Device 0 will assert ACTION_1 while Device 1 will not assert any signal because the *group_key* condition is not met. Therefore, Device 1 ignores the request and does not send back an acknowledge.

For the third request (ACTION_CMD #3):

	ACTION_CMD #3	Device 0			Device 1
		ACTION_0	ACTION_1	ACTION_2	ACTION_0
device_key	0x34638452	0x34638452	0x34638452	0x34638452	0x34638452
group_key	0x00000024	0x00000024	**0x00000042**	**0x12341244**	0x00000024
group_mask	0x00000002	**0x00000001**	0xFFFFFFFF	**0x00000000**	0x00000002

In the third example, the *group_mask* and *group_key* of Device 0 do not match with ACTION_CMD #3 for any of the ACTION_0 to ACTION_2. Therefore, Device 0 ignores the request. Device 1 asserts ACTION_0 since the 4 conditions are met.

The ACTION_CMD is flexible enough to accommodate "simultaneous" triggering of the same functional action in functionally different devices.

For instance, let's assume the software trigger of Device 0 can only be associated to its ACTION_3 and that the software trigger of Device 1 can only be associated to its ACTION_1. The ACTION_CMD defined in this specification enables an ACTION_CMD to trigger the same functional action provided that their respective action group key and masks are set in order to meet the condition of Table 14-1.

For the fourth request (ACTION_CMD #4):

	ACTION_CMD #4	Device 0	Device 1
		ACTION_3	ACTION_1
device_key	0x34638452	0x34638452	0x34638452
group_key	0x00000001	0x00000001	0x00000001
group_mask	0x00000001	0xFFFFFFFF	0xFFFFFFFF

In this case, Device 0 asserts ACTION_3 and Device 1 asserts ACTION_1 since the conditions of Table 14-1 are met. As a result of this, the software trigger of both devices can be "simultaneously" triggered even though they are associated to different action numbers.

14.4 Primary Application Switchover

For systems where redundancy and fault recovery are required, it is often necessary for a second application to take control over a device that is already under the control of a primary application. This may be needed in the case where the primary application, or part of it, becomes irresponsive or crashes and quick recovery of devices is mandatory. It can also be required for another application to take over when dictated by a higher level system management entity.

It is very important to indicate that it is outside the scope of this specification to describe when an application should take control over another one and how another application should be notified to do so.

These are left as higher level system integration considerations. It is up to the system architect to define the communication mechanisms between various applications and when the switchover needs to occur. What is critical is that this specification makes application switchover possible.

Primary application switchover is made possible by using the control access with switchover enabled access mode. By connecting to a device using this mode, a primary application enables another application to take control provided that it has the right credentials. This means that the other application needs to know the key required to grant the switchover. This key is held by the Control Switchover Key write-only register. When a device has granted control access with switchover enabled, another application can request control over the device by performing a write access to the CCP register. One field of the CCP register enables the other application to provide a 16-bit data value to use in the switchover authentication process. If the value provided matches the value held by the Control Switchover Key register, then the switchover is granted. This authentication process provides a minimal protection to prevent another application to take control over a device by mistake.

In order to notify the primary application that a switchover has occurred, a device can optionally be configured to send an event (GEV_EVENT_PRIMARY_APP_SWITCH) to make the primary application aware that it lost control of the device. In this case, the device sends the event before granting access to the new primary application. This is provided that the primary application has instantiated a message channel and enabled the GEV_EVENT_PRIMARY_APP_SWITCH event. A device issuing this event should not request an acknowledge since the message channel may be under the control of a new primary application by the time the previous primary application acknowledges the packet. It should be noted that a primary application switchover doesn't close the message channel. It is up to the new primary application to manage it.

The pseudo-code algorithm below presents how a device reacts to a write access to the CCP register. It highlights the rules that must be met in order for a primary application switchover to occur.

```
If an application writes to the CCP register then
    If the device access mode is "open access" then
        Grant the requested access mode.
    Else
        If the write comes from the primary application then
            Grant the requested access mode.
        Else
            If the device is in control access with switchover enabled then
                If the control_switchover_key field matches the value set in
                the Control Switchover Key register then
                    Grant the primary application switchover with the requested
                    access mode.
                Else
                    Deny the request.
                End if
            Else
                Deny the request.
            End if
        End if
    End if
End if
```

14.4.1 Primary Application Switchover Setup Example

This section provides guidelines regarding how primary application switchover can be setup on a device. In the process described below, it is assumed that the device is not under the control of any application at the beginning of the following procedural steps.

1. The primary application requests, and gets granted, exclusive access.
2. The primary application verifies that the device supports primary application switchover by consulting bit 10 of the GVCP Capability register.
3. The primary application configures the Control Switchover Key register.
4. The primary application requests, and gets granted, control access with switchover enabled. Please note that this is done without closing the control channel.
5. The device is now setup for primary application switchover. Another application that has the right credentials, i.e. that knows the key, can request, and gets granted, device control.

It should be noted that this section provides best usage guidelines but one could follow different steps. Its intent is not to cover all possible scenarios. For instance, one may decide that it is not necessary to consult the GVCP Capability register to determine if primary application switchover is supported by a device if it is already known that the device supports this feature.

15 GVCP Headers

GVCP defines 2 types of header:

1. Command header
2. Acknowledge header

All protocol headers are defined to be 32-bit naturally aligned. Their representation is always big-endian.

In the following section, the nomenclature refers explicitly to the control channel, but a message channel must react in an equivalent manner.

15.1 Command Header

The following figure shows the command message header. Note the figure does not show Ethernet, IP and UDP headers.

[R-129c] Command headers MUST respect the layout of Figure 15-1. [R13-1c]

[O-130c] A recipient of a command message SHOULD perform minimal validation on the command header by verifying the presence of the 0x42 key in the first byte of the packet. A packet without this key is not considered to be a GVCP message. [O13-2c]

[R-131c] If the message is not a GVCP message, then it MUST be silently discarded. [R13-3c]

[R-132c] If the command requested in the *command* field is not supported by the recipient of the command, then the recipient SHOULD return a GEV_STATUS_NOT_IMPLEMENTED status code when an acknowledge is requested. [R13-4c]

In the previous situation, the *acknowledge* message value to use to answer the unsupported command is the *command* message value + 1. The specification always defines the *acknowledge* message value to be the associated *command* message value + 1. Refer to [R-238c] for the *command* and *acknowledge* values. No data payload is put in the acknowledge packet when such an error occurs.

[R-133c] The reserved fields in a GVCP packet MUST be set to 0 on transmission and MUST be ignored on reception.

[O-134c] If any other field of the command message header is invalid, then the recipient SHOULD return a GEV_STATUS_INVALID_HEADER status code. [O13-5c]

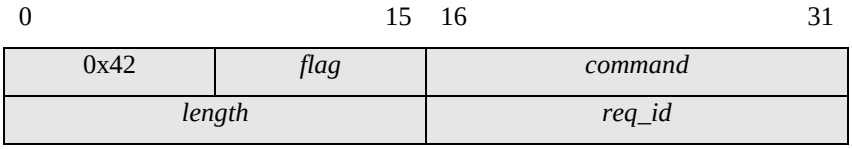


Figure 15-1: Command Message Header

COMMAND MESSAGE HEADER		
0x42	8 bits	Hard-coded key code value used for message validation.
flag	8 bits	Upper 4 bits (msb, bit 0 to bit 3) are specific to each GVCP commands. Lower 4 bits (lsb, bit 4 to bit 7) are common to all GVCP commands. All unused bits must be set to 0. bit 0 to 3 – Specific to each command. Set to 0 if the command does not use them. bit 4 – Reserved, set to 0 on transmission, ignore on reception. bit 5 – Reserved, set to 0 on transmission, ignore on reception. bit 6 – Reserved, set to 0 on transmission, ignore on reception. bit 7 – ACKNOWLEDGE: SET = requires the recipient of this message to send an acknowledge message. CLEAR = recipient of this message shall not send an acknowledge message. Typically used for UDP. Some commands might not require acknowledge. This can be used to limit the amount of bandwidth for control. If this bit is cleared and the command normally requires an acknowledge, then this bit has precedence and NO ack message is sent (for instance, reading a register to reset the heartbeat would not send back the register value to the application). Acknowledge messages can be used to enforce flow-control.
command	16 bits	Command message value (see dictionary section). Commands from 0 to 32765 are reserved for GigE Vision. Others are available for customization (device-specific).
length	16 bits	Number of valid data bytes in this message, not including this header. This represents the number of bytes of payload appended after this header.
req_id	16 bits	This field is the request ID generated by the application. The device copies this value in the corresponding acknowledges's *ack_id* field. *req_id* is used to match a command with its acknowledge. *req_id* must never be set to 0. The application must use sequential *req_id* where the value is incremented (typically by 1) between each message (except when *req_id* wraps back to 0 where it must be set to 1 since 0 is not a valid value).

15.2 Acknowledge Header

The following figure shows the acknowledge message header. Note the figure does not show Ethernet, IP and UDP headers.

[R-135c] Acknowledge headers MUST respect the layout of Figure 15-2. [R13-6c]

[O-136c] The recipient of an acknowledge message SHOULD perform minimal validation of the acknowledge header. This minimally includes the validation of the *ack_id* against the *req_id* sent previously, but it may also include the validation of the *acknowledge* field against a value listed in this specification. [O13-7c]

Note the emitter of the acknowledge message is allowed to set the *length* field to 0 when it returns a suitable error code in the *status* field if no data payload is appended to this acknowledge message header.

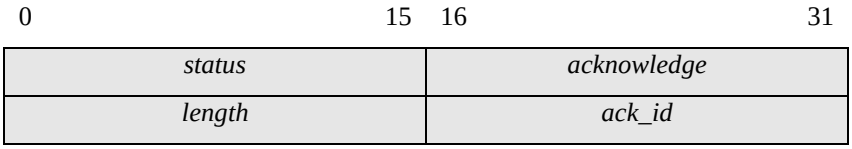


Figure 15-2: Acknowledge Message Header

ACKNOWLEDGE MESSAGE HEADER		
status	16 bits	Status of the requested operation (see status code section)
acknowledge	16 bits	Acknowledge message value (see dictionary section).
length	16 bits	Number of valid data bytes in this message, not including this header. This represents the number of bytes of payload appended after this header.
ack_id	16 bits	This field is the acknowledge ID and has the same value as the corresponding command *req_id*.

15.3 Byte Sequencing

[R-137c] GVCP MUST use network byte order, also known as big-endian. [R13-9c]

The sequence of bytes of a message to send is provided by Appendix B of RFC791. The order of transmission of the header data is illustrated in the figure below, from byte 1 to byte 8. It goes from left to right, then from top to bottom. Notice that bit 0 (msb) is on the left, while bit 31 (lsb) is on the right. Any GVCP data payload follows the same sequence when it is transmitted on the network.

The order of transmission of the header and data described in this document is resolved to the byte level. Whenever a diagram shows a group of bytes, the order of transmission of those bytes is the normal order in which they are read in English. For example, in the following diagram the bytes are transmitted in the order they are numbered, from 1 to 8.

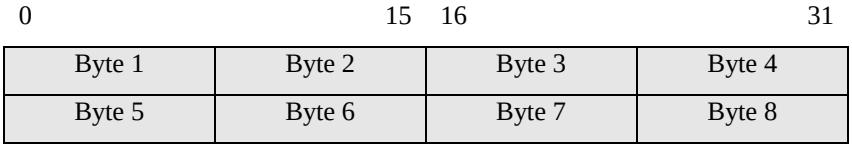


Figure 15-3: Byte Sequencing

Whenever a byte represents a numeric quantity the leftmost bit in the figure is the most significant bit. Therefore, the bit labeled 0 is the most significant bit, as shown in next figure.

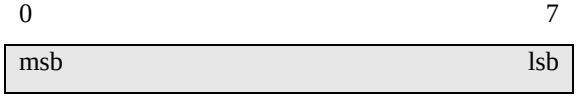


Figure 15-4: Bit Sequencing for an 8-bit Word (Big-Endian)

Similarly, whenever a multi-byte field represents a numeric quantity the leftmost bit of the whole field is the most significant bit. When a multi-byte quantity is transmitted the most significant byte is transmitted first.

For a 16-bit word:

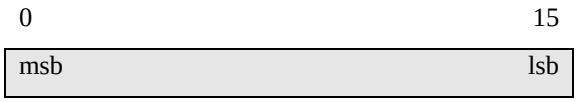


Figure 15-5: Bit Sequencing for a 16-bit Word (Big-Endian)

For a 32-bit word:

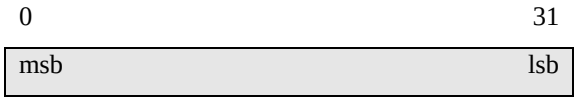


Figure 15-6: Bit Sequencing for a 32-bit Word (Big-Endian)

The following figure shows the breakdown of GVCP header into byte fields.

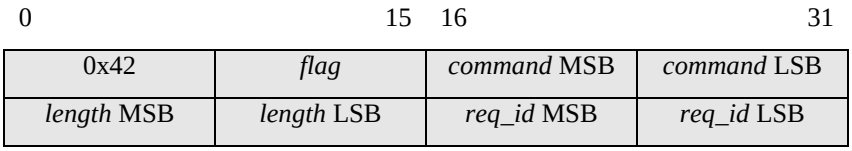


Figure 15-7: GVCP Header Showed in Byte Quantities

Therefore, for GVCP header provided above, the appropriate sequence of bytes on the network is:

Table 15-1: GVCP Header Byte Transmission

Byte	Value
1	key value 0x42
2	*flag*
3	*command* MSB
4	*command* LSB
5	*length* MSB
6	*length* LSB
7	*req_id* MSB
8	*req_id* LSB

This order of transmission is called network byte order. It is equivalent to big-endian. A little-endian machine must swap multi-byte fields (i.e. 16 and 32-bit word) to correctly interpret the value. 8-bit fields are not swapped (this includes 8-bit character strings).

The following figure shows the breakdown of a specific GVCP command header into byte fields.

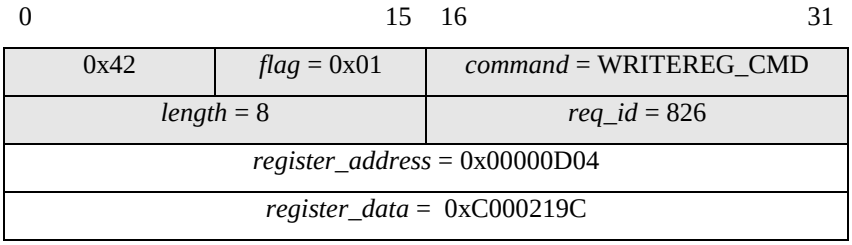


Figure 15-8: Example Command for Byte Ordering

Assume the following parameters:

flag	0x01	bit 7 is SET, requiring the recipient of this message to send an acknowledge message. Since bit 7 is the lsb, the resulting value will be 0000 0001b (0x01).
command	0x0082	WRITEREG_CMD
length	0x0008	8
req_id	0x033A	826
register_address	0x00000D04	SCPS0
register_data	0xC000219C	bit 0 and 1 are SET. This device will fire one unfragmented test packet of the size specified by bits 16-31. Since bit 0 is the most significant bit, the resulting value for the bit flags will be 1100 0000 0000 0000 0000 0000 0000 0000b (0xC0000000). The packet size will be 8604 bytes. Since bit 16 is the most significant bit for *packet_size* field, the resulting value for the packet size will be 0000 0000 0000 0000 0010 0001 1001 1100b (0x0000219C). Combining the bit fields, we get the result 1100 0000 0000 0000 0010 0001 1001 1100b (0xC000219C).

Therefore, for GVCP command provided above, the appropriate sequence of bytes on the network is:

Byte	Definition	Value
1	key value	0x42
2	*flag*	0x01
3	*command* MSB	0x00
4	*command* LSB	0x82
5	*length* MSB	0x00
6	*length* LSB	0x08
7	*req_id* MSB	0x03
8	*req_id* LSB	0x3A
9	*register_address* MSB	0x00
10	*register_address*	0x00
11	*register_address*	0x0D
12	*register_address* LSB	0x04
13	*register_data* MSB	0xC0
14	*register_data*	0x00
15	*register_data*	0x21
16	*register_data* LSB	0x9C

Note: Implementations using the socket API can rely on the following standard byte sequencing functions: ntohl() to convert a 32-bit word from network to host byte order, htonl() to convert a 32-bit from host to network byte order, ntohs() to convert a 16-bit word from network to host byte order, and htons() to convert a 16-bit word from host to network byte order. 8-bit words do not require any conversion.

16 Control Channel Dictionary

Messages in this category are sent on the Control Channel, from the application to the device. Acknowledges are sent in the opposite direction. Before executing any command, the device verifies if the requester has the necessary privilege.

[R-138cd] If the requester of a control message does not have the necessary privilege, the device MUST return a GEV_STATUS_ACCESS_DENIED in the *status* field of the acknowledge message and set the *length* field to 0. The acknowledge message only contains the header. [R14-1cd]

An application receiving this GEV_STATUS_ACCESS_DENIED error should ignore any data that might be attached to the acknowledge to facilitate interoperability with previous versions of this specification.

16.1 DISCOVERY

[R-139cd] The DISCOVERY command MUST be supported by all devices. [R14-2cd]

The DISCOVERY command is required to support device enumeration. No privilege is required to execute this command. A device has to answer this request from any application (primary and secondary) at all time once it has an IP address.

16.1.1 DISCOVERY_CMD

To enumerate devices, an application sends a DISCOVERY_CMD message.

[R-140c] DISCOVERY_CMD MUST follow the layout provided by Figure 16-1. [R14-3c]

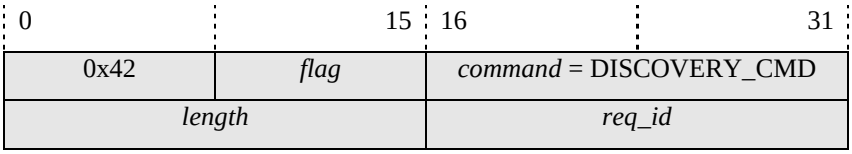


Figure 16-1: DISCOVERY_CMD Message

DISCOVERY_CMD		
flag	8 bits	bit 0 to 2 – Reserved, set to 0 on transmission, ignore on reception. bit 3 – Allows broadcast of acknowledge bit 4 to 7 – Use standard definition from GVCP header

Bit 3 of the *flag* field can be set by an application to indicate a device is allowed to broadcast its DISCOVERY_ACK message if the conditions mentioned in the DISCOVERY_ACK section below are met. When this bit is cleared, the device cannot broadcast its acknowledge message.

[R-141ca] For a DISCOVERY_CMD message, the ACKNOWLEDGE bit (bit 7) of the *flag* field MUST be set. [R14-62ca]

16.1.2 DISCOVERY_ACK

[R-142cd] A device MUST unicast its DISCOVERY_ACK if it is on the same subnet as the application. [R14-5cd]

The above requirement is true independent of the way the DISCOVERY_CMD was sent (limited broadcast, net-directed broadcast or unicast).

Note: If its IP configuration is not properly configured, it might not be possible for the device to adequately validate the subnet mask of the application.

[O-143cd] A device SHOULD broadcast its DISCOVERY_ACK when the following 2 conditions are met:

1. the DISCOVERY_CMD was not received on the device subnet;
2. bit 3 of the *flag* field of DISCOVERY_CMD was set by the application.

The above scenario typically happens when a device is configured with a Persistent IP address that does not match the subnet of the application.

When bit 3 of the *flag* field is cleared and the device is not on the same subnet as the application, then the former cannot acknowledge the device discovery. In this situation, the device is not visible to the application.

Table 16-1 : DISCOVERY Broadcast Summary

Discovery Packet	Broadcast Flag	Device and application on same subnet	DISCOVERY_ACK Broadcasted	Notes
Unicast	x	x	No	
Limited (255.255.255.255) or subnet directed broadcast	0	x	No	
	1	No	Optionally	Acknowledging using a limited or subnet directed broadcast is left as an implementation detail.
	1	Yes	No	

[O-144cd] If the Discovery ACK Delay bootstrap register is supported, the device SHOULD wait for a randomized delay with a uniform distribution from 0 up to the value in the Discovery ACK Delay register before sending the DISCOVERY_ACK. Otherwise, the device SHOULD wait for a randomized delay with a uniform distribution from 0 to 1 second before sending the DISCOVERY_ACK. [O14-6cd]

This way, if a huge number of devices receive the broadcast message simultaneously, they are less likely to saturate the application.

[R-145c] DISCOVERY_ACK MUST follow the layout provided by Figure 16-2. [R14-7c]

manufacturer_specific_information [48 bytes]
serial_number [16 bytes]
user_defined_name [16 bytes]

Figure 16-2: DISCOVERY_ACK Message

The fields in this acknowledge, except for the GVCP header, identically match to the first few bootstrap registers. Device discovery is always performed on the first network interface (interface #0) since it is the only one to support GVCP. Refer to bootstrap registers for the definitions of these fields.

Note: In GigE Vision 1.x, strings were always NULL-terminated. With GigE Vision 2.x, strings do not need to be NULL-terminated if they occupy the full memory area allocated to the string bootstrap register. In this case, it is the responsibility of the application to append the NULL character.

[R-146cd] When an error is generated during the processing of the DISCOVERY_CMD, the device MUST reply with an appropriate status code set in the DISCOVERY_ACK header. In this case, the *length* field must be set to 0 and the acknowledge message only contains the header. [R14-8cd]

The above requirement allows an application to retrieve the cause of the error.

16.2 FORCEIP

In some situations, it might be useful for the application to "force" an IP address in a device. For instance, when a device is using an unidentified Persistent IP, then it might be difficult for the application to communicate with the device since the subnet of the application would likely be different from the subnet of the device. This issue does not happen with DHCP and LLA since the IP address is negotiated.

To solve this problem, GVCP provides a mechanism to force a static IP address into the device, as long as the application knows the MAC address of the device.

[R-147cd] The FORCEIP command MUST be supported by all devices. [R14-9cd]

The FORCEIP command is required to force an IP address into the device identified by the MAC address. It offers provision to set the subnet mask and default gateway associated to this IP address.

16.2.1 FORCEIP_CMD

[CR-148ca] If supported by the application, the FORCEIP_CMD MUST be broadcasted by the application on the GVCP port. It must contain the MAC address of the device to access. [CR14-10ca]

[R-149cd] If the *MAC_address* field of FORCEIP_CMD does not match the device's MAC address, then the message MUST be silently discarded by the device. [R14-11cd]

[R-150cd] A device MUST process FORCEIP_CMD only when there is no application with exclusive or control access (without or with switchover enabled). Otherwise, this request MUST be silently discarded. [R14-12cd]

The above requirement indicates that FORCEIP_CMD is not coming from the primary application. This is necessary to allow this command to be used to recover a device with an unknown IP address.

This request can be used to accomplish two different actions on the device with the specified MAC address:

[R-151cd] If the *static_IP* field of FORCEIP_CMD is 0, then the device MUST restart its IP configuration cycle on all its network interface without sending a FORCEIP_ACK back to the application. [R14-13cd]

[R-152cd] If the *static_IP* field of FORCEIP_CMD is different from 0, then the device MUST change its IP address to the value specified in the *static_IP* field and only send back a FORCEIP_ACK (if requested) when the newly assigned static IP address is effective. If assigning the static IP address requires the device to reset its communication stack and internal state, then static IP address MUST be maintained after this reset. [R14-14cd]

Note: Any time a device gets assigned a new IP address, it is recommended for the device to announce it by sending a gratuitous ARP. This allows remote hosts to update their ARP cache.

[R-153cd] FORCEIP_CMD MUST follow the layout of Figure 16-3. [R14-15cd]

[CR-154ca] If supported, FORCEIP_CMD MUST follow the layout of Figure 16-3. [CR14-76ca]

0	15	16	31
0x42	*flag*	*command* = FORCEIP_CMD	
length		*req_id*	
reserved		*MAC_address* (high)	
MAC_address (low)			
reserved			
reserved			
reserved			
static_IP			
reserved			
reserved			
reserved			
static_subnet_mask			
reserved			

reserved
reserved
static_default_gateway

Figure 16-3: FORCEIP_CMD Message

FORCEIP_CMD		
flag	8 bits	bit 0 to 2 – Reserved, set to 0 on transmission, ignore on reception. bit 3 – Allows broadcast of acknowledge (version 1.1 and above only) bit 4 to 7 – Use standard definition from GVCP header
reserved	16 bits	Set to 0 on transmission, ignore on reception.
MAC_address	48 bits	MAC address of the device to access
reserved	96 bits	Set to 0 on transmission, ignore on reception.
static_IP	32 bits	Static IP address to force into the device
reserved	96 bits	Set to 0 on transmission, ignore on reception.
static_subnet_mask	32 bits	Subnet mask to use with the static IP address
reserved	96 bits	Set to 0 on transmission, ignore on reception.
static_default_gateway	32 bits	Default gateway to use with the static IP address

For device supporting GigE Vision specification 1.1 and above, bit 3 of the *flag* field can be set by an application to indicate a device should broadcast its FORCEIP_ACK message. This can be useful if the subnet changes during the execution of this command. When this bit is cleared, the device cannot broadcast its acknowledge message. For GigE Vision 1.0 devices, this bit must stay cleared.

16.2.2 FORCEIP_ACK

The FORCEIP_ACK is sent back by the device after the forced static IP address has been assigned. The *status* field is used to signal any problem.

[O-155cd] A device SHOULD broadcast its FORCEIP_ACK when bit 3 of the *flag* field of FORCEIP_CMD is set by the application.

[R-156cd] FORCEIP_ACK MUST follow the layout of Figure 16-4. [R14-16cd]

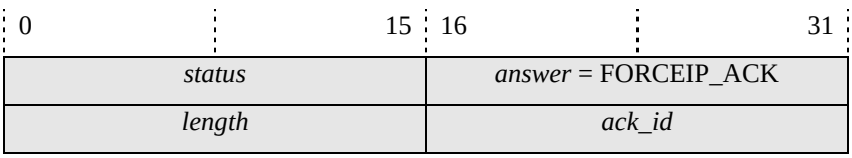


Figure 16-4: FORCEIP_ACK Message

This message must be sent with the newly forced IP address as the source address in the IP packet. This acknowledge should be broadcasted if the broadcast bit is set in the *flag* field of the FORCEIP_CMD.

16.3 READREG

[R-157cd] The READREG command MUST be supported by all devices. [R14-17cd]

The application can issue a READREG message to read a device register. Multiple reads can be concatenated in one message, as long as the total packet size is lower or equal to 576 bytes. Therefore, a maximum of 135 registers can be read with one message if the device supports concatenation of operations (36 bytes are used by the various headers). At least one register must be read. The number of read operations to perform can be deduced from the *length* field of the command header. It is equal to "*length* / 4".

[R-158cd] The device MUST have a capability flag indicating if multiple register reads can be concatenated (bit 31 of GVCP Capability register, at address 0x0934). [R14-18cd]

Note: An application cannot send a concatenated READREG command to the device when this bit is cleared.

[CR-159cd] When concatenation is supported and multiple register reads are specified, they MUST be executed sequentially by the device. [CR14-19cd]

[CR-160cd] When concatenation is supported, if an error occurs, register read operation MUST stop at the location of the error. Remaining read operations are discarded and an appropriate error code is returned in the acknowledge message. [CR14-20cd]

The *length* field of the acknowledge message is used to deduce number of read operations performed successfully (each register provides 4 bytes).

Note: The control protocol assumes the registers are 32-bit wide. It is up to the device to perform a suitable conversion if its internal registers have a different size.

[R-161cd] If no application has exclusive access, then the device MUST answer READREG messages coming from the primary or from any secondary application. [R14-21cd]

[R-162cd] If an application has exclusive access, then the device MUST only answer READREG messages coming from the primary application. In this case, READREG messages coming from secondary applications are acknowledged with a status code of GEV_STATUS_ACCESS_DENIED and the *length* field is set to 0 (no data payload). [R14-22cd]

In order for a secondary application not to affect the behavior of a primary application, it is recommended that read operations do not modify the content of a register. Clear-on-read behavior should be avoided, clear-on-write registers are preferable.

16.3.1 READREG_CMD

[R-163c] READREG_CMD MUST follow the layout of Figure 16-5. [R14-23c]

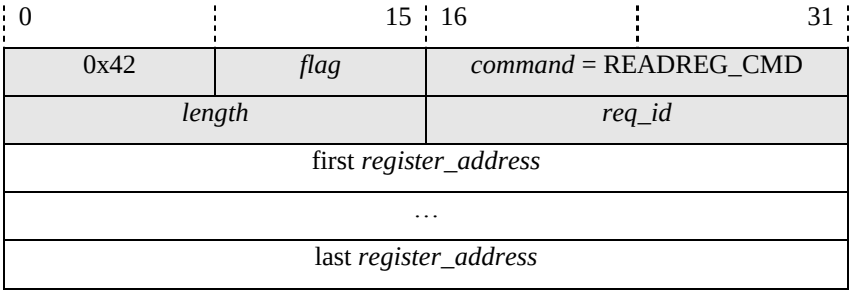


Figure 16-5: READREG_CMD Message

READREG_CMD		
flag	8 bits	ACKNOWLEDGE bit should be set
register_address	32 bits each	Address of the data being requested. It must be aligned on a 32-bit boundary (bit 31 and 30 are cleared). At least one address must be specified.

16.3.2 READREG_ACK

The acknowledge message packet is given by the following figure:

[R-164c] READREG_ACK MUST follow the layout of Figure 16-6. [R14-24c]

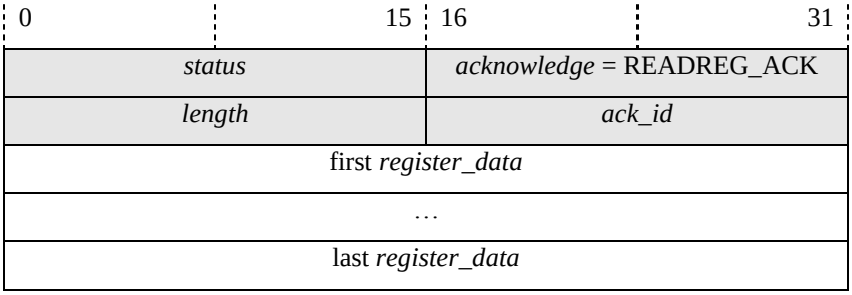


Figure 16-6: READREG_ACK Message

READREG_ACK		
register_data	32 bits each	Value of the data being requested

[R-165cd] For READREG_ACK, acknowledge message length MUST reflect the number of bytes that were successfully read. This length MUST be a multiple of 4 bytes. [R14-25cd]

This way, the application knows immediately from the message *length* field how many bytes were successfully read (it can tell the number of registers in the acknowledge payload). The other requested registers were not read. An appropriate status code indicating the cause of this failure shall be put in the *status* field.

16.4 WRITEREG

[R-166cd] The WRITEREG command MUST be supported by all devices. [R14-26cd]

The application can issue a WRITEREG message to write to a device register. Multiple writes can be concatenated in one message, as long as the total packet size is lower or equal to 576 bytes. Therefore, a maximum of 67 registers can be written with one message if the device supports concatenation of operations (36 bytes are used by the various headers). At least one register must be written to. The number of write operations to perform can be deduced from the *length* field of the command header. It is equal to "*length* / 8".

[R-167cd] The device MUST have a capability flag indicating if multiple writes can be concatenated (bit 31 of GVCP Capability register, at address 0x0934). [R14-27cd]

Note: An application cannot send a concatenated WRITEREG command to the device when this bit is cleared.

[CR-168cd] When concatenation is supported and multiple register writes are specified, they MUST be executed sequentially by the device. [CR14-28cd]

[CR-169cd] When concatenation is supported, if an error occurs, write operation MUST stop at the location of the error. Remaining write operations are discarded and an appropriate error code is returned in the acknowledge message along with the 0-based index of the write operation that failed within the list provided by the command. [CR14-29cd]

The application will thus know which write operations were executed successfully and which one is associated with the status code.

Note: The control protocol assumes the registers are 32-bit wide. It is up to the device to perform a suitable conversion if its internal registers have a different size.

Only the primary application is allowed to send a WRITEREG message with the exception of primary application switchover requests.

[R-170cd] A device MUST answer WRITEREG messages coming from the primary application. [R14-30cd]

[R-171cd] A device MUST return a GEV_STATUS_ACCESS_DENIED status code to any secondary application trying to execute this command unless the device is under control access with switchover enabled and another application, with the right credentials, requests control over the device. [R14-31cd]

When there is no primary application associated to the device, then any application is allowed to write to the CCP register to try to get exclusive or control access (without or with switchover enabled). Once a device is granted exclusive or control access, then no other application can write to CCP. However, it is possible for another application to be granted control over a device when the device was granted control access with switchover enabled as per Section 14.4.

[R-172cd] A device MUST accept WRITEREG_CMD accessing the CCP register from any application when no primary application is registered. This allows the creation of the control channel for the primary application. [R14-32cd]

16.4.1 WRITEREG_CMD

[R-173c] WRITEREG_CMD MUST follow the layout of Figure 16-7. [R14-33c]

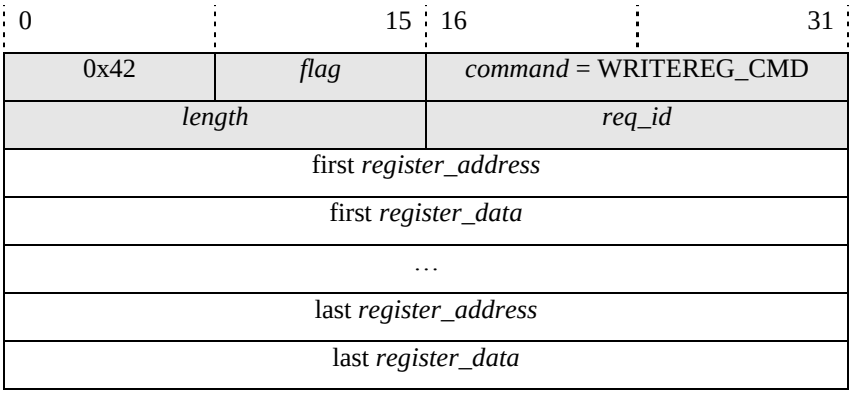


Figure 16-7: WRITEREG_CMD Message

WRITEREG_CMD		
register_address	32 bits each	Address of the data being written. It must be aligned on a 32-bit boundary (bit 31 and 30 are cleared). At least one address must be specified.
register_data	32 bits each	Value of the data to write

16.4.2 WRITEREG_ACK

[R-174c] WRITEREG_ACK MUST follow the layout of Figure 16-8. [R14-34c]

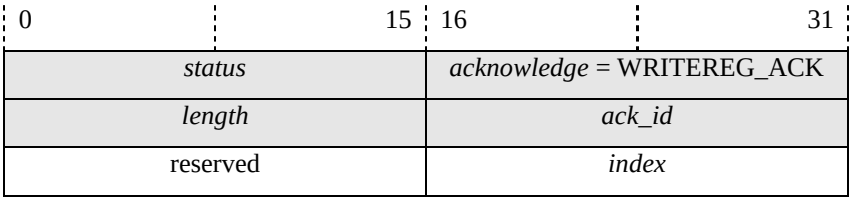


Figure 16-8: WRITEREG_ACK Message

WRITEREG_ACK		
reserved	16 bits	Set to 0 on transmission, ignore on reception.
index	16 bits	On success, this field indicates the number of write operation successfully completed. On failure, this field indicates the index of the register in the list (from 0 to 66) where the error occurred.

When an error occurs during a write operation, the acknowledge message puts the 0-based index of the register where the write error occurred (this index is between 0 and 66); otherwise it is set to the number of write operation successfully completed (with a maximum value of 67).

Write operations before the error are correctly completed by the device. All subsequent write operations after the index in this request are discarded.

[R-175cd] On a WRITEREG failure, an appropriate status code indicating the cause of failure MUST be put in the *status* field of the acknowledge message by the device. [R14-35cd]

16.5 READMEM

[R-176cd] The READMEM command MUST be supported by all devices. [R14-36cd]

This is required to permit a standard procedure to retrieve the device on-board XML description file and to facilitate inter-operability with the GenICam™ specification.

The application can issue a READMEM message to read consecutive 8-bit locations from the device. This could be useful to read strings, such as the XML description file location, or to read an on-board XML description file.

[R-177c] The number of 8-bit locations read by the READMEM command MUST be a multiple of 4. [R14-37c]

[R-178cd] If the number of 8-bit locations to read by the READMEM command is not a multiple of 4, then the device MUST return GEV_STATUS_BAD_ALIGNMENT.

[R-179c] The address specified by the READMEM command MUST be aligned on a 32-bit boundary. If it is not, then the device MUST return GEV_STATUS_BAD_ALIGNMENT. [R14-38c]

The maximum size of memory that can be read by a single READMEM command is 536 bytes (36 bytes are used by the various headers, 4 bytes by the address).

[R-180cd] If no application has exclusive access, then the device MUST answer READMEM messages coming from the primary or from any secondary application. [R14-39cd]

[R-181cd] If an application has exclusive access, then the device MUST only answer READMEM messages coming from the primary application. In this case, READMEM messages coming from secondary applications are acknowledged with a status code of GEV_STATUS_ACCESS_DENIED and the *length* field is set to 0 (no data payload). [R14-40cd]

16.5.1 READMEM_CMD

[R-182c] READMEM_CMD MUST follow the layout of Figure 16-9. [R14-41c]

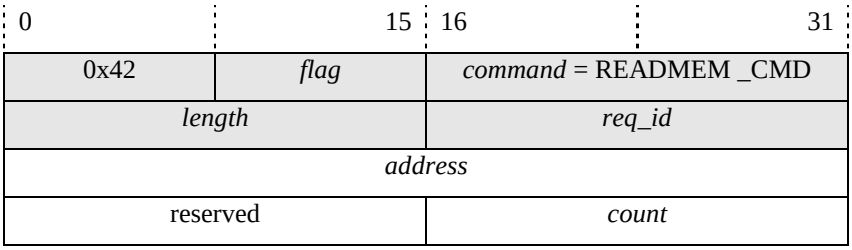


Figure 16-9: READMEM_CMD Message

READMEM_CMD		
flag	8 bits	ACKNOWLEDGE flag (bit 7) should be set.
address	32 bits	Address to read from. Address is automatically incremented for successive data. It must be aligned on a 32-bit boundary (bits 30 and 31 must be cleared).
reserved	16 bits	Set to 0 on transmission, ignore on reception.
count	16 bits	Number of bytes to read from device memory. It must be a multiple of 4 bytes (bits 30 and 31 must be cleared).

16.5.2 READMEM_ACK

[R-183c] READMEM_ACK MUST follow the layout of Figure 16-10. [R14-42c]

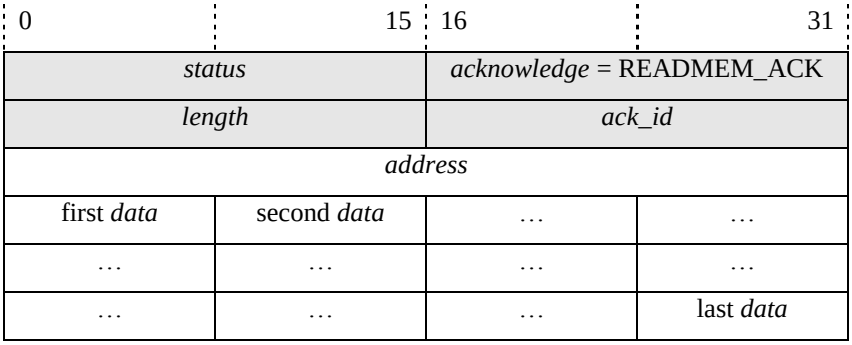


Figure 16-10: READMEM_ACK Message

READMEM_ACK		
address	32 bits	Address where the data was read from. It must be aligned on a 32-bit boundary (bits 30 and 31 must be cleared)
data	8 bits each	8-bit data retrieved from the device registers. Data is copied byte by byte from device memory into this register.

[R-184cd] READMEM is considered successful only if all requested data has been read. Otherwise the device MUST set an appropriate status code and the *length* field of the header MUST be set to 0 (no data payload). This includes the case where the address requested is not supported by the device. [R14-43cd]

16.6 WRITEMEM

[R-185cd] The WRITEMEM command MUST be supported by all devices.

[O-186ca] An application SHOULD check bit 30 of the GVCP Capability register at address 0x0934 to check if WRITEMEM is supported by the device. [O14-44ca]

Note: The preceeding objective is recommended to facilitate interoperability with devices based on GigE Vision version 1.x.

This is required to facilitate inter-operability with the GenICam standard.

The application can issue a WRITEMEM message to write to consecutive 8-bit locations on the device.

[R-187c] The number of 8-bit locations written to by the WRITEMEM command MUST be a multiple of 4. [R14-45c]

[R-188cd] If the number of 8-bit locations to write by the WRITEMEM command is not a multiple of 4, then the device MUST return GEV_STATUS_BAD_ALIGNMENT.

[R-189c] The address specified by the WRITEMEM command MUST be aligned on a 32-bit boundary. If it is not, then the device MUST return GEV_STATUS_BAD_ALIGNMENT. [R14-46c]

The maximum size of memory that can be written by a single WRITEMEM command is 536 bytes (36 bytes are used by the various headers, 4 bytes by the address).

Only the primary application is allowed to send a WRITEMEM message with the exception of primary application switchover requests.

[R-190cd] A device MUST answer WRITEMEM messages coming from the primary application. [R14-47cd]

[R-191cd] A device MUST return a GEV_STATUS_ACCESS_DENIED status code to any secondary application trying to execute this command unless the device is under control access with switchover enabled and another application, with the right credentials, requests control over the device. [R14-48cd]

16.6.1 WRITEMEM_CMD

[R-192c] WRITEMEM_CMD MUST follow the layout of Figure 16-11. [R14-49c]

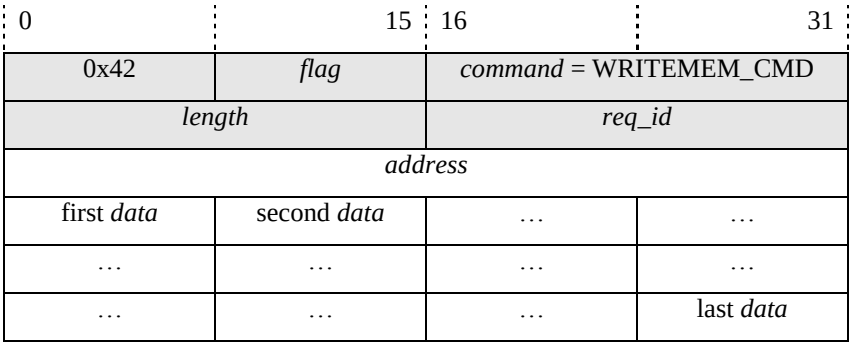


Figure 16-11: WRITEMEM_CMD Message

WRITEMEM_CMD		
address	32 bits	Address of the first data being written. Address is automatically incremented for successive data. It must be aligned on a 32-bit boundary (bits 30 and 31 must be clear)
data	8 bits each	Value of the 8-bit data to write. The number of data to write must be a multiple of 4 bytes.

16.6.2 WRITEMEM_ACK

[R-193c] WRITEMEM_ACK MUST follow the layout of Figure 16-12. [R14-50c]

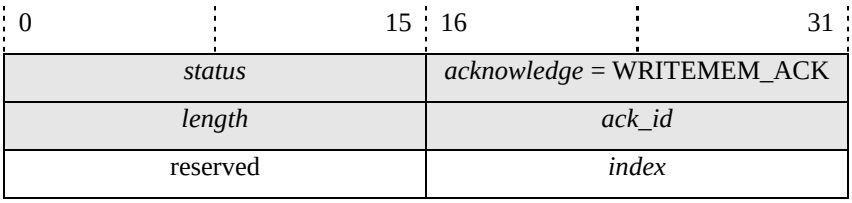


Figure 16-12: WRITEMEM_ACK Message

WRITEMEM_ACK		
reserved	16 bits	Set to 0 on transmission, ignore on reception.
index	16 bits	On success, this field indicates the number of write operations (in bytes) successfully completed. On failure, this field indicates the index of the data in the list (from 0 to 535) where the error occurred.

[R-194cd] When an error occurs during a memory write operation, the acknowledge message MUST put the 0-based index of the data where the write error occurred (this index is between 0 and 535); otherwise it is set to the number of write operations successfully completed. [R14-51cd]

Write operations before the error are correctly completed by the device. All subsequent write operations after the index in this request are discarded.

[R-195cd] When an error occurs during a WRITEMEM operation, an appropriate status code indicating the cause of failure MUST be put in the *status* field of the acknowledge message by the device. [R14-52cd]

16.7 PACKETRESEND

Packet resend can be used (when supported) to request the retransmission of packets that have been lost. This command is different from other GVCP commands since it can be fired asynchronously by the application. Therefore, the application can issue this command even if it is waiting for the acknowledge message from the previous command. This allows fast retransmission of stream data.

[O-196cd] The PACKETRESEND command SHOULD be supported by a device. [O14-53cd]

[R-197ca] An application MUST not ask the device for a PACKETRESEND_CMD unless bit 29 (*PACKETRESEND*) of the GVCP Capability register (at address 0x0934) is set on the device.

Any application (primary and secondary) associated to a GVSP receiver can issue a PACKETRESEND message at any time to tell the device associated to a GVSP transmitter to resend a packet of the data stream, typically when this packet was not received by the GVSP receiver. An application can request any number of sequential packets to be retransmitted for a given block ID. An application cannot request an

acknowledge for this command: the actual stream packet is used to validate reception and processing of the resend request.

[CR-198ca] If the application supports PACKETRESEND, then it MUST clear the ACKNOWLEDGE bit of the GVCP header when requesting a PACKETRESEND. Therefore, the application cannot request an acknowledge. The reception of the requested stream packet is used to validate execution of this command. [CR14-55ca]

[CR-199cd] A device MUST answer PACKETRESEND messages coming from any application, primary and secondary, if the command is supported. [CR14-56cd]

It is up to secondary applications (associated to GVSP receivers) to decide if they want to issue PACKETRESEND or not. This is left as a quality of implementation. A typical scenario is when GVSP transmitters use multicast. The main danger is that many applications might request the same packet to be resent, possibly overloading the device associated to a GVSP transmitter. In this case, the device might receive a large number of resend requests. It is up to the device to decide how to handle duplicated resend requests for the same packet coming from different applications.

By default, the GVSP packet that is retransmitted is sent to the IP address specified by the SCDAx register. This is the destination IP address used by the original GVSP packet. But a device might elect to support an alternative destination that corresponds to the source IP address of the PACKETRESEND_CMD packet. This could be useful when the SCDAx address corresponds to a multicast address and the management entity prefers retransmission to be unicasted to the application that generated the packet resend request. This capability is indicated for each stream channel by the Packet Resend Destination option bit in the SCCx register and it is controlled through the *packet_resend_destination* bit of the SCCFGx register.

16.7.1 PACKETRESEND_CMD

One or multiple packets of a given block ID can be requested for retransmission by the PACKETRESEND command.

[CR-200c] If supported, PACKETRESEND_CMD MUST follow the layout of Figure 16-13. [CR14-59c]

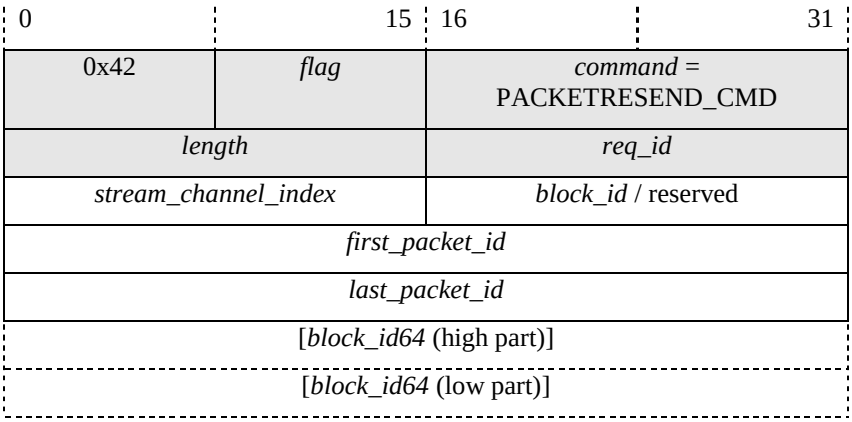


Figure 16-13: PACKETRESEND_CMD Message

PACKETRESEND_CMD		
flag	8 bits	bit 0 to 2 – Reserved, set to 0 on transmission, ignore on reception. bit 3 – Extended ID flag Set to indicate a 64-bit *block_id64* and 32-bit *packet_id32*, Clear for a 16-bit *block_id* and 24-bit *packet_id*. bit 4 to 7 – Use standard definition from GVCP header
stream_channel_index	16 bits	Index of the stream channel (0..511)
block_id / reserved	16 bits	If Extended ID flag = 0 → *block_id* The 16-bit data block ID. For example, this could represent the "frame" index. 0 is reserved for the test packet and cannot be used. If Extended ID flag = 1 → reserved Reserved field. Set to 0 on transmission, ignore on reception.
first_packet_id	32 bits	If Extended ID flag = 0 → first *packet_id* to resend The 24-bit *packet_id* of first packet to resend. MSB must be set to 0. If Extended ID flag = 1 → first *packet_id32* to resend The 32-bit *packet_id32* of the first packet to resend.
last_packet_id	32 bits	If Extended ID flag = 0 → last *packet_id* to resend The 24-bit *packet_id* of the last packet to resend. MSB must be set to 0. It must be equal or greater than *first_packet_id*. If equal, only one packet is resent. Otherwise, all packets starting with *first_packet_id* up to (and including) the *last_packet_id* are resent. When *last_packet_id* is equal to 0xFFFFFF, this indicates to resend all packets up to (and including) the data trailer. This might be useful for variable size block. If Extended ID flag = 1 → last *packet_id32* to resend The 32-bit *packet_id32* of the last packet to resend. It must be equal or greater than *first_packet_id*. If equal, only one packet is resent. Otherwise, all packets starting with *first_packet_id* up to (and including) the *last_packet_id* are resent. When *last_packet_id* is equal to 0xFFFF FFFF, this indicates to resend all packets up to (and including) the data trailer. This might be useful for variable size block.
[*block_id64*]	64 bits	This field is only available when the Extended ID flag is set (bit 3 of the *flag* field). Otherwise these 64 bits are not present in the packet. Same definition as *block_id* above, but with a wider counter to avoid wrap-around.

Note that *last_packet_id* can take a special value (0xFFFFFF for *packet_id*, 0xFFFF FFFF for *packet_id32*) indicating to retransmit all packets from the specified *first_packet_id* up to the data trailer (included).

16.7.2 PACKETRESEND Response

Since the application associated to a GVSP receiver is not allowed to request a packet resend acknowledge on the control channel, PACKETRESEND_ACK does not exist. The validation of execution of packet resend is performed directly on the stream channel.

[CR-201cd] If PACKETRESEND_CMD is supported, then the device MUST never issue the equivalent of a PACKETRESEND_ACK (i.e. return an acknowledge message using the value of PACKETRESEND_CMD + 1) on the control channel. PACKETRESEND_ACK does not exist. [CR14-60cd]

Packet Resend in Standard ID mode

This section applies to a device operating in Standard ID mode (16-bit *block_id* and 24-bit *packet_id*). The behavior in this mode is identical to version 1.2 of this specification.

[CO-202st] If PACKETRESEND is supported and the GVSP transmitter does not operate in Extended ID mode, when the transmitter resends the requested packet or packets, it SHOULD use the GEV_STATUS_PACKET_RESEND status code provided that the application has enabled it to use extended status codes. [CO14-63cd]

Version 1.0 of the specification asked for GEV_STATUS_SUCCESS to be used in the above scenario. So both implementations are acceptable.

[CR-203st] When PACKETRESEND is supported and the GVSP transmitter does not operate in Extended ID mode, if the transmitter is unable to resend the packet data (it might not be available anymore from its memory or the *packet_id* might be out of range for the given *block_id*), then it MUST send a streaming data packet with only the header (no data) and status code set to GEV_STATUS_PACKET_UNAVAILABLE (when the packet data has been discarded), GEV_STATUS_PACKET_NOT_YET_AVAILABLE (when the packet has yet to be acquired), GEV_STATUS_PACKET_AND_PREV_REMOVED_FROM_MEMORY (when the requested packet and all previous ones are not available anymore and have been discarded from the GVSP transmitter memory) or GEV_STATUS_PACKET_REMOVED_FROM_MEMORY (when the requested packet is not available anymore and has been discarded from the GVSP transmitter memory) for each applicable *packet_id*. GEV_STATUS_PACKET_NOT_YET_AVAILABLE, GEV_STATUS_PACKET_AND_PREV_REMOVED_FROM_MEMORY and GEV_STATUS_PACKET_REMOVED_FROM_MEMORY can only be used after the application has enabled GVSP transmitters to use extended status codes. [CR14-61cd]

Note: A GVSP transmitter can use the GEV_STATUS_PACKET_UNAVAILABLE, GEV_STATUS_PACKET_AND_PREV_REMOVED_FROM_MEMORY or GEV_STATUS_PACKET_REMOVED_FROM_MEMORY to notify a GVSP receiver that the packet requested is not available anymore.

Using GEV_STATUS_PACKET_AND_PREV_REMOVED_FROM_MEMORY or GEV_STATUS_PACKET_REMOVED_FROM_MEMORY over GEV_STATUS_PACKET_UNAVAILABLE enables a GVSP transmitter to provide better feedback to a receiver with respect to the reason why a packet is not available anymore. The exact status code to use is left as a quality of implementation and depends of the capabilities of the GVSP transmitter.

Note that in Standard ID mode, the *last_packet_id* can take a special value (0xFFFFFF) indicating to retransmit all packets from the specified *first_packet_id* up to the data trailer (included). The GEV_STATUS_PACKET_UNAVAILABLE, GEV_STATUS_PACKET_NOT_YET_AVAILABLE, GEV_STATUS_PACKET_AND_PREV_REMOVED_FROM_MEMORY or GEV_STATUS_PACKET_REMOVED_FROM_MEMORY error code must be returned for any packet in that range that cannot be resent.

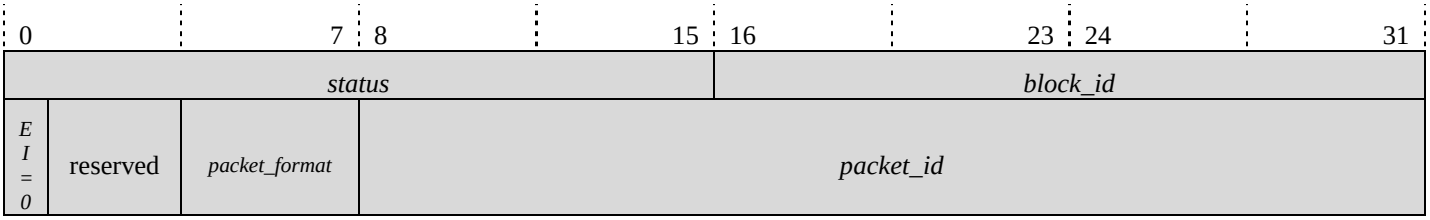


Figure 16-14: GVSP Packet Returning an Error in Standard ID mode

GVSP PACKET returning an error (Standard ID mode)		
status	16 bits	See Figure 24-3 for a description of this field. The content of this field must be one of the status code listed in Table 16-2.
block_id	16 bits	See Figure 24-3 for a description of this field.
EI	1 bit	See Figure 24-3 for a description of this field. This field must be 0 in Standard ID mode.
reserved	3 bits	Set to 0 on transmission, ignore on reception.
packet_format	4 bits	See Figure 24-3 for a description of this field. Content of this field is don't care
packet_id	24 bits	See Figure 24-3 for a description of this field. ID of the packet to which the status code applies.

Version 1.0 of the specification asked for the GEV_STATUS_PACKET_UNAVAILABLE status code to be used when the GVSP transmitter is unable to resend packet data. It didn't make any distinction between the various causes that could have led to a GVSP transmitter not being able to resend a packet. Therefore, a

GEV transmitter should use GEV_STATUS_PACKET_UNAVAILABLE in any of the situations where extended resend status codes are neither supported nor enabled.

Packet Resend in Extended ID mode

This section applies to a device operating in Extended ID mode (64-bit *block_id64* and 32-bit *packet_id32*). The behavior might not be backward compatible with version 1.x of this specification, but can minimize overhead when a range of packets cannot be resent.

[CR-204st] If PACKETRESEND is supported and the GVSP transmitter operates in Extended ID mode, when the transmitter resends the requested packet or packets, then it MUST set the GEV_FLAG_PACKET_RESEND flag (bit 15) of the GVSP packet *flag* field to indicate this packet is the result of a packet resend request.

The above conditional requirement enables the transmission of a status code different than GEV_STATUS_SUCCESS or GEV_STATUS_PACKET_RESEND even for retransmitted GVSP packets, as long as the device operates with Extended ID mode.

In some scenarios, the data cannot be resent due to various reasons. The *status* field of the GVSP packet header is used to provide to receiver with the cause.

[CR-205st] When PACKETRESEND is supported and the GVSP transmitter operates in Extended ID mode, if the transmitter is unable to resend the packet data, then it MUST send a streaming data packet with only the header (no data appended) where the *packet_format* field is "don't care" and provide a suitable status code from Table 16-2. GEV_STATUS_PACKET_TEMPORARILY_UNAVAILABLE, GEV_STATUS_PACKET_NOT_YET_AVAILABLE, GEV_STATUS_PACKET_AND_PREV_REMOVED_FROM_MEMORY and GEV_STATUS_PACKET_REMOVED_FROM_MEMORY can only be used after the application has enabled GVSP transmitters to use extended status codes.

Table 16-2 : Error Status Codes for PACKETRESEND

Status Code	Cause of Error
GEV_STATUS_PACKET_UNAVAILABLE	The requested packet data has been discarded. Use this status code only if none of the following status codes better represent the cause of the error or when appropriate extended status code enable bit is not set.
GEV_STATUS_PACKET_TEMPORARILY_UNAVAILABLE	The requested packet cannot be resent at the moment due to temporary bandwidth issues and should be requested again in the future. Note: In this situation, an additional entry is provided in the payload section of the packet to suggest a hold-off time to the application.
GEV_STATUS_PACKET_NOT_YET_AVAILABLE	The requested packet has yet to be acquired
GEV_STATUS_PACKET_AND_PREV_REMOVED_FROM_MEMORY	The requested packet and all previous ones are not available anymore and have been discarded from the GVSP transmitter memory
GEV_STATUS_PACKET_REMOVED_FROM_MEMORY	The requested packet is not available anymore and has been discarded from the GVSP transmitter memory

Note: A GVSP transmitter can use the GEV_STATUS_PACKET_UNAVAILABLE, GEV_STATUS_PACKET_AND_PREV_REMOVED_FROM_MEMORY or GEV_STATUS_PACKET_REMOVED_FROM_MEMORY to notify a GVSP receiver that the packet requested is not available anymore.

Using GEV_STATUS_PACKET_AND_PREV_REMOVED_FROM_MEMORY or GEV_STATUS_PACKET_REMOVED_FROM_MEMORY over GEV_STATUS_PACKET_UNAVAILABLE enables a GVSP transmitter to provide better feedback to a receiver with respect to the reason why a packet is not available anymore. The exact status code to use is left as a quality of implementation and depends of the capabilities of the GVSP transmitter.

[CO-206st] When PACKETRESEND is supported and the GVSP transmitter operates in Extended ID mode, if the transmitter is unable to resend a range of packet data and all those packet data have the same cause of error, then the device SHOULD:

1. Set the GEV_FLAG_RESEND_RANGE_ERROR flag
2. Use the *packet_id32* field to identify the first packet of the range for which the status code applies.
3. Augment the header by using the *last_packet_id32* field to indicate the last packet of the range for which the status code applies.

The above requirement indirectly asks for a packet with a different resend range (indicated by the *packet_id32* and *last_packet_id32* fields) to be used if the status code is different, since the status code applies to all packets within that range.

In Extended ID mode, it is still permitted to send individual GVSP packet with the error status code, as explained in the previous section for Standard ID mode.

[CR-207sr] If a GVSP receiver supports Packet Resend commands for a range of *packet_id32*, then it MUST support receiving a GVSP packet with the GEV_FLAG_RESEND_RANGE_ERROR flag set.

[CR-208st] When PACKETRESEND is supported and the GVSP transmitter operates in Extended ID mode, if the transmitter is unable to resend the packet data and the status code is set to GEV_STATUS_PACKET_TEMPORARILY_UNAVAILABLE, then the header section is augmented and the last 32 bits of this header MUST contains an hold-off time that can be used by the application to determine the delay to wait before re-issuing a packet resend request.

This additional hold-off data is appended either directly after the generic GVSP header (when the error does not apply to a range) or after the *last_packet_id* field (when the error applies to a range of packets). The receiver must look at the status code and at the *flag* field to determine if these additional fields are appended to the packet. Hence the actual size of the packet depends on the presence or not of these additional fields. And both the *last_packet_id32* and *hold_off/HO_shift* fields are independent. So they might or might not be present at the same time.

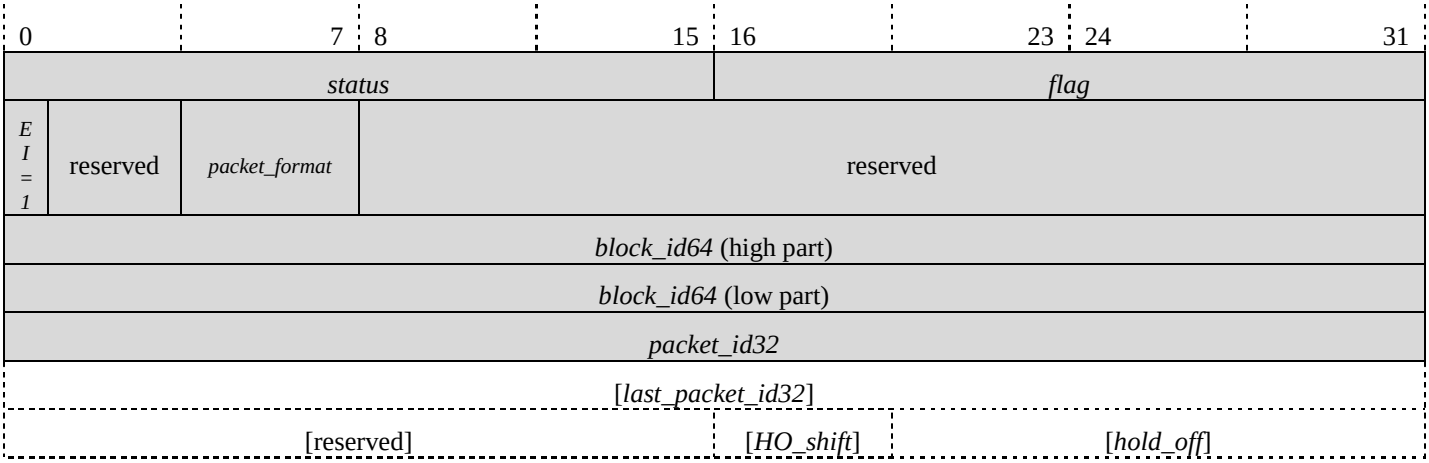


Figure 16-15: GVSP Packet Returning an Error in Extended ID mode

GVSP PACKET returning an error (Extended ID mode)		
status	16 bits	See Figure 24-3 for a description of this field. The content of this field must be one of the status code listed in Table 16-2.
flag	16 bits	See Figure 24-3 for a description of this field. GEV_FLAG_RESEND_RANGE_ERROR flag is set when a range of packets are combined in a single packet with the given status code. For that case, the *last_packet_id32* field is present. This is only available in Extended ID mode.
EI	1 bit	See Figure 24-3 for a description of this field. This field must be 1 when the GEV_FLAG_RESEND_RANGE_ERROR flag is set. That is the device must be operated in Extended ID mode in order to use that status code.
reserved	3 bits	Set to 0 on transmission, ignore on reception.
packet_format	4 bits	See Figure 24-3 for a description of this field. Content of this field is don't care
reserved	24 bits	See Figure 24-3 for a description of this field.
block_id64	64 bits	See Figure 24-3 for a description of this field.
packet_id32	32 bits	See Figure 24-3 for a description of this field. ID of the first packet to which the status code applies.
[*last_packet_id32*]	32 bits	ID of the last packet in the range that could not be retransmitted. This field is only present when the GEV_FLAG_RESEND_RANGE_ERROR flag is set. That is, packet is larger by 4 bytes when GEV_FLAG_RESEND_RANGE_ERROR is set.
[reserved]	16 bits	Set to 0 on transmission, ignore on reception. This field is only present when status = GEV_STATUS_PACKET_TEMPORARILY_UNAVAILABLE.

[*HO_shift*]	4 bits	Represents the number of bits to shift left the hold-off time to accommodate a wider range of value. This field is only present when status = GEV_STATUS_PACKET_TEMPORARILY_UNAVAILABLE.
[*hold_off*]	12 bits	Represents the hold-off time measured in microseconds. This hold-off time can be used as a recommendation from the device to represent the amount of time the application should wait before asking for retransmission. This field is only present when status = GEV_STATUS_PACKET_TEMPORARILY_UNAVAILABLE.

16.7.3 Packet Resend handling on the GVSP receiver side

Since the amount of memory on the GVSP transmitter side is limited, it is important for the application associated to a GVSP receiver to send any PACKETRESEND command as fast as possible to ensure the missing data is still available from the GVSP transmitter memory. Otherwise, the GVSP transmitter will return GEV_STATUS_PACKET_UNAVAILABLE, GEV_STATUS_PACKET_AND_PREV_REMOVED_FROM_MEMORY or GEV_STATUS_PACKET_REMOVED_FROM_MEMORY because the data has been lost (possibly overwritten by newer data).

There is a special case for linescan cameras which might be triggered at a variable rate. This specification provides the GEV_STATUS_PACKET_NOT_YET_AVAILABLE error code to announce to GVSP receivers that a particular data packet has yet to be acquired and made available for transmission. This might be the case when the line trigger rate is momentarily slower than the timeout used by the packet resend logic on the GVSP receivers side.

Handling of PACKETRESEND message by the GVSP receiver and its application is left as a quality of implementation. One must remember that UDP packets may arrive out of order at their destination. Therefore, simply looking for consecutive packet ID is not a bulletproof solution to handles all situations.

Some network topologies guarantee that UDP packets will always arrive in order. This is the case if the GVSP transmitter is directly connected to the PC hosting a GVSP receiver using a cross-over cable. Other more complex network topologies, especially those offering multiple routes for the UDP packet to travel from source to destination (using gateways and routers), cannot guarantee UDP packets will arrive in sequential order.

When UDP packets are guaranteed to arrive in order, a GVSP receiver and its application can use the packet ID to track down the sequence of packets. If a packet ID is skipped, then GVSP receiver can command its application to request right away the missing packet. A timeout can be used to detect if the data trailer is missing.

When UDP packets are not guaranteed to arrive in order, a GVSP receiver and its application cannot assume packet ID values are sequential. Therefore, the packet resend mechanism cannot react as fast as for the other case. In this situation, many schemes can be implemented, some using timeouts. It is left as a quality of implementation for a GVSP receiver and its application to offer a mechanism that suits the uncertainty introduced by packet re-ordering.

16.8 PENDING

The PENDING acknowledge is an optional message introduced with version 1.1 of this specification.

When PENDING acknowledge generation is enabled, it is issued to indicate the processing of a command will take a longer period to execute. The application can therefore adjust its acknowledge timeout value. The Pending Timeout bootstrap register (at address 0x0958) indicates the maximum amount of time command execution can take before the PENDING_ACK is issued. If PENDING_ACK is not supported, then the Pending Timeout register provides the worst-case command execution time for a single command (not considering concatenated commands).

[CR-209cd] If PENDING message is supported, PENDING_ACK MUST not be issued for DISCOVERY_CMD, FORCEIP_CMD and PACKETRESEND_CMD. [CR14-64cd]

[CR-210cd] If PENDING message is supported, PENDING_ACK MUST be issued by the device before command execution exceeds the value reported by the Pending Timeout register. [CR14-65cd]

PENDING_ACK will actually extend execution time for a given command. It is important to ensure this does not impact the device enumeration process performed by other applications on the network.

[CR-211cd] If PENDING message is supported, a device MUST answer DISCOVERY_CMD even if a PENDING_ACK is in progress. [CR14-66cd]

It is recommended for the device to issue the PENDING_ACK as soon as it knows command execution is going to exceed the value indicated by the Pending Timeout register. This enables better command timeout management on the application side.

[CR-212cd] If PENDING message is supported, PENDING_ACK will temporarily suspend the heartbeat timeout counter between the time PENDING_ACK is issued and the time the command ACK is sent. [CR14-67cd]

The previous conditional requirement is necessary since the application cannot issue the next command before the PENDING_ACK timeout has expired. This PENDING_ACK timeout might be longer than the heartbeat timeout period.

16.8.1 PENDING_ACK

The PENDING_ACK should be issued as soon as the device knows the request execution exceeds the maximum reported value indicated by the Pending Timeout register.

[CR-213c] If PENDING message is supported, PENDING_ACK MUST follow the layout of Figure 16-16. [CR14-68c]

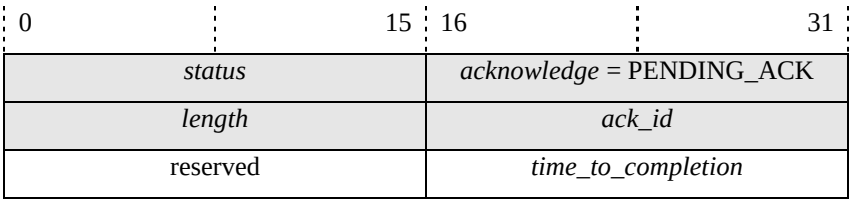


Figure 16-16: PENDING_ACK Message

PENDING_ACK		
reserved	16 bits	Set to 0 on transmission, ignore on reception.
time_to_completion	16 bits	Number of ms to complete the pending request. The application ACK timeout should be set to a value larger or equal to this value.

[CR-214cd] If PENDING_ACK is supported, the *ack_id* field MUST be identical to the *req_id* field of the command that generated the PENDING_ACK. [CR14-69cd]

[CR-215cd] If PENDING_ACK is supported, the *time_to_completion* field MUST indicate the time in ms to complete command execution. If the device is unable to complete execution within this time, it is allowed to issue another PENDING_ACK to extend the command ACK timeout even further. There is no limit to the number of PENDING_ACK re-issue. [CR14-70cd]

The *time_to_completion* field is 16 bits, leading to a maximal timeout of 65 seconds. Upon reception of the PENDING_ACK, the application resets the acknowledge timeout to the value indicated by this message, plus some margin to allow for network latencies (quality of implementation).

16.9 ACTION

The ACTION command may optionally be supported by a device.

[O-216ca] An application SHOULD check bit 25 (*ACTION*) of the GVCP Capability register at address 0x0934 to check if ACTION is supported by the device. [O14-71ca]

[O-217ca] An application SHOULD check bit 14 (*scheduled_action_command*) of the GVCP Capability register at address 0x0934 to check if Scheduled Action Commands are supported by the device.

ACTION command can be unicasted or broadcasted to a device in order to trigger various actions. This can be used to simultaneously issue the same action to multiple devices residing on the same subnet. The information provided in the request provides mechanism for the device to determine if it is part of the list of devices required to act on this request. If the device must react, the device will decide which action(s) to trigger in accord with the characteristics of the request. It is possible to delay the execution of the action at a specific time in the future by providing an optional action time.

[CR-218cd] If ACTION message is supported, then the device MUST process ACTION_CMD requests coming from the primary or any secondary application. [CR14-72cd]

16.9.1 ACTION_CMD

[CR-219c] If supported, ACTION_CMD must follow the layout of Figure 16-17. [CR14-73c]

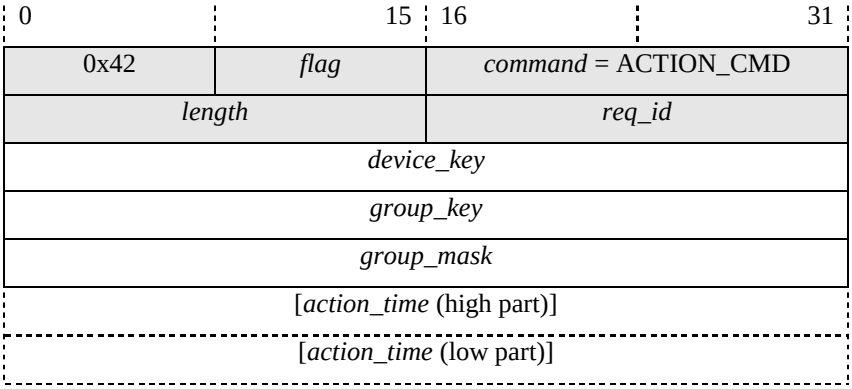


Figure 16-17: ACTION_CMD Message

ACTION_CMD		
flag	8 bits	bit 0 – Action time available (Scheduled Action Commands) 0: *action_time* field is not available, GVCP packet is 20 bytes. 1: *action_time* field (64 bits) is available, GVCP packet is 28 bytes. bit 1 to 3 – Reserved. Set to 0 on transmission, ignore on reception. bit 4 to 7 – Use standard definition from GVCP header
device_key	32 bits	Key each sender of an ACTION_CMD has to embed into the packet to prevent erroneous execution of the associated action The *device_key* has to match the internal Action Device Key register (at address 0x090C) to let a device accept the ACTION_CMD
group_key	32 bits	Key defining a group of devices on which actions have to be executed. *group_mask* can be used to filter out some of these devices from the group.
group_mask	32 bits	The *group_mask* is compared against the mask associated to the given action number (ACTION_GROUP_MASKx) • 0x00000000 - reserved • 0xFFFFFFFF - override configured mask in all devices. Indicates that any device with a non nil ACTION_GROUP_MASKx for the given action will assert the signal provided that requirements of Table 14-1 are also met. • Any other value – to match against the configured value in the device
[*action_time*]	64 bits	This field is only present for Scheduled Action Commands when a future action time is specified. Check bit 0 of the *flag* field. Time when to assert the action signal. Must be in same time domain as the Timestamp Value register.

This command can be unicasted or broadcasted to the device by the primary or any secondary applications.

[CR-220cd] If supported, an ACTION_CMD with an action time MUST be queue for execution when the device timestamp equals the specified action time.

[CR-221cd] If the device supports Scheduled Action Commands, then if the time tag of the action command is in the past (relative to the device timestamp) and it is a valid action command, then:

1. The device MUST execute this action command immediately (no queuing) and return a GEV_STATUS_ACTION_LATE status code if an ACK was requested.

2. The device SHOULD fire a GEV_EVENT_ACTION_LATE event if the message channel is opened and this event generation has been enabled.

16.9.2 ACTION_ACK

The ACTION_ACK is only sent back if the 4 conditions to execute the ACTION are met. Otherwise, the device shall ignore the ACTION_CMD request.

[CR-222cd] If ACTION_CMD is supported, then a device MUST return an ACTION_ACK message only if an acknowledge is requested and the 4 conditions provided in Table 14-1 are respected by the ACTION_CMD packet. [CR14-74cd]

[CR-223cd] If ACTION_CMD is supported, then when the action is a Scheduled Action Command, the ACTION_ACK message MUST be sent as soon as the action command is queued for execution at a future time.

The above requirement is necessary not to block the control channel for an indeterminate period of time.

[CR-224c] If supported, the ACTION_ACK message must follow the layout of Figure 16-18. [CR14-75c]

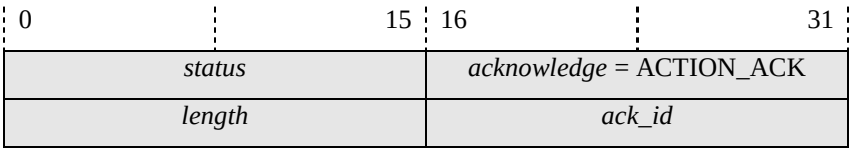


Figure 16-18: ACTION_ACK Message

17 Message Channel Dictionary

Messages in this category are sent on the Message Channel (if one exists). An application can verify if a message channel is available by reading the Number of Message Channels register at address 0x0900. The device always initiates the transaction on the message channel.

[O-225ca] An application SHOULD check bits 27 (*EVENTDATA*) and 28 (*EVENT*) of the GVCP Capability register at address 0x0934 to check if EVENT or EVENTDATA commands are supported by the device before opening a message channel. [O15-15ca]

[CR-226cd] If Message Channel is available, a device MUST offer a way to enable/disable the generation of event messages. [CR15-1cd]

This is typically done using enable register bits. The exact way to do this is left as a quality of implementation. But this specification suggests that each event or group of events could be enabled or disabled individually by writing into enable registers. Each bit of an enable register could represent one event or a group of related events.

[CO-227cd] The registers used to enable event generation SHOULD be provided in the XML device description file. [CO15-2cd]

17.1 EVENT

[O-228cd] The EVENT message SHOULD be supported by a device. [O15-3cd]

The device may use EVENT messages to notify the application that asynchronous events have occurred. Multiple events can be concatenated in one message, as long as the total packet size is lower or equal to 576 bytes. Therefore, the maximum number of events in one packet is 33 when using 16-bit *block_id* and 22 for 64-bit *block_id64*. It is up to the device to decide if multiple events are concatenated or sent as separate packets (this is independent of the concatenation flag, bit 31 (*concatenation*) of GVCP Capability register at address 0x0934). For 16-bit *block_id*, the number of events in the message is equal to the header "*length* field / 16". For 64-bit *block_id64*, the number of events in the message is equal to the header "*length* field / 24".

[CR-229cd] If EVENT message is supported, each EVENT command MUST be tagged with a 64-bit *timestamp* representing the time at which the event was generated on the device. The frequency of this timestamp is available from the Timestamp Tick Frequency registers at address 0x093C and 0x0940. No timestamp is available if this register is set to 0. In this case, the *timestamp* field of the message MUST be set to 0. [CR15-5cd]

[CO-230cd] This protocol reserves event identifier from 0 to 36863 for GigE Vision usage. Other event identifiers are available for device-specific events and SHOULD be described in the XML device description file when EVENT message is supported. [CO15-6cd]

Note that events from 32769 to 36863 directly maps to standard GigE Vision status code. This can be used by a device to asynchronously report an error.

17.1.1 EVENT_CMD

[CR-231c] If EVENT message is supported, EVENT_CMD MUST follow the layout of Figure 17-1. [CR15-7c]

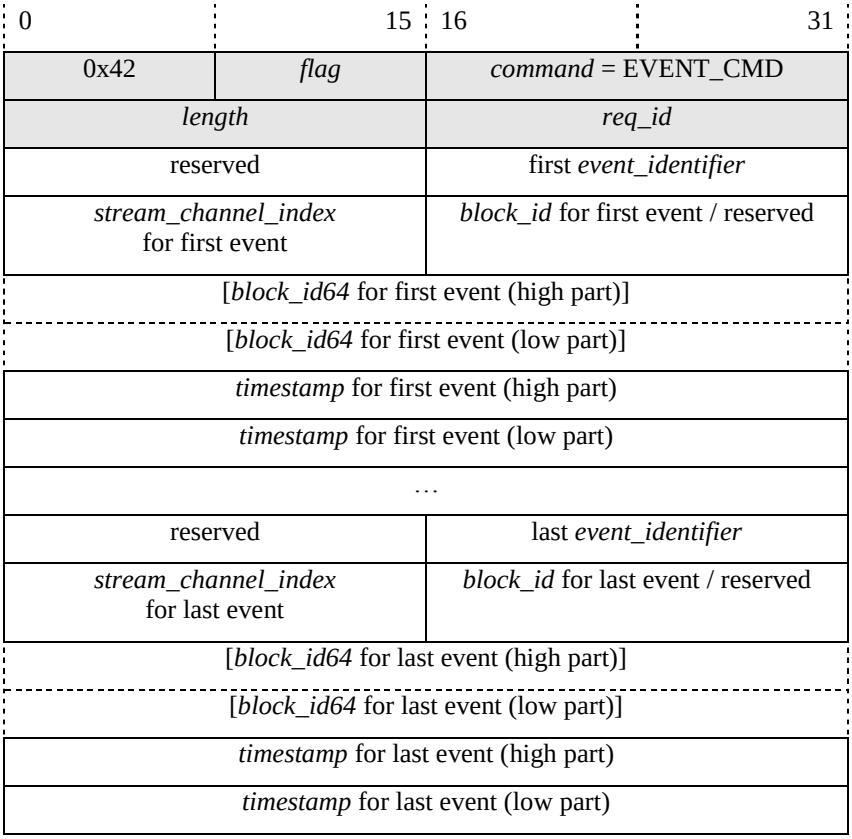


Figure 17-1: EVENT_CMD Message

EVENT_CMD		
flag	8 bits	bit 0 to 2 – Reserved, set to 0 on transmission, ignore on reception. bit 3 – Extended ID flag Set to indicate a 64-bit *block_id64* and 32-bit *packet_id32*, Clear for a 16-bit *block_id* and 24-bit *packet_id*. bit 4 to 7 – Use standard definition from GVCP header
reserved	16 bits	Set to 0 on transmission, ignore on reception.
event_identifier	16 bits	event number as defined by the GigE Vision specification for value between 0 and 36863 or the XML device description file for value between 36864 and 65535
stream_channel_index	16 bits	Index of stream channel associated with this event. 0xFFFF if no stream channel involved.

block_id / reserved	16 bits	If Extended ID flag = 0 → *block_id* ID of the data block associated to this event. 0 if no *block_id* associated to this event. If Extended ID flag = 1 → reserved Reserved field. Set to 0 on transmission, ignore on reception.
[*block_id64*]	64 bits	This field is only available when the Extended ID flag is set (bit 3 of the *flag* field). Otherwise these 64 bits are not present in the packet (packet is thus shorter by 8 bytes for each event concatenated in this message). Same definition as *block_id* above, but with a wider counter to avoid wrap-around.
timestamp	64 bits	64-bit timestamp generated by the device to indicate when this event happened. 0 if timestamp is not supported. **Note**: The GenICam specification supports 64-bit signed integer while this field is 64-bit unsigned. Therefore, only 63 bits are available through the GenICam interface.

17.1.2 EVENT_ACK

A device is not required to request an acknowledge message.

[CR-232c] If EVENT message is supported, EVENT_ACK MUST follow the layout of Figure 17-2. [CR15-8c]

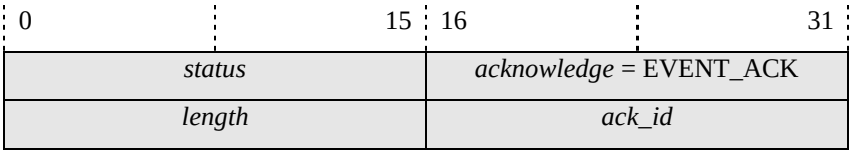


Figure 17-2: EVENT_ACK Message

17.2 EVENTDATA

The EVENTDATA message may be supported by a device.

The device may use EVENTDATA messages to notify the application that asynchronous events have occurred. The main difference from the EVENT message is that device-specific data can be attached to this message. The total packet size must be lower or equal to 576 bytes. ~~Therefore, device-specific data is limited to 540 bytes.~~

[CR-233cd] If EVENTDATA message is supported, each EVENTDATA command MUST be tagged with a 64-bit timestamp representing the time at which the event was generated on the device. The frequency of this timestamp is available from the Timestamp Tick Frequency registers at address 0x093C and 0x0940. No timestamp is available if this register is set to 0. In this case, the *timestamp* field of the message is set to 0. [CR15-10cd]

[CR-234cd] It is not possible to concatenate multiple events into an EVENTDATA command when this command is supported. A device MUST only put a single event in the EVENTDATA message when this message is supported. [CR15-11cd]

The event identifiers are the same as the ones associated to the EVENT message. This protocol reserves event identifier from 0 to 36863 for GigE Vision usage. Other event identifiers are available for device-specific events and should be described in the XML device description file.

[CO-235cd] The content of the device-specific data associated to EVENTDATA SHOULD be specified in the XML device description file when this command is supported. [CO15-12cd]

17.2.1 EVENTDATA_CMD

[CR-236c] If EVENTDATA message is supported, EVENTDATA_CMD MUST follow the layout of Figure 17-3. [CR15-13c]

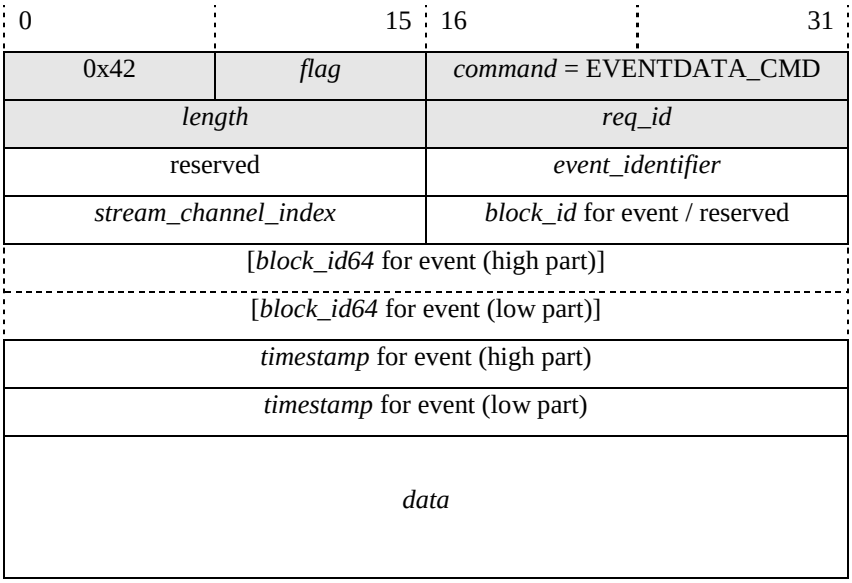


Figure 17-3: EVENTDATA_CMD Message

EVENTDATA_CMD		
flag	8 bits	bit 0 to 2 – Reserved, set to 0 on transmission, ignore on reception. bit 3 – Extended ID flag Set to indicate a 64-bit *block_id64* and 32-bit *packet_id32*, Clear for a 16-bit *block_id* and 24-bit *packet_id*. bit 4 to 7 – Use standard definition from GVCP header
reserved	16 bits	Set to 0 on transmission, ignore on reception.
event_identifier	16 bits	event number as defined by the GigE Vision specification for value between 0 and 36863 or the XML device description file for value between 36864 and 65535
stream_channel_index	16 bits	Index of stream channel associated with this event. 0xFFFF if no stream channel involved.

block_id / reserved	16 bits	If Extended ID flag = 0 → *block_id* ID of the data block associated to this event. 0 if no *block_id* associated to this event. If Extended ID flag = 1 → reserved Reserved field. Set to 0 on transmission, ignore on reception.
[*block_id64*]	64 bits	This field is only available when the Extended ID flag is set (bit 3 of the *flag* field). Otherwise these 64 bits are not present in the packet (packet is thus shorter by 8 bytes). Same definition as *block_id* above, but with a wider counter to avoid wrap-around.
timestamp	64 bits	64-bit timestamp generated by the device to indicate when this event happened. 0 if timestamp is not supported. **Note**: The GenICam specification supports 64-bit signed integer while this field is 64-bit unsigned. Therefore, only 63 bits are available through the GenICam interface.
data	up to packet size	Raw data. Specified by the XML device description file.

17.2.2 EVENTDATA_ACK

A device is not required to request an acknowledge message.

[CR-237c] If EVENTDATA message is supported, EVENTDATA_ACK MUST follow the layout of Figure 17-4. [CR15-14c]

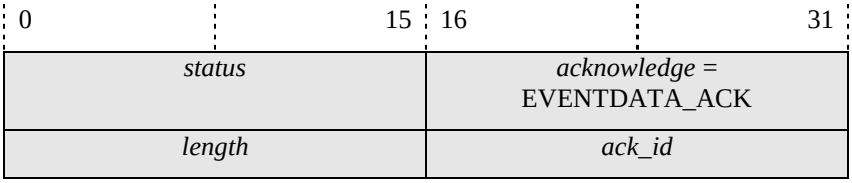


Figure 17-4: EVENTDATA_ACK Message

18 Command and Acknowledge Values

The following table lists the numerical value associated with each message defined in this specification. The channel type indicates on which channel the message is transmitted.

This specification reserves command and acknowledge values from 0 to 32767 for GVCP (most-significant bit cleared). Other values, from 32768 to 65535 (most-significant bit set), are available for device-specific messages.

Note that the acknowledge message associated to a command message always have the value of the corresponding command message plus one. Therefore, command messages have their least-significant bit cleared while acknowledge messages have their least-significant bit set.

[R-238c] Device and application MUST use the following value to represent the various messages. Notice the value of the acknowledge message is always the associated command message + 1 (when such a command message exists), with PACKETRESEND_CMD and PENDING_ACK being exceptions. [R16-1c]

Message	Channel	Value
Discovery Protocol Control		
DISCOVERY_CMD	Control	0x0002
DISCOVERY_ACK	Control	0x0003
FORCEIP_CMD	Control	0x0004
FORCEIP_ACK	Control	0x0005
Streaming Protocol Control		
PACKETRESEND_CMD	Control	0x0040
Invalid. Resent packet must be on the stream channel	Control	0x0041
Device Memory Access		
READREG_CMD	Control	0x0080
READREG_ACK	Control	0x0081
WRITEREG_CMD	Control	0x0082
WRITEREG_ACK	Control	0x0083
READMEM_CMD	Control	0x0084
READMEM_ACK	Control	0x0085
WRITEMEM_CMD	Control	0x0086
WRITEMEM_ACK	Control	0x0087
PENDING_ACK	Control	0x0089
Asynchronous Events		
EVENT_CMD	Message	0x00C0

EVENT_ACK	Message	0x00C1
EVENTDATA_CMD	Message	0x00C2
EVENTDATA_ACK	Message	0x00C3
Miscellaneous		
ACTION_CMD	Control	0x0100
ACTION_ACK	Control	0x0101

19 Status Code

This section lists the various status codes that can be returned in an acknowledge message or a GVSP header. This specification defines two categories of status code:

1. Standard status code
2. Device-specific status code

Standard status codes are defined in this specification. Refer to appendix 2 for additional information about supporting those codes.

[O-239cd] A device SHOULD return the status code that provides as much information as possible about the source of error. [O18-1cd]

[R-240cd] If a device cannot return a more descriptive status code, the device MUST at least return the GEV_STATUS_ERROR status code. [R18-2cd]

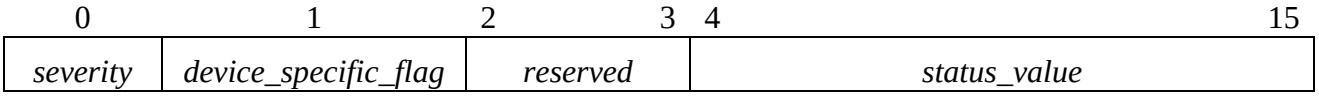
[O-241st] A GVSP transmitter SHOULD return the status code that provides as much information as possible about the status of streaming activities. [O18-3st]

For instance, a memory or data overrun error in the GVSP transmitter should lead to a data trailer packet sent with a *status* field set to GEV_STATUS_DATA_OVERRUN.

[R-242st] If a GVSP transmitter cannot return a more descriptive status code upon the occurrence of a streaming error, it MUST at least return the GEV_STATUS_ERROR status code. [R18-4st]

A device or GVSP transmitter is not required to support all status codes. Some status codes are suited to the application.

Status register is mapped as follows:



severity	0 = info, 1 = error
device_specific_flag	0 for standard GigE Vision status codes, 1 for device-specific status codes
reserved	Set to 0 on transmission, ignore on reception.
status_value	Actual value of status code

Table 19-1: List of Standard Status Codes

Status Value	Description	Value (hexadecimal)
GEV_STATUS_SUCCESS	Command executed successfully	0x0000
GEV_STATUS_PACKET_RESEND[1]	Only applies to packet being resent. This flag is preferred over the GEV_STATUS_SUCCESS when the GVSP transmitter sends a resent packet. This can be used by a GVSP receiver to better monitor packet resend. Note that bit 15 of the GVSP packet *flag* field must be set for a retransmitted GVSP packet when *block_id64* are used.	0x0100
GEV_STATUS_NOT_IMPLEMENTED	Command is not supported by the device	0x8001
GEV_STATUS_INVALID_PARAMETER	At least one parameter provided in the command is invalid (or out of range) for the device	0x8002
GEV_STATUS_INVALID_ADDRESS	An attempt was made to access a non-existent address space location.	0x8003
GEV_STATUS_WRITE_PROTECT	The addressed register cannot be written to	0x8004
GEV_STATUS_BAD_ALIGNMENT	A badly aligned address offset or data size was specified.	0x8005
GEV_STATUS_ACCESS_DENIED	An attempt was made to access an address location which is currently/momentary not accessible. This depends on the current state of the device, in particular the current privilege of the application.	0x8006
GEV_STATUS_BUSY	A required resource to service the request is not currently available. The request may be retried at a later time.	0x8007
Deprecated	Not used starting with GigE Vision 2.0. Consult GigE Vision 1.2 for a detailed list.	0x8008 to 0x0800B
GEV_STATUS_PACKET_UNAVAILABLE	The requested packet is not available anymore.	0x800C
GEV_STATUS_DATA_OVERRUN	Internal memory of GVSP transmitter overrun (typically for image acquisition)	0x800D
GEV_STATUS_INVALID_HEADER	The message header is not valid. Some of its fields do not match the specification.	0x800E
Deprecated	Not used starting with GigE Vision 2.0. Consult GigE Vision 1.2 for a detailed list.	0x800F
GEV_STATUS_PACKET_NOT_YET_AVAILABLE[1]	The requested packet has not yet been acquired. Can be used for linescan cameras device when line trigger rate is slower than application timeout.	0x8010
GEV_STATUS_PACKET_AND_PREV_REMOVED_FROM_MEMORY[1]	The requested packet and all previous ones are not available anymore and have been discarded from the GVSP transmitter memory. An application associated to a GVSP receiver should not request retransmission of these packets again.	0x8011

Status Value	Description	Value (hexadecimal)
GEV_STATUS_PACKET_REMOVED_FROM_MEMORY[1]	The requested packet is not available anymore and has been discarded from the GVSP transmitter memory. However, applications associated to GVSP receivers can still continue using their internal resend algorithm on earlier packets that are still outstanding. This does not necessarily indicate that any previous data is actually available, just that the application should not just assume everything earlier is no longer available.	0x8012
GEV_STATUS_NO_REF_TIME[1]	The device is not synchronized to a master clock to be used as time reference. Typically used when Scheduled Action Commands cannot be scheduled at a future time since the reference time coming from IEEE 1588 is not locked.	0x8013
GEV_STATUS_PACKET_TEMPORARILY_UNAVAILABLE[1]	The packet cannot be resent at the moment due to temporary bandwidth issues and should be requested again in the future	0x8014
GEV_STATUS_OVERFLOW[1]	A device queue or packet data has overflowed. Typically used when Scheduled Action Commands queue is full and cannot take the addition request.	0x8015
GEV_STATUS_ACTION_LATE	The requested scheduled action command was requested at a time that is already past. Only available starting with GEV 2.0 when an *action_time* is specified.	0x8016
GEV_STATUS_ERROR	Generic error. Try to avoid and use a more descriptive status code from list above.	0x8FFF

[1]: This extended status code must be used only when appropriate extended status codes are enabled in the GVCP Configuration register.

20 Events

The following table lists the standard events defined by this specification.

[CR-243cd] When a message channel is supported, event identifier with data MUST be sent using EVENTDATA command (if this command is supported). [CR19-1cd]

Note: Events related to image acquisition have been deprecated in version 1.1 of this specification. The proper definition of events is managed by the GenICam™ Standard Features Naming Convention for GigE Vision.

Table 20-1: List of Events

Event Identifier	Description	Data	Value (hexadecimal)
Deprecated	Obsolete. Defined by a previous version of the specification.	-	0x0000 to 0x0006
GEV_EVENT_PRIMARY_APP_SWITCH	Primary application switchover has been granted.	None	0x0007
GEV_EVENT_LINK_SPEED_CHANGE	Indicates that the link speed has changed.	None	0x0008
GEV_EVENT_ACTION_LATE	Execution of a Scheduled Action Command was late	None	0x0009
Reserved	-	-	0x000A to 0x8000
GEV_EVENT_ERROR_xxx	An asynchronous error occurred on the device. The value of the event identifier represents the status code for the asynchronous error. Refer to the list of status code	None	0x8001 to 0x8FFF
Device-specific Event	Refer to the XML device description file.	Check device documentation	0x9000 to 0xFFFF

21 ICMP

[R-244cd] A device MUST minimally support the subset of ICMP provided in the Table 21-1 on all its supported network interfaces. [R20-1cd]

Each ICMP message is defined to be either mandatory or optional. Only “Echo Reply”, “Echo” and “Destination Unreachable” are mandatory.

For instance, a device must support the ‘ping’ command for packet size up to 576 bytes. Therefore, the device must correctly receive an ICMP Echo message and must correctly transmit an ICMP Echo Reply message.

Note: The Destination Unreachable ICMP message must be managed according to the recommendation of the RFC. This message might result from a routing transient, so it is a hint, not a proof, that the specified destination is unreachable. RFC1122 provides additional details about Destination Unreachable. This RFC provides recommendation on the action to take upon the reception of this message based on the error code embedded into it. For instance, a connection must not be aborted if a soft error is received while it should in the case of the reception of hard errors. If a device decides to close its channels due to a communication error, it must follow [R-072cd].

Table 21-1: ICMP Messages

Message code	Message description	Device reception	Device transmission
0	Echo Reply	Optional	Mandatory (for packets size up to 576 bytes), not required for packet size larger than 576.
3	Destination Unreachable	Mandatory	Optional
4	Source Quench	Optional	Optional
5	Redirect	Optional	Optional
8	Echo	Mandatory (for packets size up to 576 bytes), not required for packet size larger than 576.	Optional
9	Router Advertisement	Optional	Optional
10	Router Solicitation	Optional	Optional
11	Time Exceeded	Optional	Optional
12	Parameter Problem	Optional	Optional
13	Timestamp	Optional	Optional
14	Timestamp Reply	Optional	Optional
15	Information Request	Optional	Optional
16	Information Reply	Optional	Optional

17	Address Mask Request	Optional	Optional
18	Address Mask Reply	Optional	Optional

PART 3 – GVSP

22 GVSP Summary

22.1 Overview

GVSP is an application layer protocol relying on the UDP transport layer protocol. It allows a GVSP receiver to receive image data, image information or other information from a GVSP transmitter.

GVSP packets always travel from a GVSP transmitter to a receiver.

The current version on the specification uses UDP IPv4 as the transport layer protocol. Because UDP is unreliable, GVSP provides mechanisms to guarantee the reliability of packet transmission (through GVCP) and to ensure minimal flow control.

22.2 Goals

The goals for GigE Vision Streaming Protocol (GVSP) are:

- Define a mechanism for a GVSP transmitter to send image data, image status or other data to a GVSP receiver.
- Support self-describing data.
- Minimize IP stack complexity in the GigE Vision entities.
- Minimize the network bandwidth overhead.

22.3 Scope

This section provides:

- Specification of the stream protocol.
- Headers for the various packet types.
- List of pixel formats.

23 GVSP Transport Protocol Considerations

[R-245s] The stream protocol MUST use UDP with IPv4 transport protocol. [R22-1s]

[R-246s] GVSP transmitters and receivers MUST NOT use any IP options in the IP datagram for GVSP. This way, the IP header size is fixed at 20 bytes. This is to allow efficient hardware implementation of UDP/IP offload engines. [R22-2s]

23.1 UDP

UDP is a connectionless protocol that adds no reliability, flow-control or error recovery to IP. Because of this lack of functionality, GVSP imposes restrictions on how stream channels are handled.

[R-247sr] GVSP receivers MUST be able to accommodate packets arriving out of order because UDP cannot guarantee the order of packet delivery. [R22-3sr]

23.1.1 Fragmentation

Since a stream channel always transmits from a GVSP transmitter to a GVSP receiver, it is up to the receiver to decide if it can support IP fragmentation. The SCPSx of GVSP transmitters provides a *do_not_fragment* bit that is copied into the IP header of each packet of the corresponding stream channel. The SCCx capability register can be used by GVSP receivers to advertise if IP reassembly is supported. It is up to the management entity of the system to make sure that GVSP transmitters are configured according to the capabilities of the GVSP receivers. In case of doubts, IP fragmentation should not be used.

23.1.2 Packet Size Requirements

Contrary to the GVCP, the GVSP packet size is not required to be a multiple of 32 bits: it has a byte resolution. The packet size to use can be determined by firing test packets to find the MTU of the transmission medium. Refer to the SCPSx bootstrap register and to the GevSCPSPacketSize feature.

Packet size requested through SCPSx refers to data payload packets, not to data leader and data trailer. The data leader and data trailer must be contained within a 576 bytes packet.

23.1.3 Reliability and Error Recovery

UDP is inherently an unreliable transport protocol. GVSP provides reliability by monitoring the *packet_id/packet_id32* field and validating that all packets composing a block have been received. If packets are missing, then the primary application associated to GVSP receivers can request them using the PACKETRESEND command (when this command is supported by the GVSP transmitter). *packet_id/packet_id32* is incremented by one for successive packets.

UDP checksum support is not mandatory.

The following figure shows a typical packet exchange when a packet is lost. Note the time it takes for the GVSP receiver and its application to react to a missing packet is implementation dependent. Remember that UDP does not guarantee packets are delivered in the same order they were transmitted by the GVSP transmitter.

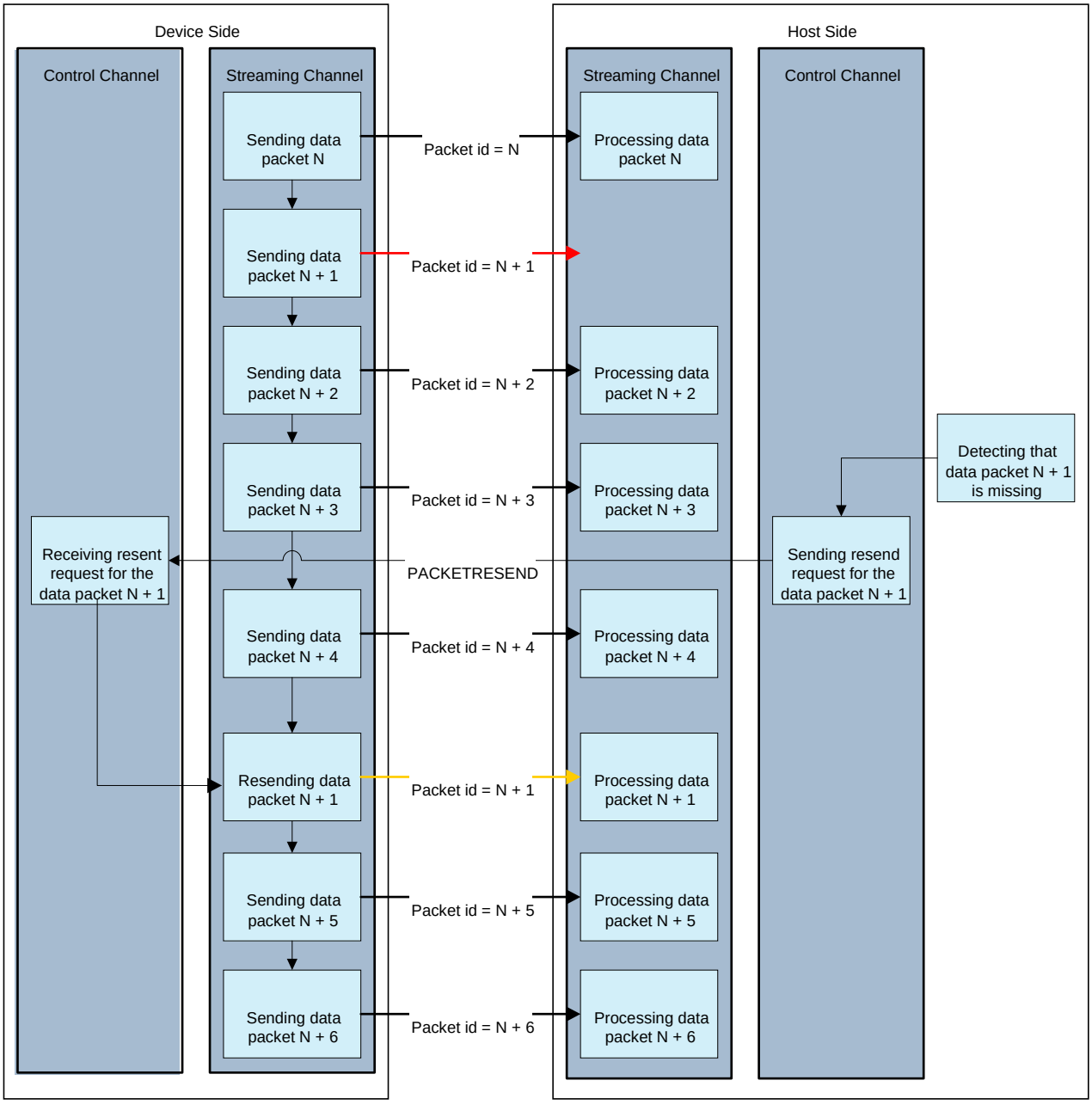


Figure 23-1: Data Resend Flowchart

23.1.4 Flow Control

The GigE Vision specification defines mechanisms to optimize the efficiency of bandwidth utilization via the control of the packet size being used when streaming data. It also provides a crude flow control mechanism via the control of the inter-packet delay. The SCPDx register exposes the inter-packet *delay* value: it indicates the minimal amount of time (in timestamp counter unit) to wait between the transmission on a given physical link of two streaming packets associated to a stream channel. This amount of time is measured from the end of the first packet to the start of the next packet. The timestamp counter is typically

the timer with the finest granularity on the device. If there is no timestamp counter, then SCPDx cannot be used to manage flow control.

These mechanisms can be used to try to reduce the number of overflows and packet resends issued by an application associated to its GVSP receivers. However, it can be appreciated that these mechanisms might not be sufficient to efficiently pace the data exchange between a GigE Vision transmitter and receivers because of current Ethernet controller design.

A number of Ethernet controllers have very little local memory for temporary storage of received packets. This local memory is used to store packets locally before they can be transferred by a DMA transaction to the host networking stack. System conditions exist that can prevent the Ethernet controller from transferring packets from this local memory to the networking stack. If these conditions persist, then the local memory becomes insufficient and packet loss occurs. This can take place even if the utilized Ethernet bandwidth is low.

For this reason, in many cases, standard Ethernet flow control can be used to prevent this scenario from occurring. Before receive local memory runs out, the MAC sublayer (inside the Ethernet controller) can issue a MAC Control request. One of these requests is the PAUSE request. It can be sent to inhibit data transmission for a period of time. During the PAUSE period, the latter Ethernet controller should have enough time to transfer the remaining packets from local memory to the networking stack avoiding the above scenario. Going forward to 10 Gbps speed, this scenario might become more prevalent thus the need for flow control.

This specification refers to flow control as defined in the following sections of the IEEE 802.3-2008 specification, namely:

- Clause 31 (MAC Control): This clause refers to an optional MAC Control sublayer that provides real-time control and manipulation of MAC sublayer operation.
- Annex 31A (MAC Control opcodes): This annex lists the defined opcodes for the MAC Control sublayer.
- Annex 31B (MAC Control PAUSE operation): This annex describes the MAC Control PAUSE request.

Referring to the OSI reference model, this mechanism is handled at the Data Link layer and may be transparent to the layers above. Thus GigE Vision control, stream and message channels may transparently benefit from this. It should be noted that it may be needed to propagate back pressure information to upper layers. For instance, a GVSP transmitter may need to start dropping images on a frame boundary should its network interface has been notified to pause data transmission in order to prevent data overrun at the layer where GVSP resides in the protocol stack.

Given the fact that entities with multiple interfaces might have different capabilities from one interface to the other, it is required to implement PAUSE capabilities and control on a per network interface basis.

Note: For link aggregation, all physical MAC part of the same aggregator use the same PAUSE configuration since they are presented as a single logical link.

The following table presents the mapping between the *PAUSE_reception* and *PAUSE_generation* configuration bits of the Network Interface Capability and Network Interface Configuration registers, and the PAUSE and ASM_DIR bits advertised by the network interface during link negotiation.

Table 23-1 : PAUSE Negotiation

***PAUSE_reception* Enable**	***PAUSE_generation* Enable**	**PAUSE**	**ASM_DIR**
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	1

This table indicates that a GigE Vision network interface will never present itself as an interface that is only supporting symmetric PAUSE. From a link negotiation standpoint, a GigE Vision network interface that is only interested to receive pause frames would present itself as both a symmetric PAUSE partner and as an asymmetric PAUSE partner capable to receive PAUSE frames (PAUSE = 1 and ASM_DIR = 1). In the latter case, the network interface will be able to receive PAUSE frame from symmetric and asymmetric PAUSE link partners and doesn't have to generate any PAUSE frames.

It is desirable for an application to be able to query statistics on the number of PAUSE frames transmitted or received on a per network interface basis. In order to do so, one can query the recommended (but optional) aPAUSEMACCtrlFramesTransmitted and aPAUSEMACCtrlFramesReceived attributes of the oMACControlFunctionEntity managed object class of the MAC of the network interface. These 2 attributes are only 2 of many attributes of a given managed object class (and there are also many managed object classes). These attributes should be available through the XML device description file. Alternatively, they could be retrieved using other standard methods (such as SNMP) if supported by the GigE Vision device.

23.1.5 End-to-End Connection

Connections are managed using GVCP. GVCP heartbeat monitors the presence of device and application.

GVSP provides no provision for end-to-end connection. If the application is disconnected, the GVCP heartbeat timer will expire and automatically will reset the streaming connection of GVSP transmitters unless they have been configured to continue to stream as per Section 11.10. By default, if the device is disconnected, the GVSP receivers will stop to receive streaming packets from the GVSP transmitters associated to that device.

23.1.6 Device error handling during acquisition and transmission

[R-248st] If an acquisition error occurs in the GVSP transmitter, the transmitter might not be able to send all data packets for that data block. But if it sent at least one packet for a given data block, then it MUST send the corresponding data trailer with an appropriate status code. [R22-4st]

One of the most typical case such error occur is when data acquisition occurs faster than stream packets can be transmitted. In this situation, some packets are lost inside the GVSP transmitter before they had the chance to be transmitted on the stream channel. For instance, this might be the case if the link speed is lower than expected, if the inter-packet delay is too long or if too many IEEE 802.3 PAUSE frame are sent on the link. But if a full data block is lost, then the above requirement does not apply.

Two scenarios must be managed:

1) Data packets are lost within a block

This situation occurs when the data leader has been transmitted, but some data payload packets are discarded.

[R-249st] When a data payload packet is discarded due to insufficient internal memory in the GVSP transmitter, then the transmitter MUST set the corresponding data trailer *status* field to GEV_STATUS_DATA_OVERRUN.

The decision to transmit or not remaining data payload packets in this data block is left as a quality of implementation. But in accordance with [R-248st], GVSP transmitter is not allowed to discard the data trailer.

2) Data blocks are lost

This situation happens when one or more data blocks are dropped internally, including the data leader. Hence no corresponding data leader has been transmitted to the GVSP receiver, but that data block was acquired and expected by the GVSP receiver.

[R-250st] When a GVSP transmitter discards a full data block that was acquired, then the transmitter MUST count this block in the *block_id/block_id64* counter and it MUST set the GEV_FLAG_PREVIOUS_BLOCK_DROPPED flag bit of the next valid data leader transmitted on the link when the Extended ID (EI) flag is set in the GVSP header.

On the following valid data leader transmitted, the GVSP transmitter puts the GEV_FLAG_PREVIOUS_BLOCK_DROPPED flag and uses the same *block_id/block_id64* as if all blocks were transmitted. This provides a mechanism for the GVSP receiver to determine how many data blocks were lost by computing the gap in the *block_id/block_id64* count.

24 Data Block

The data block is divided into a number of elements to provide the GVSP receiver with the information necessary to decode the block. A Data Leader starts the data block and provides information about the payload type of the block. This leader can also provide additional information to help in decoding the block. The Data Leader is followed by the Data Payload which contains the actual data to be streamed. Finally, the data block is completed by a Data Trailer indicating the end of the data block. Within the data block, Data Payload packets must be sent sequentially (*packet_id/packet_id32* increments by 1 from one packet to the next).

Data blocks are sent sequentially on a given stream channel.

[R-251st] A GVSP transmitter MUST send data blocks sequentially. It is not allowed to send the data leader of the next block before the data trailer of the current block has been transferred. Only exception is for PACKETRESEND which might request any packet(s) in the current or in a previous data block. [R23-2st]

For images, multiple ROIs from the same frame can be provided in two different ways:

1) **on a single stream channel -** When they are transmitted on the same stream channel, each ROI must be transmitted with a different block ID. In this case, identical timestamps must be used to facilitate matching the ROI to a given exposure, as specified by [CR-253st]. A device might elect to send all ROIs on the same stream channel to simplify the transmission (only one stream channel to configure).
2) **on different stream channels** – When they are transmitted on different stream channels (one ROI per stream channel), then the ROIs should use the same block ID value to facilitate matching. In this case, the timestamps can still be used to facilitate matching the ROI to a given exposure. A device might elect to use different stream channels to allow different ROIs to be sent to different destination.

In either method, the data leader correctly reflects ROI position within the original image.

[CR-252st] If an image source supports multiple ROI, then overlapping data from multiple ROIs MUST be retransmitted with each ROI so that each image is provided completely in its block. [CR24-10st]

[CR-253st] If a camera device supports multiple ROI, then different ROIs from the same exposure MUST use the same timestamp. [CR24-11st]

A special situation occurs with an interlaced image source (or its generalization: a multi-field image source). For these sources, this specification treats each field as a separate data block. Additional information is provided in the Data Leader packet to enable the GVSP receiver to determine the number of fields and the field index of the data block within the image. Using that information, the receiver can de-interlace the image if necessary. Note that it is possible for a GVSP transmitter to de-interlace the image before transmission so it appears as progressive scan from the GVSP receiver's perspective.

[CR-254st] If a GVSP transmitter provides a mode to transmit interlaced image data, then it MUST support a de-interlaced transmission mode.

Note 1: If a GigE Vision 1.x GVSP receiver ignores the bits related to interlacing in the reserved field of the Image Data Leader packets, then it will consider each block as a progressive scan image composed of one field from the transmitter.

Note 2: GVSP is not reporting field ordering. Field order configuration is left as a de-interlacer control parameter.

24.1 Data Block Transmission Modes

The data block is put in packets to allow transmission on the stream channel. Two transmission modes are supported: Standard and All-in Transmission modes.

Standard Transmission Mode

In Standard Transmission mode, the Data Leader, Data Payload and Data Trailer are separated in different packets. This is the standard approach supported since the inception of GigE Vision. The leader and trailer delimit the data payload and offer clear separation points between consecutive data blocks.

Three different packet types exist in Standard Transmission mode:

1. Data Leader Packet
2. Data Payload Packet
3. Data Trailer Packet

The figure below illustrates the sequence of packets in Standard Transmission mode.

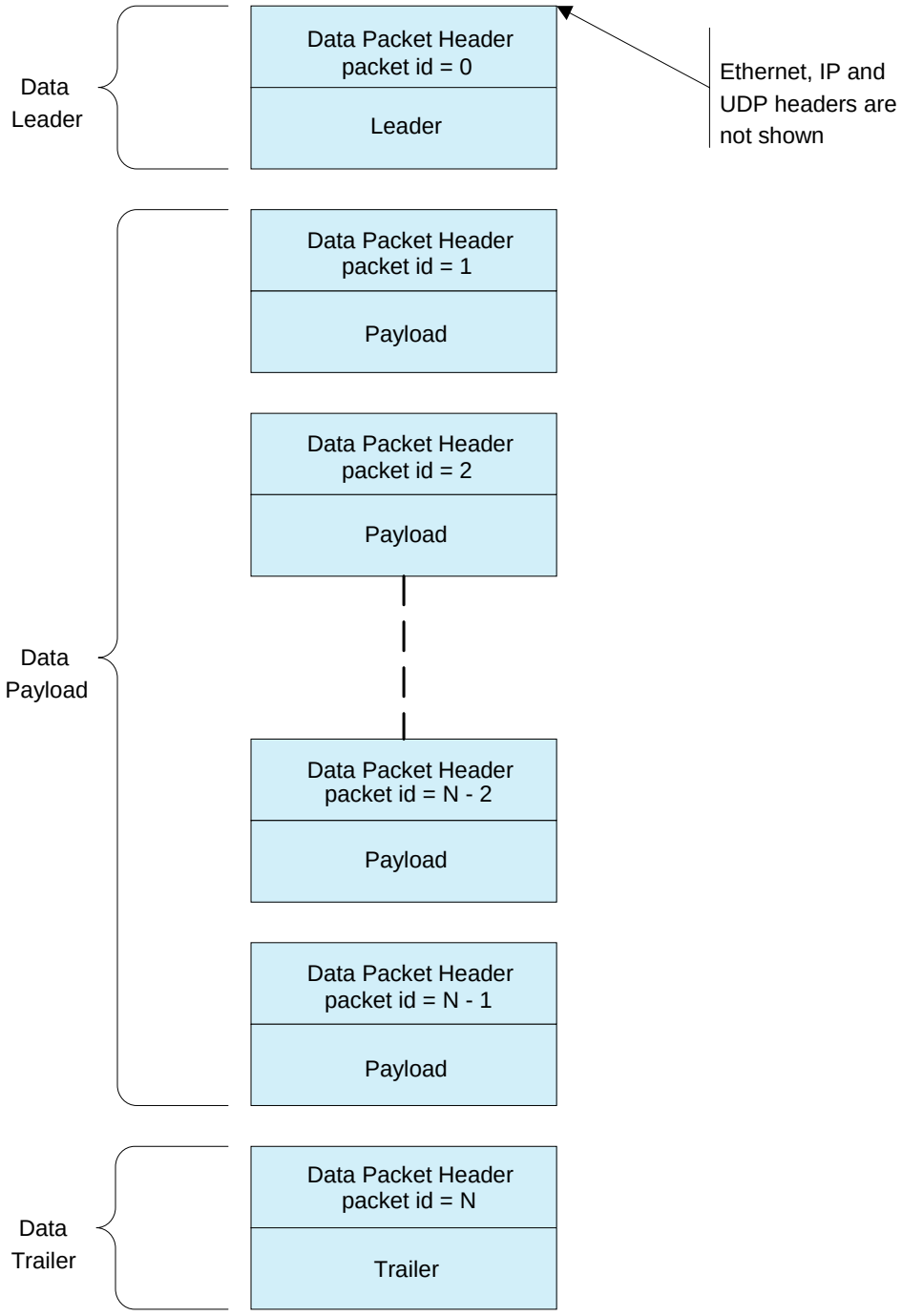


Figure 24-1: Data Block – Standard Transmission Mode

[R-255s] GVSP transmitters and GVSP receivers MUST support the Standard Transmission mode.

[R-256s] For Standard Transmission mode, the data block MUST contain three elements:

1. **Data Leader packet**: One packet signaling the beginning of a new data block.
2. **Data Payload packet**: One or more packets containing the actual information to be streamed for the current data block.
3. **Data Trailer packet**: One packet signaling the end of the data block. [R23-1s]

All-in Transmission Mode

Optionally, if the Data Leader, Data Payload and Data Trailer can fit into a single packet, then the GVSP transmitter can be configured to regroup them. This is the All-in transmission mode that can be used to reduce the overhead of having separate Data Leader and Data Trailer packets. An application must verify if this transmission mode is supported (bit 29 of SCCx) and enable it (bit 29 of SCCFGx), otherwise the standard transmission mode is used.

Note that to facilitate the task of the GVSP receiver, the offset at which the Data Payload starts is fixed by having both the Data Leader and Data Trailer in front of the Data Payload in the packet.

A single packet type exists for All-in Transmission mode: the All-in packet.

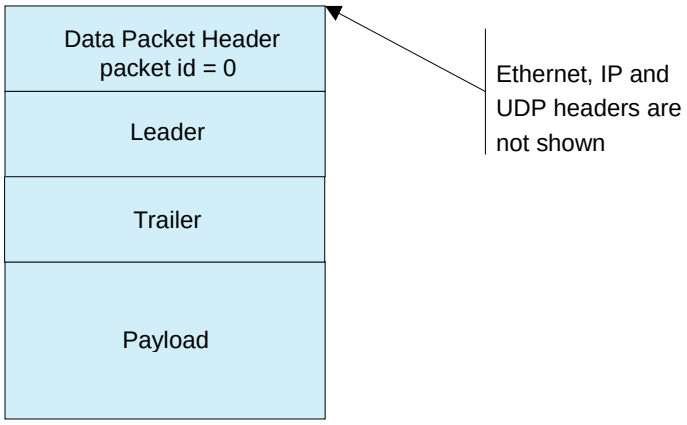


Figure 24-2: Data Block – All-in Transmission Mode

GVSP transmitters and GVSP receiver MAY support the All-in Transmission mode. This mode will typically be supported on transmitters that need to stream very short data block.

24.2 Data Block Packet Header

All GVSP packets share the same basic header. The GVSP protocol header does not include a *length* field: GVSP receivers use the UDP length information to determine the packet size.

For the standard transmission mode, with the exception of the last data packet which may be smaller to match the data payload size, all Data Payload packets are the same size. However, it is also possible to pad the last data packet so that all Data Payload packets are exactly the same size (unless specified otherwise by the payload type).

The management entity of the system is responsible to optimize packet size. This could be achieved using the test packet for instance. This facilitates the positioning of information by the GVSP receiver in the buffer if a packet is missing.

[R-257s] All GVSP headers MUST use network byte order (big-endian). [R24-2s]

[R-258s] The data packet header illustrated in Figure 24-3 MUST be available on all GVSP packets. [R24-1s]

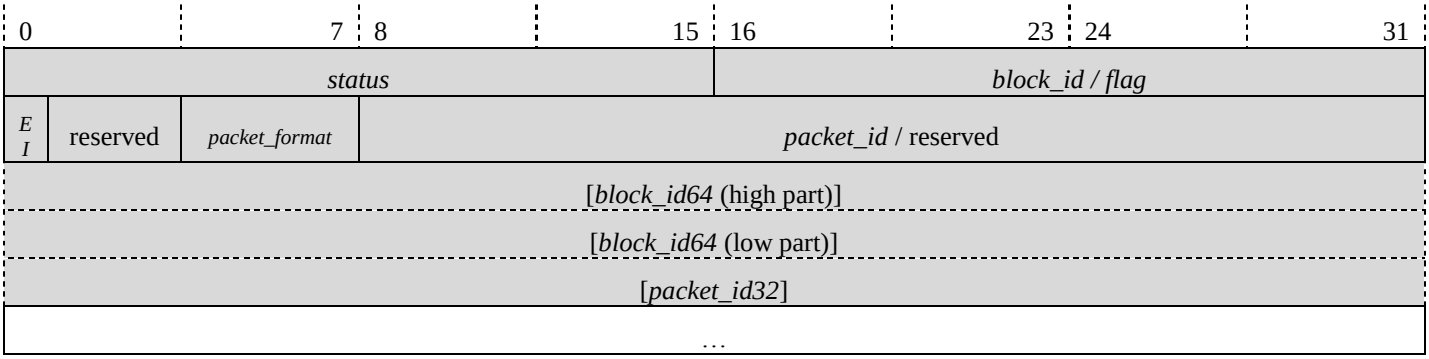


Figure 24-3: GVSP Packet Header

GVSP PACKET HEADER		
status	16 bits	Status of the streaming operation (see status code section)
block_id / flag	16 bits	When EI flag = 0 → *block_id* ID of the data block. Sequential and incrementing starting at 1. A *block_id* of 0 is reserved for the test packet. *block_id* wraps-around to 1 when it reaches its maximum value. For a GVSP transmitter, the *block_id* is reset to 1 when the stream channel is opened. When EI flag = 1→ *flag* Provide GVSP stream status flag information. Upper 8 bits (msb, bit 0 to bit 7) are device-specific. Lower 8 bits (lsb, bit 8 to bit 15) are specified by GigE Vision. All unused bits must be set to 0 on transmission and ignored on reception. bit 0 to 7: device-specific bit 8 to 12: Reserved. See Table 24-1. bit 13: GEV_FLAG_RESEND_RANGE_ERROR. See Table 24-1. bit 14: GEV_FLAG_PREVIOUS_BLOCK_DROPPED. See Table 24-1. bit 15: GEV_FLAG_PACKET_RESEND. See Table 24-1.

EI	1 bit	Extended ID (EI) flag indicating if block IDs are 64 bits and packet IDs are 32 bits. This is necessary to correctly decode this packet and determine if the block ID is put in the *block_id* or in the *block_id64* field, and if the packet ID is put in the *packet_id* or in the *packet_id32* field. It is also used to indicate how *block_id / flag* field is used. 0: Standard ID - 16-bit block ID using the *block_id* field and 24-bit packet ID in the *packet_id* field. *block_id64* and *packet_id32* fields are not present. 1: Extended ID - 64-bit block ID using the *block_id64* field and 32-bit packet ID using the *packet_id32* field. The *flag* field is available.
reserved	3 bits	Set to 0 on transmission, ignore on reception.
packet_format	4 bits	For Standard Transmission Mode: • DATA_LEADER_FORMAT (=1): The packet contains a data leader. • DATA_TRAILER_FORMAT (=2): The packet contains a data trailer. • DATA_PAYLOAD_FORMAT: The packet contains a part of the data payload. A number of payload format are supported: ○ (=3): Generic data payload format ○ (=5): H.264 data payload format ○ (=6): Multi-zone Image data payload format For All-in Transmission Mode: • ALL_IN _FORMAT (=4): The packet contains the data leader, data trailer and all data payload. The data block must be contained in a single packet. All other values are reserved.
packet_id	24 bits	When *EI* flag = 0 → *packet_id* ID of packet in the block. The *packet_id* is reset to 0 at the start of each data block. *packet_id* 0 is thus the data leader for the current *block_id*. When *EI* flag = 1 → reserved Reserved field. Set to 0 on transmission, ignore on reception.
[*block_id64*]	64 bits	Same definition as *block_id* above, but with a wider counter to avoid wrap-around. This field is only available when the *EI* flag is set. Otherwise these 64 bits are not present in the packet. Check the *EI* flag.
[*packet_id32*]	32 bits	Same definition as the *packet_id* above, but with a wider counter to support larger data blocks. This field is only available when the *EI* flag is set. Otherwise these 32 bits are not present in the packet. Check *the* EI flag.

In order to enable a GVSP receiver to determine if data blocks have been lost at the beginning of a new connection, it is necessary for the GVSP transmitter to reset the *block_id/block_id64* field so it starts with a known value.

[R-259st] A GVSP transmitter MUST reset the *block_id* (or *block_id64*) associated to a stream channel to a value of 1 when this stream channel is opened.

24.2.1 GVSP Status Flags

When the EI (Extended ID) bit is set in the GVSP packet header, the *flag* field is available and can report the following status.

Table 24-1 : List of GVSP Status Flags

Status Value	Description	bit
Device-specific	Defined by the manufacturer of the device.	0 – 7
Reserved	Set to 0 on transmission, ignore on reception.	8 – 12
GEV_FLAG_RESEND_RANGE_ERROR	Flag indicating that the GVSP packet carries error condition for a range of packets, as specified in [CR-203st].	13
GEV_FLAG_PREVIOUS_BLOCK_DROPPED	Set to 1 when one or more of the previous data blocks have been discarded. Look at the *block_id/block_id64* to determine how many blocks have been dropped	14
GEV_FLAG_PACKET_RESEND	Set to 1 when the current packet is the result of a PACKETRESEND request.	15

24.3 Standard Transmission Mode Packets

The following packets are defined for Standard Transmission mode.

24.3.1 Data Leader Packet

[R-260st] The data leader packet MUST be the first packet of the block. [R24-3st]

[R-261st] The data leader packet MUST be sent in a separate packet with *packet_id*/*packet_id32* set to 0. It MUST fit in one packet of 576 bytes at most (including the IP, UDP and GVSP header). [R24-4st]

[R-262s] The data leader packet MUST follow the layout of Figure 24-4.

For different payload types, additional information might be appended to this header.

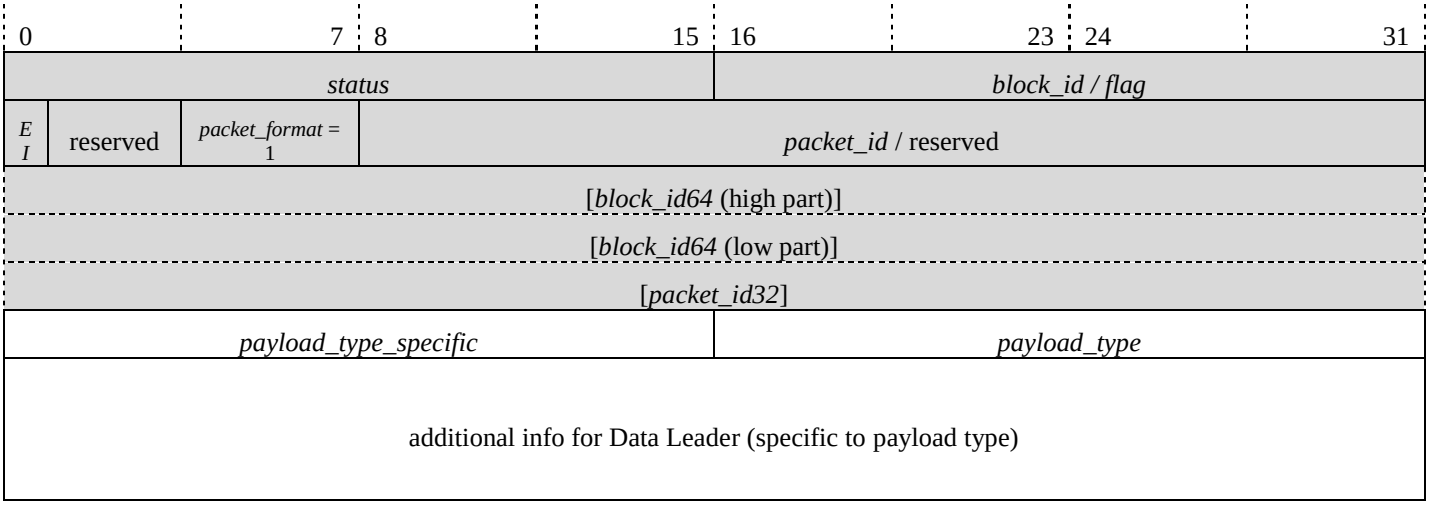


Figure 24-4: Generic Data Leader Packet

DATA LEADER PACKET		
payload_type_specific	16 bits	This field is specific to each payload type. Look at the payload type definition for details.
payload_type	16 bits	Unique ID identifying the type of data block we can expect to receive. Refer to Table 25-1 for the list of supported payload type 16-bit values. bit 0 (msb) is used to indicate a device-specific payload type. bit 1 is used to report Extended Chunk mode for certain payload types, as explained in Section 25.1. 0: Extended Chunk mode is disabled 1: Extended Chunk mode is enabled

24.3.2 Data Payload Packet

Data payload packets transport the actual information to a GVSP receiver. There might be up to 16 million data payload packets per block (for 24-bit *packet_id* field).

The last data payload packet of a block might have a smaller size than the other payload packets since total amount of data to transfer is not necessarily a multiple of the packet size.

[R-263st] All data payload packets except the last one MUST use the requested *packet_size* from the SCPSx register. Note that this requirement does not fully apply to the H.264 (see section 25.9) and to the multi-zone image payload types (see section 25.10). [R24-12st]

[R-264st] Data payload packets must be sent with a sequential packet ID that must be incremented by one from one packet to the next.

[R-265s] The data payload packet header MUST follow the layout of Figure 24-5

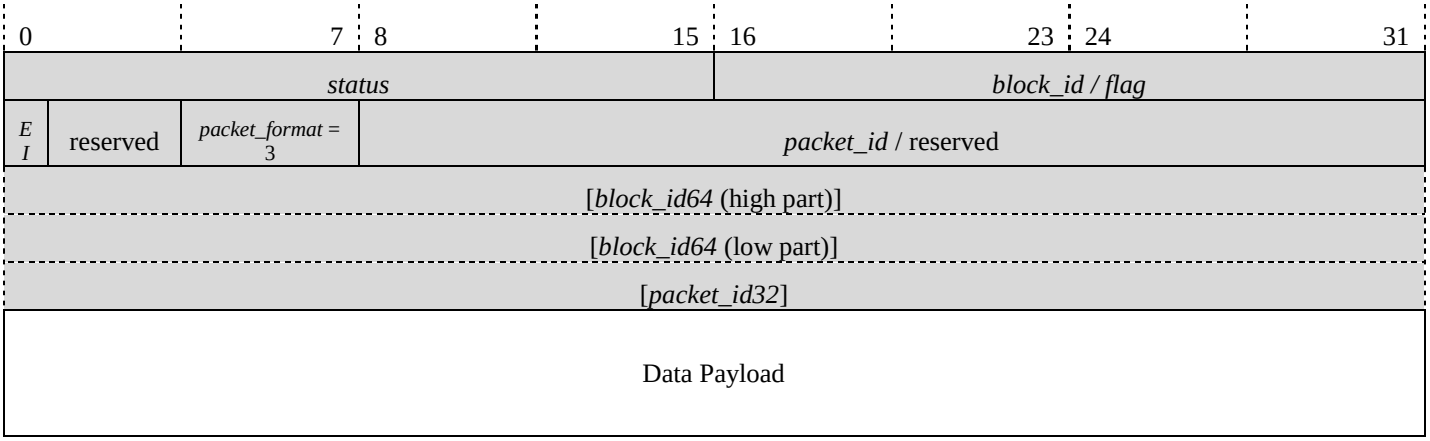


Figure 24-5: Generic Data Payload Packet

24.3.3 Data Trailer Packet

[R-266st] The data trailer packet MUST be the last packet of the block. [R24-18st]

[R-267st] The data trailer packet MUST be sent as a separate packet that fits in one packet of 576 bytes at most (including the IP, UDP and GVSP header). [R24-19st]

[R-268st] After the data trailer packet is sent, the *block_id/block_id64* MUST be incremented to its next valid value and the *packet_id/packet_id32* MUST be reset to 0 for the next block. [R24-20st]

[R-269s] The Data Trailer packet MUST follow the layout of Figure 24-6. [R24-21s]

For different payload types, additional information might be appended to this header.

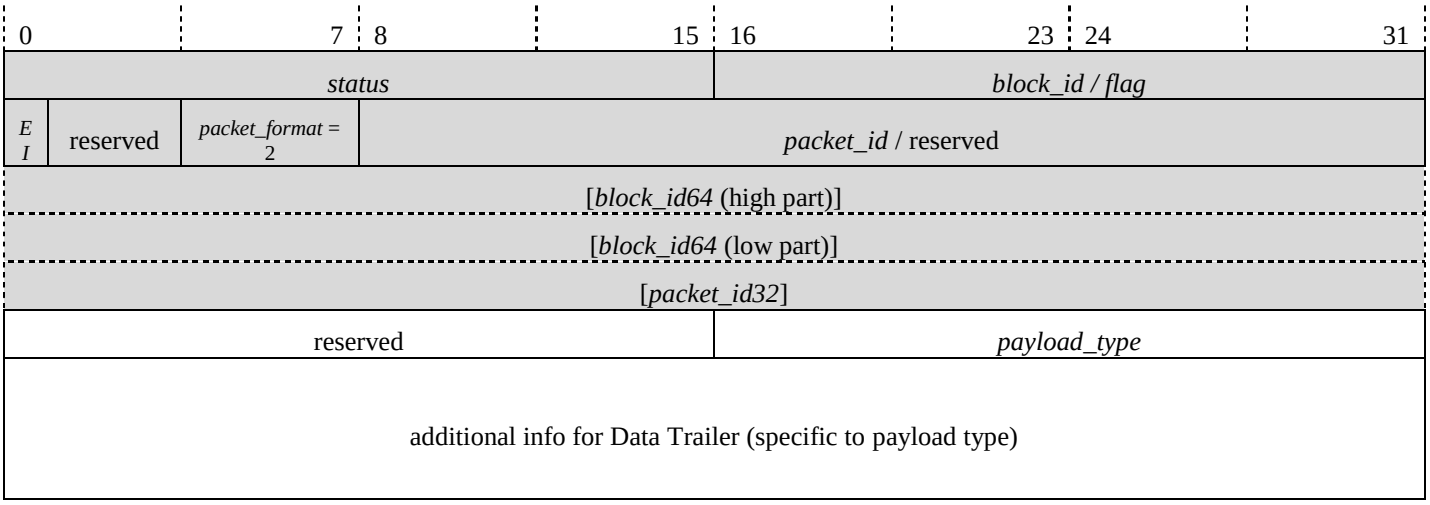


Figure 24-6: Generic Data Trailer Packet

DATA TRAILER PACKET		
reserved	16 bits	Set to 0 on transmission, ignore on reception.
payload_type	16 bits	Same as the *payload_type* field of the Data Leader Packet (Figure 24-4).

24.4 All-in Transmission Mode Packet

The All-in Transmission mode has a single packet type. This packet provides the same information found in the Data Leader packet, Data Payload packet and Data Trailer packet of the Standard Transmission mode presented in the previous section.

[CR-270st] If All-in Transmission is supported, then all the data pertaining to a block MUST be contained in a single packet when this transmission mode is enabled.

[CR-271st] If All-in Transmission is supported, the All-in packet MUST have its *packet_id/packet_id32* set to 0.

[CR-272st] If All-in Transmission is supported, then the *block_id/block_id64* MUST be incremented to its next valid value after the All-in packet is sent.

[CR-273st] If All-in Transmission is supported, then the maximum size of the All-in packet is given by the *packet_size* field of the SCPSx register.

[CR-274st] If the All-in Transmission mode is enabled, when the payload data exceed the available space in the payload data packet, then the GVSP transmitter MUST return a GEV_STATUS_DATA_OVERRUN status code in this All-in packet. In this case, the quantity of data to return in the payload data section of the All-in packet is left as a quality of implementation.

When too much data is available to put in the All-in packet, it is acceptable to truncate the amount of data to fit in the packet. In this case, the rest of the data is discarded since only a single packet can be sent in All-in Transmission mode.

[CR-275s] If All-in Transmission is supported, then the All-in packet MUST follow the layout of Figure 24-7.

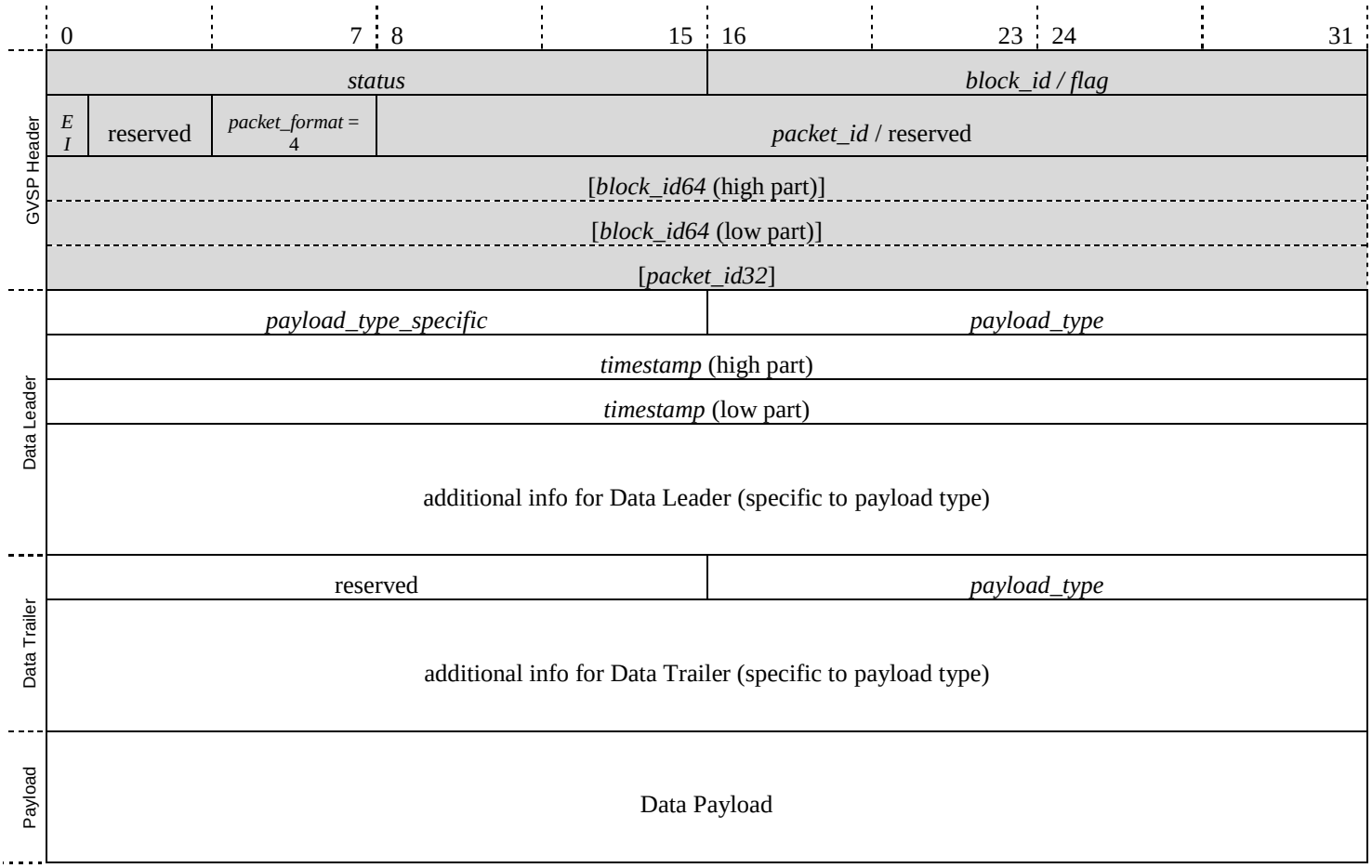


Figure 24-7: Generic All-in Packet

24.5 Chunk Data

Chunks are tagged blocks of data that can be grouped within a data block to send metadata. Examples for chunks are:

- image
- data extracted from image
- AOI / pixel format
- state of I/O pins
- exposure number

Each chunk consists of the chunk data and a trailing tag. The tag contains:

- A unique chunk identifier, which identifies the structure of the chunk data and the chunk feature associated with this chunk
- The length of the chunk data.

As the chunk tags (*CID* and *length* fields) are headers embedded in the payload of chunk format block, their byte order is big-endian.

[CR-276s] If Chunk Data is supported, then all chunk tags MUST use network byte order (big-endian). [CR23-12s]

Endianess of the chunk data itself is defined in the XML device description file (if applicable to the data type). A special situation happens when Chunk Data contains an image. In this case, to follow the convention of the Image payload type, the pixel information should be transmitted in little-endian.

[CR-277s] If Chunk Data is supported, then in order to minimize processing time on the GVSP receiver side, chunk image data using multiple-byte in the payload data packet MUST be little-endian by default. [CR23-13s]

A GVSP transmitter or receiver may implement a converter to translate image chunk pixel data from little-endian to big-endian. This converter can be activated using the SCPSx bootstrap register when supported. The SCCx register indicates if big-endian is supported by the given stream channel.

The table below describes the general structure of any chunk

Table 24-2: Chunk Data Content

Position	Format	Description
0	*data* [K Bytes]	The data that the chunk is transporting. This section must be a multiple of 4 bytes. If it is not, the data has to be padded with zeros to a multiple of four bytes such that K is a multiple of 4 bytes. This ensures *CID* and *length* fields are 32-bit aligned within the chunk.
K	*CID* [4 Bytes]	The chunk identifier
K+4	*length* [4 Bytes]	The length of the data (in bytes, must be a multiple of 4)

The chunk structure with the tag behind the data allows GVSP transmitters to output data in chunk format as a stream, even in case of variable length data (such as in line scan applications).

A chunk data block consists of a sequence of chunks in chunk data format as depicted below:

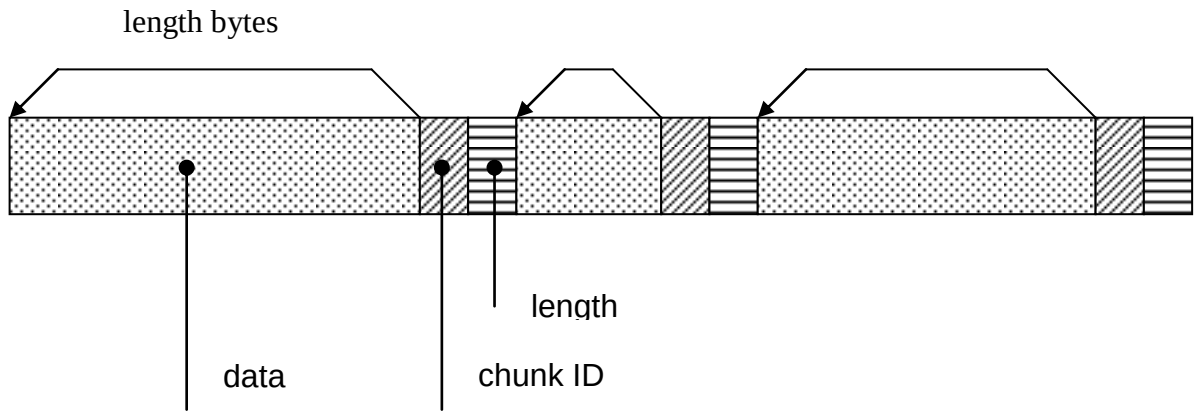


Figure 24-8: Chunk Data Diagram

[CR-278st] If Chunk Data is supported, a stream using chunk data MUST output data using the chunk data format. [CR23-9st]

A block in chunk format is decoded with the help of a GenICam™ chunk parser. To let the chunk parser distinguish between the chunks added by multiple enabled chunk features, each chunk carries a chunk ID (*CID*). The *CID* for each chunk is transferred just before the chunk's *length* information.

[CO-279st] If Chunk Data is supported, the chunks SHOULD be defined in the XML device description file. [CO23-10st]

The chunk parser decodes a block in chunk format beginning with the last received data walking to the beginning of the block. The chunk parser exposes the chunk data as read-only features.

24.5.1 Byte Ordering Example for Chunk Data

The following figure shows the breakdown of the payload of a specific GVSP block in chunk format into byte fields.

Assume the following parameters:

- SCPSx is set to little-endian (default)
- Image with a 1 x 1 AOI
  - ChunkOffsetX: 3
  - ChunkOffsetY: 6
  - Mono8 gray value: 0x83
- and a 32-bit frame counter with the value 0x00000008.

	Byte 0	Byte 1	Byte 2	Byte 3	Description
Image chunk	0x83	0x00	0x00	0x00	Image data + padding bytes
	0x61	0x7d	0x18	0xdb	Chunk id for image data (big-endian)
	0x00	0x00	0x00	0x04	Length of the chunk (big-endian)
Frame counter chunk	0x08	0x00	0x00	0x00	Frame counter (little-endian)
	0x8c	0x0f	0x1c	0xd8	Chunk id for frame counter (big-endian)
	0x00	0x00	0x00	0x04	Length of the chunk (big-endian)
Image info chunk	0x03	0x00	0x06	0x00	ChunkOffsetX, ChunkOffsetY
	0x01	0x00	0x01	0x00	ChunkWidth, ChunkHeight
	0x23	0x0f	0x1c	0x23	Chunk id for image info (big-endian)
	0x00	0x00	0x00	0x08	Length of the chunk (big-endian)

24.5.2 GenICam Chunk Definition Example

The GenICam fragment to define the ImageInfo chunk in the above example is:

```
<RegisterDescription>
…
                <MaskedIntReg Name="ChunkOffsetXValue">
                        <Address>0x00</Address>
                        <Length>4</Length>
                        <AccessMode>RO</AccessMode>
                        <pPort>ImageInfoPort</pPort>
                        <LSB>16</LSB>
                        <MSB>31</MSB>
                        <Sign>Unsigned</Sign>
                        <Endianess>LittleEndian</Endianess>
                </MaskedIntReg>
                <MaskedIntReg Name="ChunkOffsetYValue">
                        <Address>0x00</Address>
                        <Length>4</Length>
                        <AccessMode>RO</AccessMode>
                        <pPort>ImageInfoPort</pPort>
                        <LSB>0</LSB>
                        <MSB>15</MSB>
                        <Sign>Unsigned</Sign>
                        <Endianess>LittleEndian</Endianess>
                </MaskedIntReg>
```

```
                <MaskedIntReg Name="ChunkWidthValue">
                        <Address>0x04</Address>
                        <Length>4</Length>
                        <AccessMode>RO</AccessMode>
                        <pPort>ImageInfoPort</pPort>
                        <LSB>16</LSB>
                        <MSB>31</MSB>
                        <Sign>Unsigned</Sign>
                        <Endianess>LittleEndian</Endianess>
                </MaskedIntReg>
                <MaskedIntReg Name="ChunkHeightValue">
                        <Address>0x04</Address>
                        <Length>4</Length>
                        <AccessMode>RO</AccessMode>
                        <pPort>ImageInfoPort</pPort>
                        <LSB>0</LSB>
                        <MSB>15</MSB>
                        <Sign>Unsigned</Sign>
                        <Endianess>LittleEndian</Endianess>
                </MaskedIntReg>
        <Port Name="ImageInfoPort">
            <ChunkID>230f1c23</ChunkID>
        </Port>
</RegisterDescription>
```

Refer to the GenICam specification for additional information about Chunk Data.

24.6 Test Packet

A management entity can use test packets to discover the MTU of the connection. This is done on each network interfaces associated to a GVSP transmitter using the stream channel registers.

[CR-280st] If the GVSP transmitter supports the *test_data* capability in the GVCP Capability register, it MUST fill the *payload_data* section of the test packet with the output of the linear feedback shift register (LFSR) generator provided in Section 0. In this case, the first byte is set to the LFSR initial value LSB and equals 0xFF. Otherwise, the payload data is "don't care". This is required for all the network interfaces associated to a GVSP transmitter. [CR24-29st]

[CO-281sr] If the GVSP transmitter supports the *test_data* capability in the GVCP Capability register for the test packet and it fires a test packet, then a GVSP receiver and its application requesting this test packet SHOULD compare the content of the payload data against the data generated by the LFSR generator provided in Section 0. [CO24-30sr]

A GVSP transmitter could create a test packet by clearing the GVSP header and *payload_data* field of the packet. Obviously, Ethernet, IP and UDP headers must have valid values (they are not cleared).

[R-282s] The Test Packet MUST follow the layout of Figure 24-9. [R24-28st]

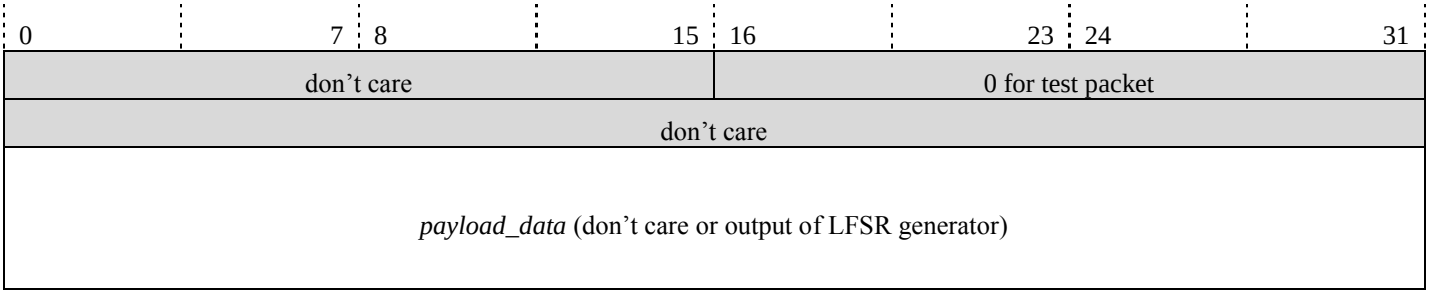


Figure 24-9: Test Packet

Test packet		
payload_data	up to packet size	Filled with don't care data or output of LFSR generator. Total data must be such that IP Header (20 bytes) + UDP Header (8 bytes) + GVSP Header + payload data is equal to the packet size specified in the Stream Channel Packet Size register of the GVSP transmitter. The don't fragment bit must also be set in the IP Header.

Note: The format of the test packet is defined to maintain compatibility with version 1.x of this specification. Note that the Extended ID field is not defined (don't care) and hence a receiver must be configured to anticipate that a test packet is coming and decode it accordingly.

24.6.1 LFSR Generator

The data for the test packet is generated with a maximum sequence 16-bit LFSR. This allows a GVSP transmitter implementation using minimum resources. The period of such an LFSR is (2^{16} -1). The LFSR is clocked once for every byte of output data generated. The byte is filled with the LSB byte of the shift register.

The generator has the following properties:

- 16-bit, right-shifted
- initial value = 0xFFFF
- Polynomial (feedback connection) = 0x8016

Here is a possible VHDL pseudo-implementation:

```
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity NumberGenerator is
	port(
		iClock  : in std_logic;
		iStart  : in std_logic;
		iEnable : in std_logic;
		oData   : out std_logic_vector(7 downto 0)
	);
end NumberGenerator;

architecture behavior of NumberGenerator is
	constant POLYNOMIAL  : std_logic_vector(15 downto 0) := X"8016";
	constant INIT_VALUE  : std_logic_vector(15 downto 0) := X"FFFF";
	signal ShiftRegister : std_logic_vector(15 downto 0);
begin
	process(iClock)
		variable Temp    : std_logic_vector(15 downto 0);
	begin
		if rising_edge(iClock) then
			if iStart = '1' then
				ShiftRegister <= INIT_VALUE;
			elsif iEnable = '1' then
				Temp := ShiftRegister;
			     if Temp(0) = '1' then
					Temp := '0' & Temp(15 downto 1);
					Temp := Temp xor POLYNOMIAL;
				else
					Temp := '0' & Temp(15 downto 1);
				end if;
				ShiftRegister <= Temp;
			end if;
		end if;
	end process;

	oData <= ShiftRegister(7 downto 0);

end behavior;
```

Here is a possible C pseudo-implementation:

```
unsigned short POLYNOMIAL = 0x8016;
unsigned short INIT_VALUE = 0xFFFF;

// reset to init value
unsigned short lfsr = INIT_VALUE;
// the output byte
unsigned char  databyte;

// loop kernel for each databyte
       databyte = (lfsr & 0xff); /// LSB byte of lfsr
       // calculate next lfsr state
      lfsr = ((lfsr >> 1) ^ (-(lfsr & 1) & POLYNOMIAL));
```

25 Payload Types

To efficiently transport information, GVSP defines various payload types that can be streamed out of a GVSP transmitter. These payload types are listed in the following table.

Table 25-1: Payload Types

Payload Type	Value	Description
Image	0x0001 0x4001[a]	Uncompressed image data Support for Image extended chunk data is allowed. In this case, Image must be the first chunk.
Raw data	0x0002 0x4002[a]	Raw binary data Support for Raw data extended chunk data is allowed. In this case, Raw data section must be the first chunk.
File	0x0003 0x4003[a]	A computer file ready to be saved to PC hard drive Support for File extended chunk data is allowed. In this case, File must be the first chunk.
Chunk data	0x0004	Tagged blocks of data, as per the GenICam™ specification.
Extended chunk data (deprecated)	0x0005	Tagged blocks of data, where the first chunk is an image. **Not recommended for new designs**. Use Extended Chunk Mode with Image payload type as an alternative.
JPEG	0x0006[a] 0x4006[a]	JPEG compressed image, as per ITU-T Rec. T.81 Support for JPEG extended chunk data is allowed. In this case, JPEG compressed image must be the first chunk.
JPEG 2000	0x0007[a] 0x4007[a]	JPEG 2000 codestream, as per ITU-T Rec. T.800 Support for JPEG 2000 extended chunk data is allowed. In this case, JPEG 2000 codestream must be the first chunk.
H.264	0x0008[a] 0x4008[a]	H.264 access unit, as per ITU-T Rec. H.264 Support for H.264 extended chunk data is allowed. In this case, H.264 access unit must be the first chunk.
Multi-zone Image	0x0009[a] 0x4009[a]	Uncompressed image data sliced into horizontal bands called zones. Support for Multi-zone Image extended chunk data is allowed by using an additional top-down zone. In this case, Multi-zone Image must be the first chunk.
Device-specific	≥ 0x8000	Payload type specific to the device vendor

[a]: These payload types are only available in Extended ID mode

25.1 Extended Chunk Mode

Metadata can be appended to some of the payload type by setting bit 1 of the *payload_type* field (the extended chunk mode bit). This metadata is attached as Chunk Data following the rules presented in section 24.5.

Refer to Table 25-1 to determine which payload type supports an Extended Chunk mode.

When the Extended Chunk mode bit is set (bit 1 of the *payload_type* field):

1. The Data Leader Packet remains the same as for the standard Data Leader packet of the given payload type, with the exception of the Extended Chunk mode bit.
2. The Data Payload Packet contains chunk data, with the first chunk being equivalent to the selected Payload Format. The packet format of all data payload packets must be the same and match the one indicated by the basic payload type. This is especially important for H.264 and Multi-zone Image payload type which define a dedicated packet format.
3. The Data Trailer Packet is appended with 2 additional fields:
    a. *chunk_data_payload_length*
    b. *chunk_layout_id*

These 2 fields are illustrated below:

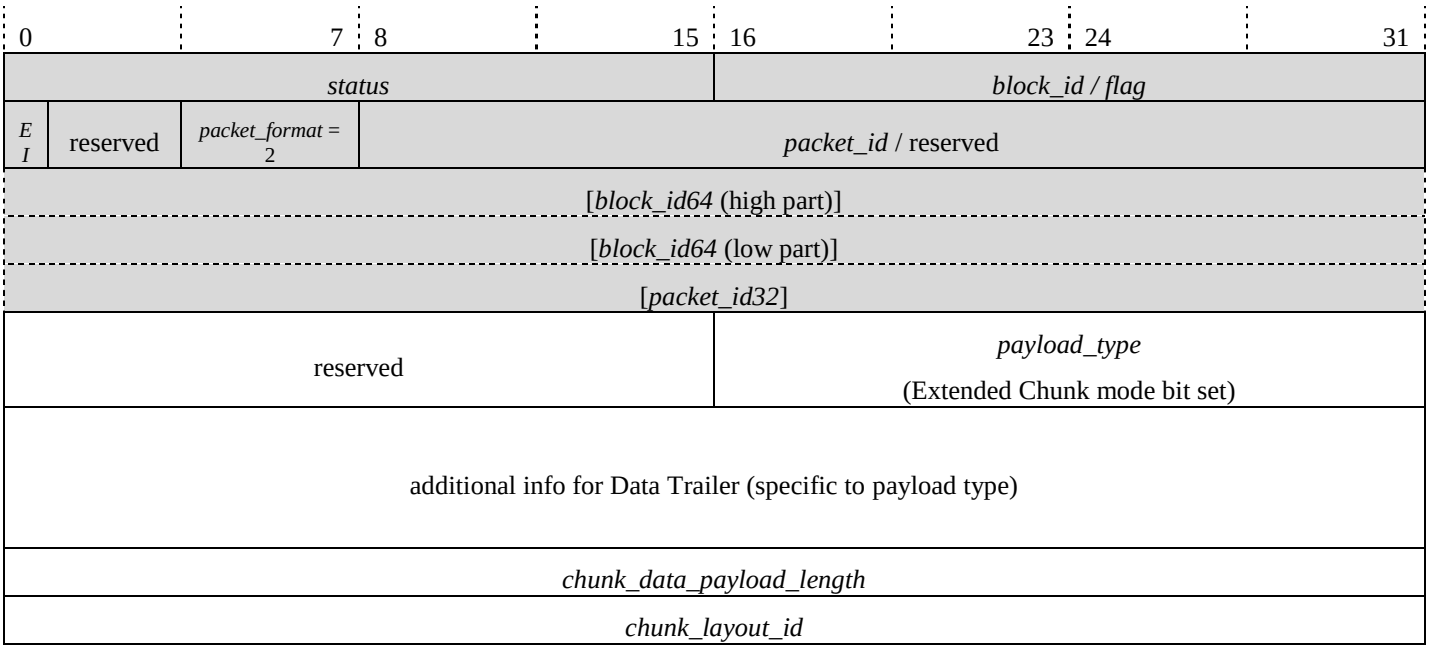


Figure 25-1: Data Trailer Packet with Chunk Data

DATA TRAILER PACKET WITH CHUNK DATA		
payload_type	16 bits	bit 1: Extended Chunk mode data = 1
chunk_data_payload_length	32 bits	Sum of the length of all the chunks data payload in bytes. It must be a multiple of 4 bytes (bits 30 and 31 must be cleared). This field is put here to help GVSP receivers finding the end of the chunk data payload even if the last GVSP payload packet of a block is padded.
chunk_layout_id	32 bits	This field serves as an indicator that the chunk layout has changed and the GVSP receiver should re-parse the chunk layout in the buffer. When a chunk layout (availability or position of individual chunks) changes since the last block, the GVSP transmitter MUST change the chunk layout ID. As long as the chunk layout remains the same, the GVSP transmitter MUST keep the chunk layout ID identical. When switching back to a layout, which was already used before, the GVSP transmitter can use the same ID that was used then or use a new ID. A chunk layout ID value of 0 is invalid. It is reserved to be used by GVSP transmitters not supporting the layout ID functionality. If the GVSP transmitter does not support the chunk layout ID functionality, it MUST set this field to 0 for all blocks. The algorithm used to compute the chunk layout ID is left as quality of implementation.

Extended Chunk for Multi-zone Image

A special situation appears for Multi-zone Image payload type as the payload data is segmented into zones. To facilitate the retrieval of the chunk information for this payload type, it is necessary to define an additional zone that is used exclusively to carry the chunk tags of the first chunk and all the additional chunks. This additional zone is always top-down and must follow the same rules used by any zone. The previous zones carry a regular Multi-zone Image (with no chunk tag appended). Hence a minimum of two zones are necessary to carry Extended Chunk for Multi-zone Image.

[CR-283st] If Multi-zone Image payload is supported, then an additional zone MUST be used to carry the first chunk tags and all additional chunks when Extended Chunk is used with that payload type.

25.2 Image Payload Type

This payload type is used to transmit uncompressed images.

[CR-284st] If Image payload is supported, a stream using the image payload type MUST put data in each Data Payload packet in raster-scan format. [CR23-3st]

This means the image is reconstructed in the GVSP transmitter memory before being transmitted from left to right, then top to bottom. This is typical with single-tap sensor.

Note: GigE Vision version 2.0 introduces the optional Multi-zone Image payload type to facilitate transmission of image data coming from multi-tap sensors. Refer to section 25.10 for additional information.

The position of the first pixel in a given data packet can be obtained by multiplying packet size (SCPSx register) with "*packet_id/packet_id32* – 1". That means there is no gap in the image data.

[CR-285st] If Image payload is supported, the GVSP transmitter MUST send Data Payload packets sequentially (from 1 to N-1 as per Figure 24-1). [CR23-4st]

If packets are lost, a GVSP receiver and its application can use the PACKETRESEND command (when supported by the GVSP transmitter) to ask for a group of consecutive packets.

[CR-286st] If Image payload is supported, for multi-tap data to be sent on a single stream channel, the GVSP transmitter MUST reconstruct the image locally so it can be sent using sequential packet ID. [CR23-5st]

[CR-287st] If Image payload is supported, when a multi-tap transmitter cannot reconstruct the image before transmission, then it SHOULD use a separate stream channel for each tap to transmit. [CR23-6st]

It is then up to the GVSP receiver to reconstruct the image (at the expense of increased processing time and resources). The tap configuration information and the way they are associated to stream channels should be available in the XML device description file or in the product documentation.

This specification recommends that a product transmitting images reconstructs the image in raster-scan format before transmission.

25.2.1 Image Data Leader Packet

[CR-288s] If Image payload is supported, its Data Leader packet MUST follow the layout of Figure 25-2. [CR24-5s]

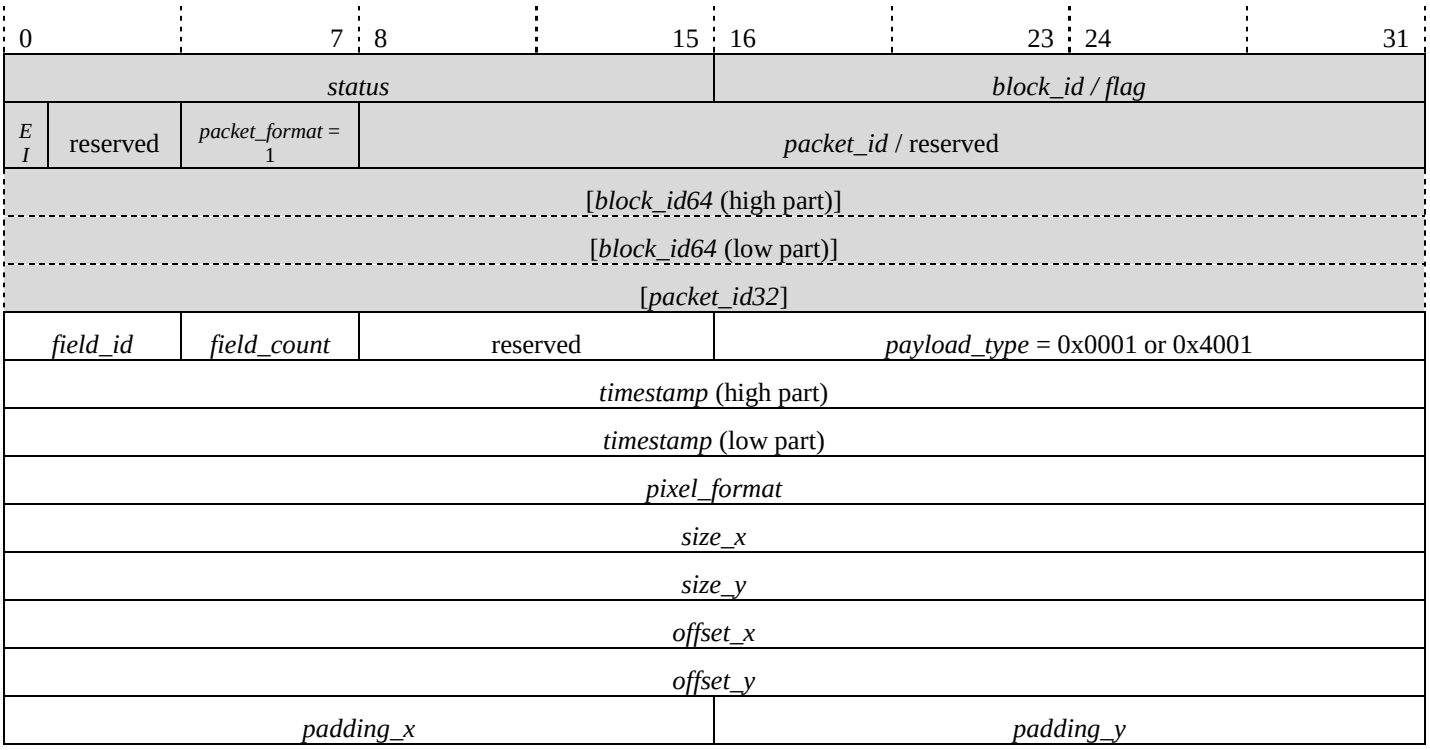


Figure 25-2: Image Data Leader Packet

IMAGE DATA LEADER PACKET		
field_id	4 bits	For interlaced data, provides the index of the field in the image. *Progressive data*: always 0 *2-field interlaced data*: Field 1 = 0, Field 2 = 1 *3-field interlaced data*: Field 1 = 0, Field 2 = 1, Field 3 = 2 Other values are reserved GVSP receivers that cannot de-interlace should ignore this field. **Note**: The first line of a de-interlaced image comes from the first line of Field 1, the second line from Field 2 and so on.
field_count	4 bits	For interlaced data, provides the total number of fields minus 1. *Progressive data*: always 0 *2-field interlaced data*: field count = 1 *3-field interlaced data*: field count = 2 Other values are reserved GVSP receivers that cannot de-interlace should ignore this field.
reserved	8 bits	Set to 0 on transmission, ignore on reception.
payload_type	16 bits	Refer to Table 25-1.
timestamp	64 bits	64-bit timestamp representing when the block of data was generated. Timestamps are optional. This field should be 0 when timestamps are not supported. Different ROIs from the same exposure must use the same timestamp, as per [CR-253st]. **Note**: The GenICam specification supports 64-bit signed integer while this field is 64-bit unsigned. Therefore, only 63 bits are available through the GenICam interface.

pixel_format	32 bits	This field gives the pixel format of payload data. Refer to Pixel Layout section.
size_x	32 bits	Width in pixels of the image transported in the data payload portion of this data block.
size_y	32 bits	Height in lines of the image (for progressive scan) or field (for interlaced scan) transported in the data payload portion of this data block. For interlaced image, this represents the height of one field as each field is considered to be a data block. In variable frame size, the GVSP transmitter can transmit less than this value. If this is the case, the *size_y* field of the data trailer provides the actual value transmitted.
offset_x	32 bits	Offset in pixels from image origin. Used for ROI support. When no ROI is defined this field must be set to 0. For CFA, it is recommended to have *offset_x* increment by the full width of the CFA. This is to ensure the *pixel_format* field matches the PixelFormat feature configured by the primary application. Other pixel formats might pose restrictions on the value taken by this field.
offset_y	32 bits	Offset in lines from image origin. Used for ROI support. When no ROI is defined this field must be set to 0. For CFA, it is recommended to have *offset_y* increment by the full height of the CFA. This is to ensure the *pixel_format* field matches the PixelFormat feature configured by the primary application. Other pixel formats might pose restrictions on the value taken by this field.
padding_x	16 bits	Horizontal padding expressed in bytes. Number of extra bytes transmitted at the end of each line to facilitate image alignment in buffers. This can typically used to have 32-bit aligned image lines. This is similar to the horizontal invalid (or horizontal blanking) in analog cameras. Set to 0 when no horizontal padding is used. If *padding_x* is different from 0, then the last pixel of the line must be padded if necessary (this is referred to as line padding in the Pixel Format Naming Convention) and then the number of bytes specified by the *padding_x* field are appended.
padding_y	16 bits	Vertical padding expressed in bytes. Number of extra bytes transmitted at the end of the image to facilitate image alignment in buffers. This could be used to align buffers to certain block size (for instance 4 KB). This is similar to the vertical invalid (or vertical blanking) in analog cameras. Set to 0 when no vertical padding is used. **Note**: When *padding_y* is used for variable frame size, then the transmitter must send the number of padding bytes indicated by *padding_y*, even if the actual *size_y* reported in the data trailer is different than the one indicated in the data leader.

25.2.2 Image Data Payload Packet

[CR-289s] If Image payload is supported, its Data Payload packet MUST follow layout of Figure 25-3. [CR24-13s]

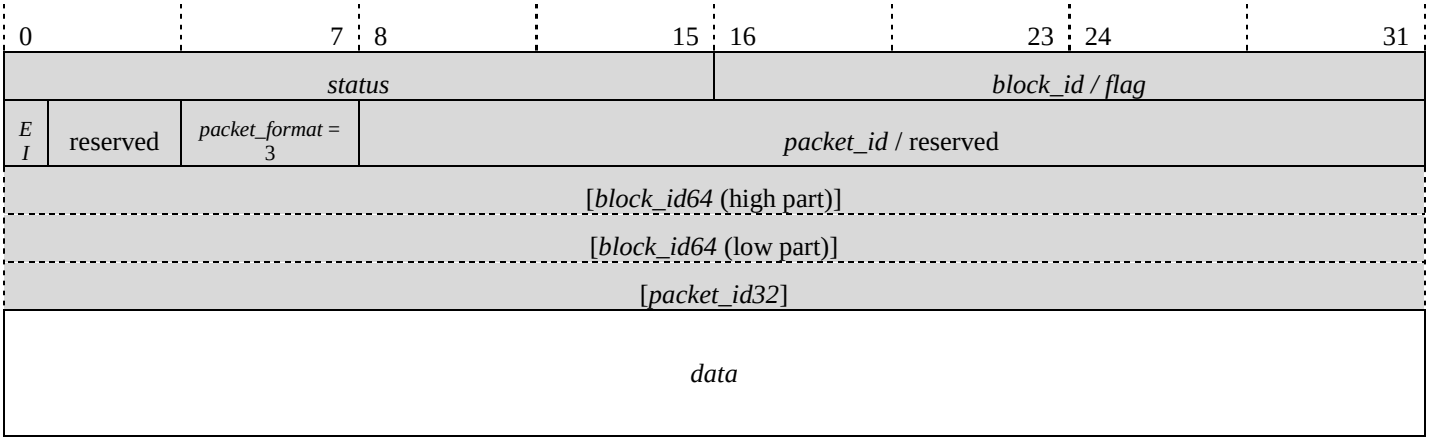


Figure 25-3: Image Data Payload Packet

IMAGE DATA PAYLOAD PACKET		
data	up to packet size	Image data formatted as specified in the Data Leader *pixel_format* field. For Data Payload packets, the IP Header + UDP Header + GVSP Header + data must be equal to the packet size specified in the Stream Channel Packet Size register of the GVSP transmitter. The only exception is the last Data Payload packet which may be smaller than the specified packet size. However, it is also possible to pad the last data packet so that all data payload packets are exactly the same size.

25.2.3 Image Data Trailer Packet

[CR-290s] If Image payload is supported, its Data Trailer packet MUST follow the layout of Figure 25-4. [CR24-22s]

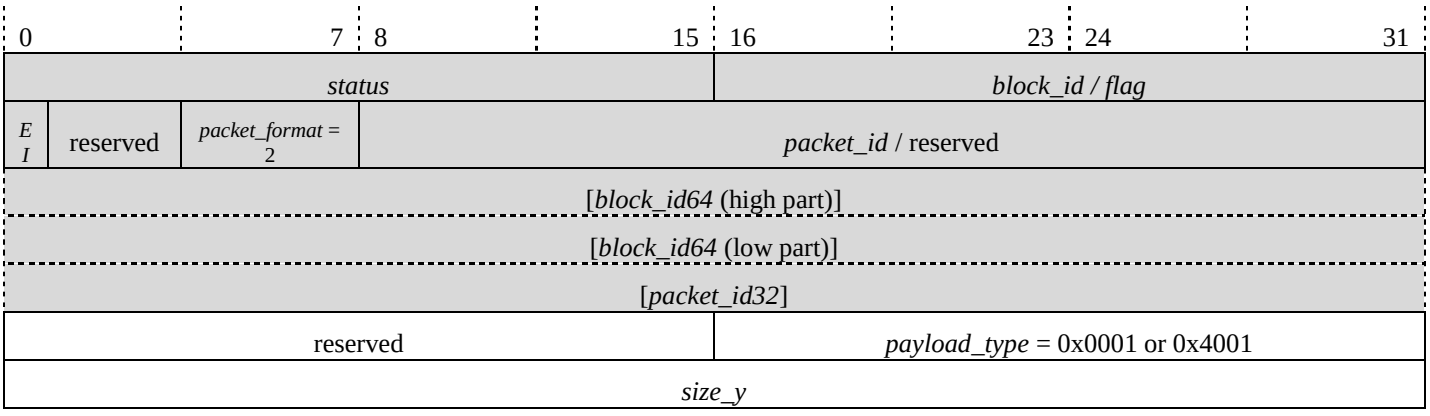


Figure 25-4: Image Data Trailer Packet

IMAGE DATA TRAILER PACKET		
reserved	16 bits	Set to 0 on transmission, ignore on reception.
payload_type	16 bits	Refer to Table 25-1.
size_y	32 bits	This field is the actual height, in lines, for this particular data block. This is done to support variable frame size image sources once the actual number of lines transmitted has been confirmed. Only image height can be variable. For interlaced image, this represents the height of one field as each field is considered to be a data block.

25.2.4 Image All-in Packet

[CR-291s] When All-in transmission mode is supported, if Image payload is supported, then its All-in packet MUST regroup the Image Data Leader, Image Data Trailer and Image Data Payload according to Figure 24-7.

[CR-292st] If Image Extended Chunk payload type is supported, then the transmitter MUST use the Extended ID mode for the stream channels associated to the Image Extended Chunk payload type.

25.3 Raw Data Payload Type

This payload type is used to stream raw data from a GVSP transmitter to GVSP receivers. For instance, this can be used to send acquisition statistics.

[CR-293st] If Raw Data payload is supported, the amount of information to transfer MUST be provided in the data leader. It can be variable from one data block to the next. [CR23-7st]

25.3.1 Raw Data Leader Packet

[CR-294s] If Raw Data payload is supported, its Data Leader packet MUST follow the layout of Figure 25-5. [CR24-6s]

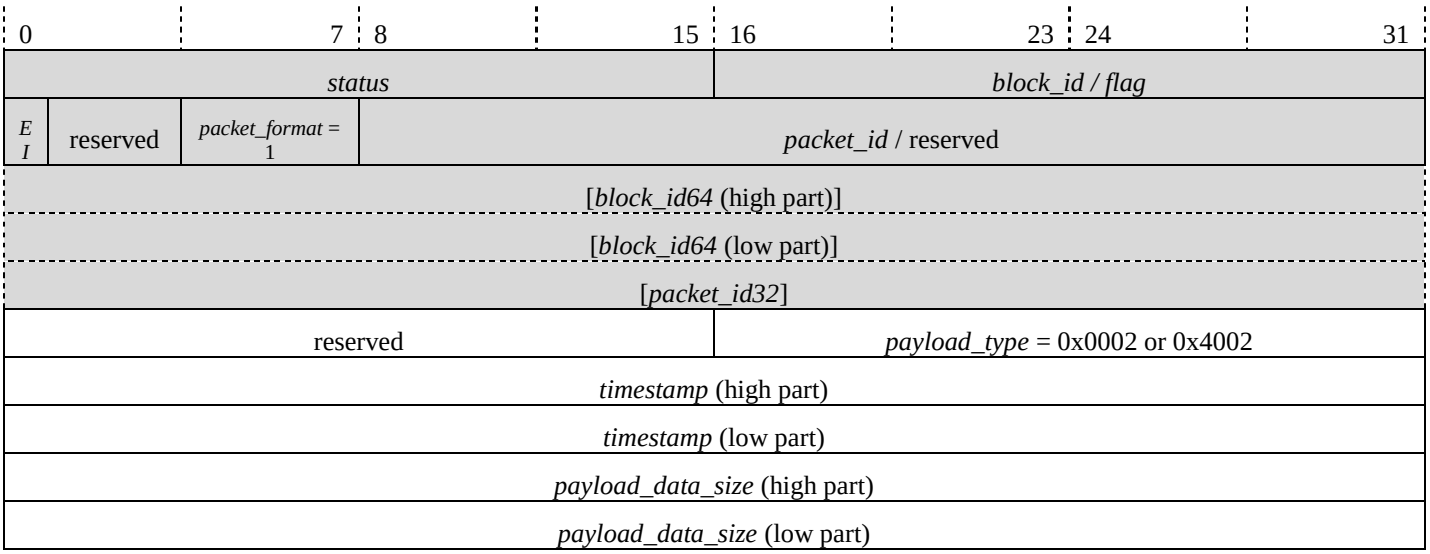


Figure 25-5: Raw Data Leader Packet

RAW DATA LEADER PACKET		
reserved	16 bits	Set to 0 on transmission, ignore on reception.
payload_type	16 bits	Refer to Table 25-1.
timestamp	64 bits	See Image payload type.
payload_data_size	64 bits	Total size of payload data in bytes.

25.3.2 Raw Data Payload Packet

[CR-295s] If Raw Data payload is supported, its Data Payload packet MUST follow layout of Figure 25-6. [CR24-14s]

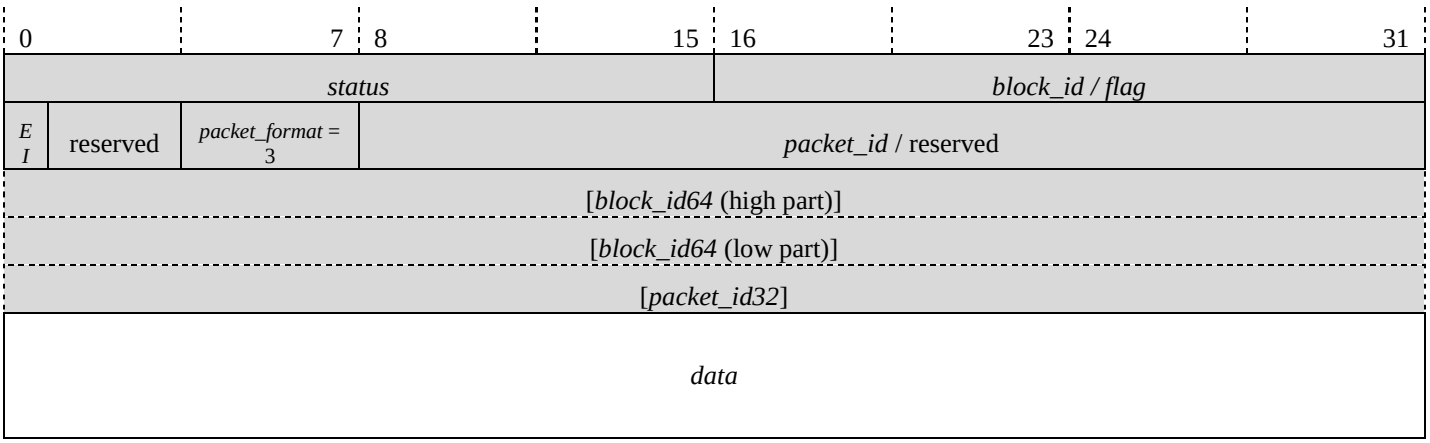


Figure 25-6: Raw Data Payload Packet

RAW DATA PAYLOAD PACKET		
data	up to packet size	Raw data. For Data Payload packets, the IP Header + UDP Header + GVSP Header + data must be equal to the packet size specified in the Stream Channel Packet Size register of the GVSP transmitter. The only exception is the last Data Payload packet which may be smaller than the specified packet size. However, it is also possible to pad the last data packet so that all data payload packets are exactly the same size.

25.3.3 Raw Data Trailer Packet

[CR-296s] If Raw Data payload is supported, its Data Trailer packet MUST follow the layout of Figure 25-7. [CR24-23s]

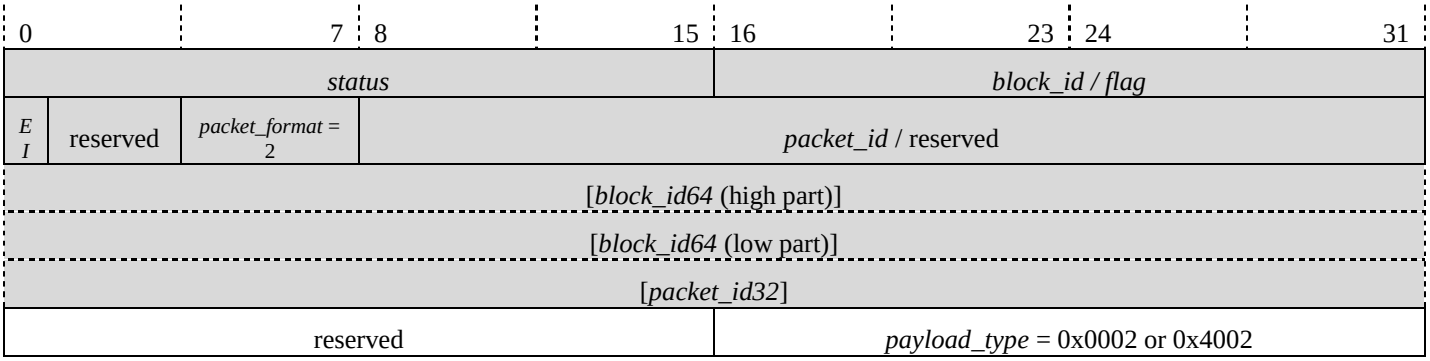


Figure 25-7: Raw Data Trailer Packet

RAW DATA TRAILER PACKET		
reserved	16 bits	Set to 0 on transmission, ignore on reception.
payload_type	16 bits	Refer to Table 25-1.

25.3.4 Raw All-in Packet

[CR-297s] When All-in transmission mode is supported, if Raw Data payload is supported, then its All-in packet MUST regroup the Raw Data Leader, Raw Data Trailer and Raw Data Payload according to Figure 24-7.

[CR-298st] If Raw Data Extended Chunk payload type is supported, then the transmitter MUST use the Extended ID mode for the stream channels associated to the Raw Data Extended Chunk payload type.

25.4 File Payload Type

This payload type is used to transfer files that can be directly saved to a GVSP receiver hard disk. For instance, a GVSP transmitter could use GIF image compression to deliver compressed files to a GVSP receiver.

[CR-299st] If File payload is supported, the amount of information to transfer MUST be provided in the data leader. It can be variable from one data block to the next. [CR23-8st]

25.4.1 File Data Leader Packet

[CR-300s] If File payload is supported, its Data Leader packet MUST follow the layout of Figure 25-8. [CR24-7s]

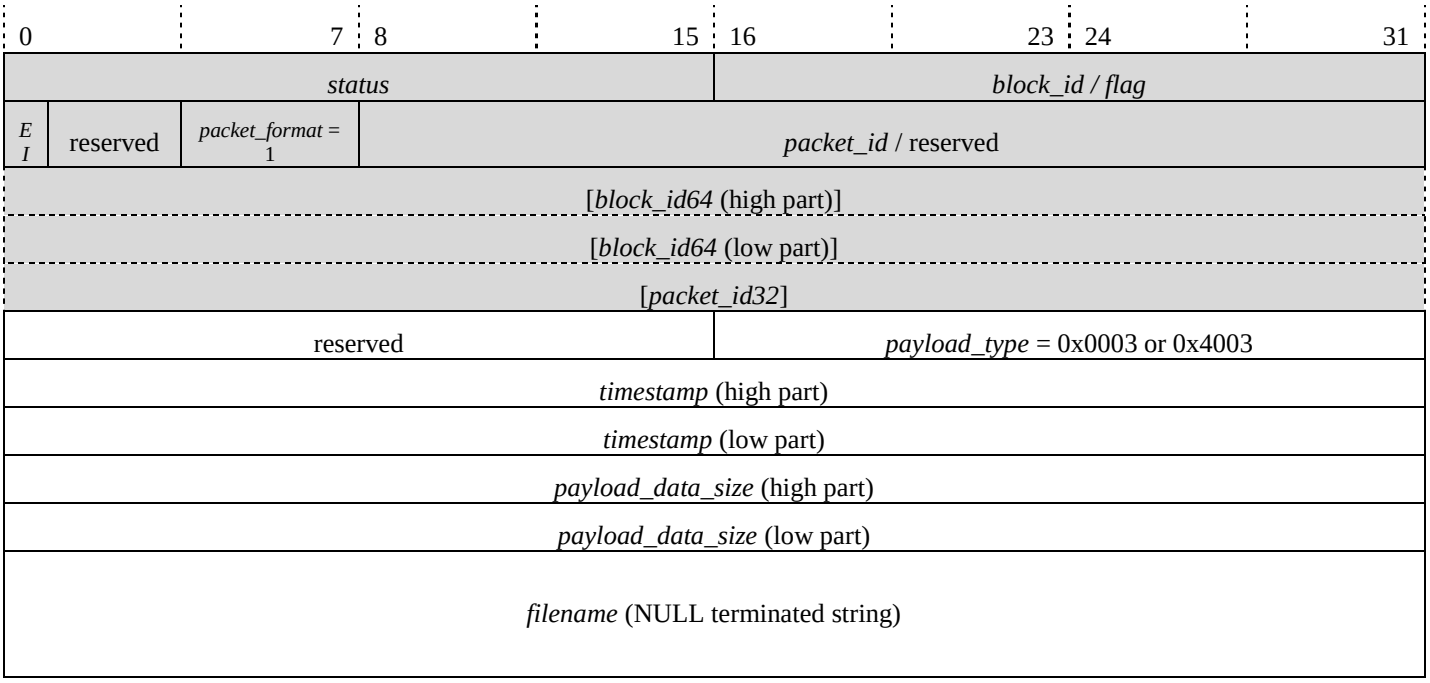


Figure 25-8: File Data Leader Packet

FILE DATA LEADER PACKET		
reserved	16 bits	Set to 0 on transmission, ignore on reception.
payload_type	16 bits	Refer to Table 25-1.
timestamp	64 bits	See Image payload type.
payload_data_size	64 bits	See Raw Data payload type.
filename	string size	NULL terminated string using the current character set. The NULL character must be explicit.

25.4.2 File Data Payload Packet

[CR-301s] If File payload is supported, its Data Payload packet MUST follow layout of Figure 25-9. [CR24-15s]

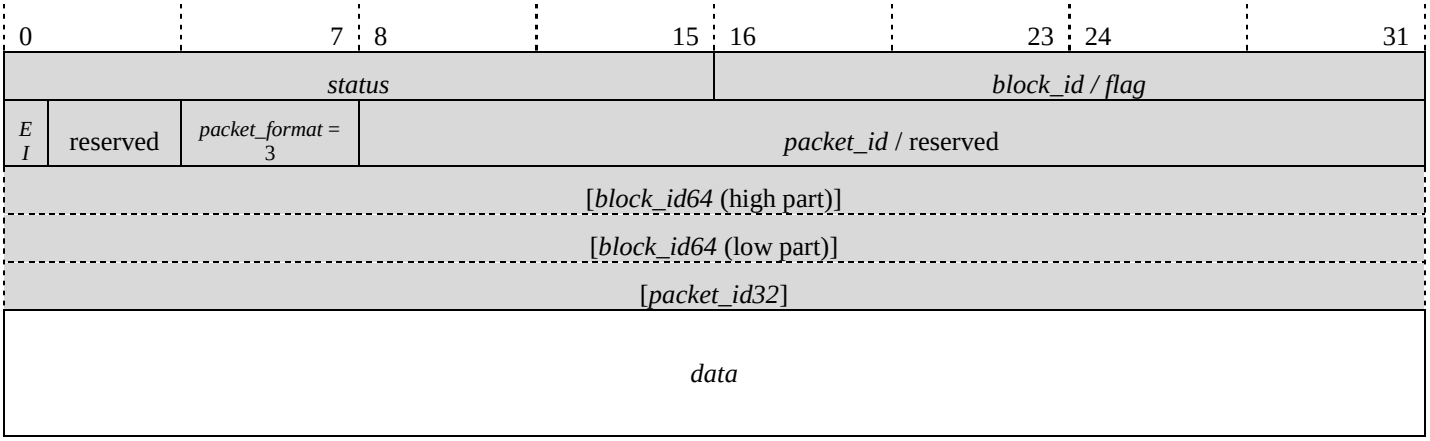


Figure 25-9: File Data Payload Packet

FILE DATA PAYLOAD PACKET		
data	up to packet size	File data. For Data Payload packets, the IP Header + UDP Header + GVSP Header + data must be equal to the packet size specified in the Stream Channel Packet Size register of the GVSP transmitter. The only exception is the last Data Payload packet which may be smaller than the specified packet size. However, it is also possible to pad the last data packet so that all data payload packets are exactly the same size.

25.4.3 File Data Trailer Packet

[CR-302s] If File payload is supported, its Data Trailer packet MUST follow the layout of Figure 25-10. [CR24-24s]

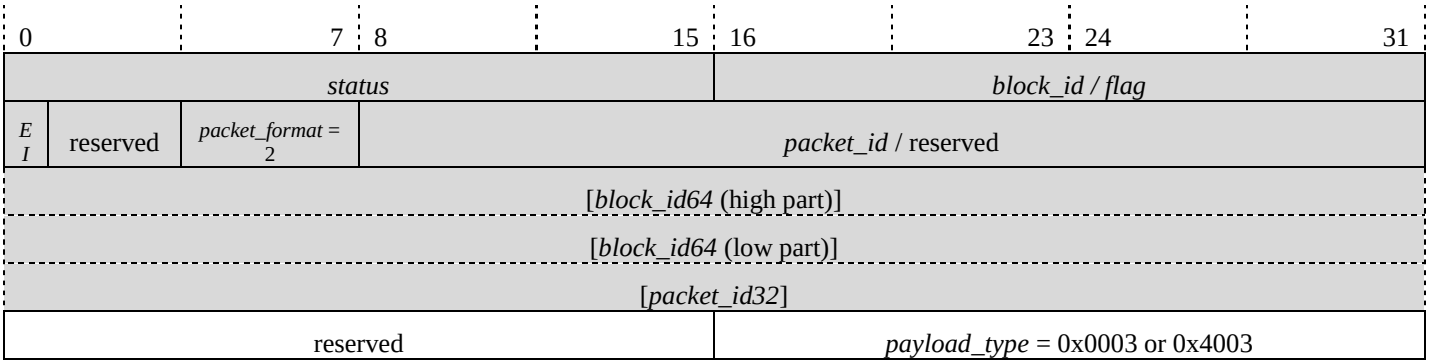


Figure 25-10: File Data Trailer Packet

FILE DATA TRAILER PACKET		
reserved	16 bits	Set to 0 on transmission, ignore on reception.
payload_type	16 bits	Refer to Table 25-1.

25.4.4 File All-in Packet

[CR-303s] When All-in transmission mode is supported, if File payload is supported, then its All-in packet MUST regroup the File Data Leader, File Data Trailer and File Data Payload according to Figure 24-7.

[CR-304st] If File Extended Chunk payload type is supported, then the transmitter MUST use the Extended ID mode for the stream channels associated to the File Extended Chunk payload type.

25.5 Chunk Data Payload Type

This payload type is used to stream chunks. This is similar to the Extended Chunk mode, but with no primary payload type. Hence the first chunk can be anything.

25.5.1 Chunk Data Leader Packet

[CR-305s] If Chunk Data payload is supported, its Data Leader packet MUST follow the layout of Figure 25-11. [CR24-8s]

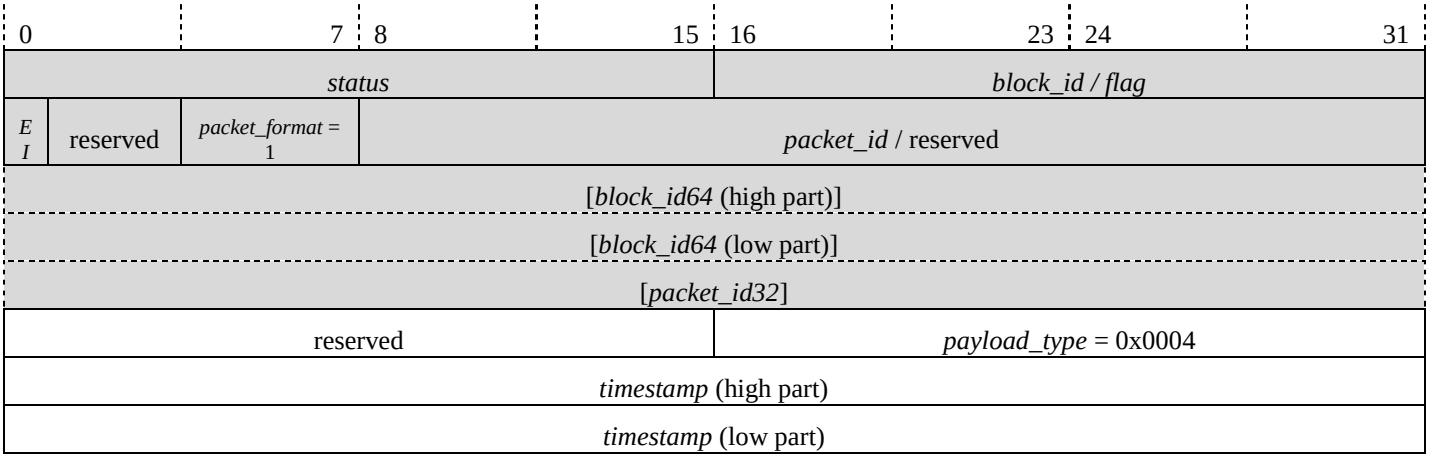


Figure 25-11: Chunk Data Leader Packet

CHUNK DATA LEADER PACKET		
reserved	16 bits	Set to 0 on transmission, ignore on reception.
payload_type	16 bits	Refer to Table 25-1.
timestamp	64 bits	See Image payload type.

25.5.2 Chunk Data Payload Packet

[CR-306s] If Chunk Data payload is supported, its Data Payload packet MUST follow the layout of Figure 25-12. [CR24-16s]

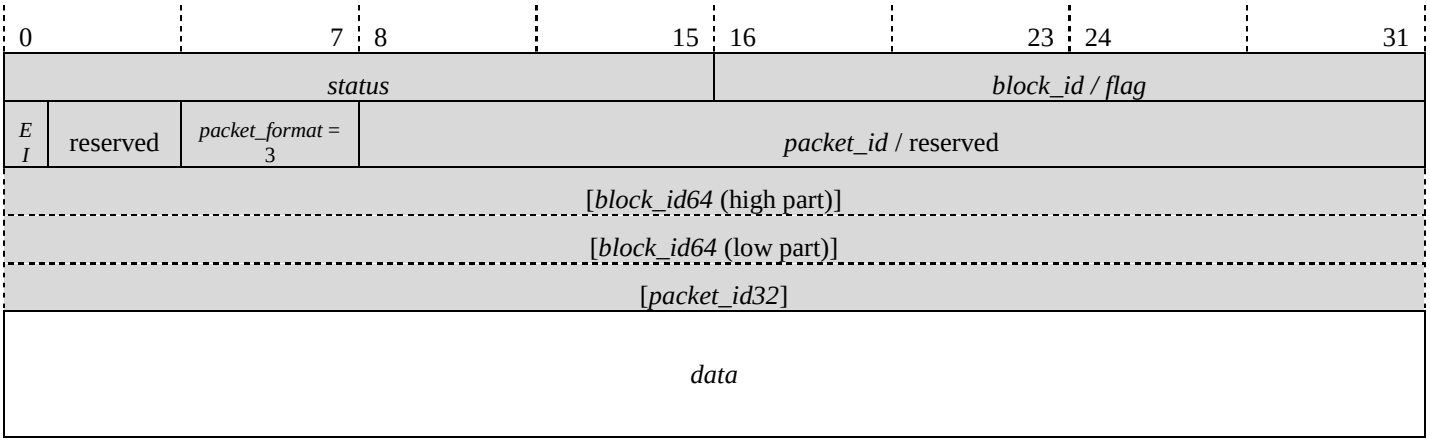


Figure 25-12: Chunk Data Data Payload Packet

CHUNK DATA PAYLOAD PACKET		
data	up to packet size	Chunk data. For chunk data payload packets, the IP Header + UDP Header + GVSP Header + data must be equal to the packet size specified in the Stream Channel Packet Size register of the GVSP transmitter. The only exception is the last Data Payload packet which may be smaller than the specified packet size. However, it is also possible to pad the last data packet so that all data payload packets are exactly the same size.

25.5.3 Chunk Data Trailer Packet

[CR-307s] If Chunk Data Payload is supported, its Data Trailer packet MUST follow the layout of Figure 25-13. [CR24-25s]

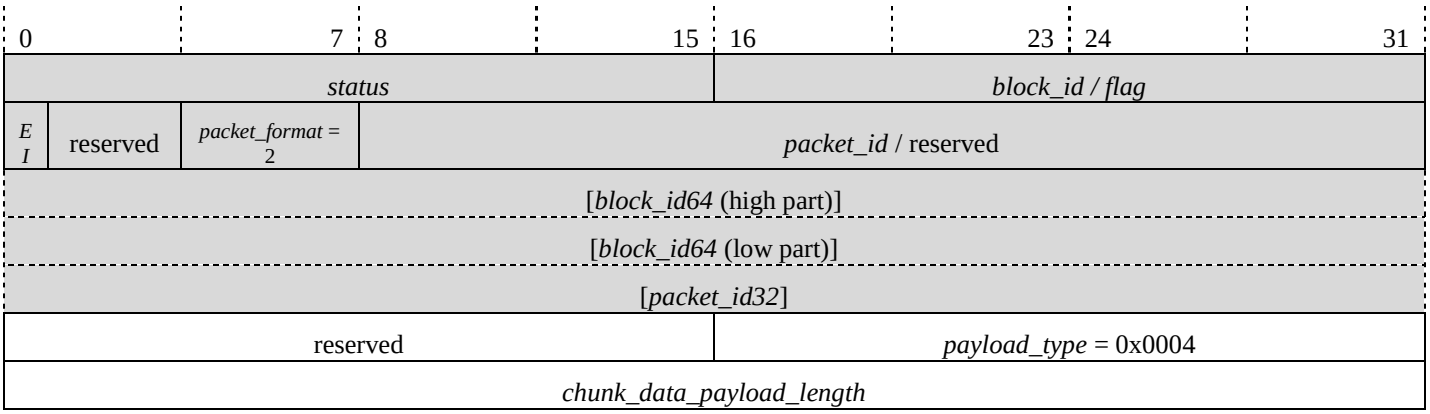


Figure 25-13: Chunk Data Trailer Packet

CHUNK DATA TRAILER PACKET		
reserved	16 bits	Set to 0 on transmission, ignore on reception.
payload_type	16 bits	Refer to Table 25-1.
chunk_data_payload_length	32 bits	Total length of the chunk data payload in bytes. It must be a multiple of 4 bytes (bits 30 and 31 must be cleared). This field is put here to help GVSP receivers finding the end of the chunk data payload.

25.5.4 Chunk All-in Packet

[CR-308s] When All-in transmission mode is supported and enabled, if Chunk payload is used, then its All-in packet MUST regroup the Chunk Data Leader, Chunk Data Trailer and Chunk Data Payload according to Figure 24-7.

25.6 Extended Chunk Data Payload Type (deprecated)

Warning This payload type is deprecated in favor of the Extended Chunk Mode and should not be used for new designs. This section is left in the document to facilitate backward compatibility with GigE Vision 1.x devices.

[CR-309s] When the Extended Chunk Data payload type is used and an image is present in the block, the image data chunk MUST be the first chunk in the data payload of the block. It MUST be aligned, with an offset of zero, with the beginning of the data payload of the block. [CR23-14s]

25.6.1 Extended Chunk Data Leader Packet (deprecated)

[CR-310s] If Extended Chunk Data payload is supported, its Data Leader packet MUST follow the layout of Figure 25-14, derived from the Image Data Leader packet. [CR24-31s]

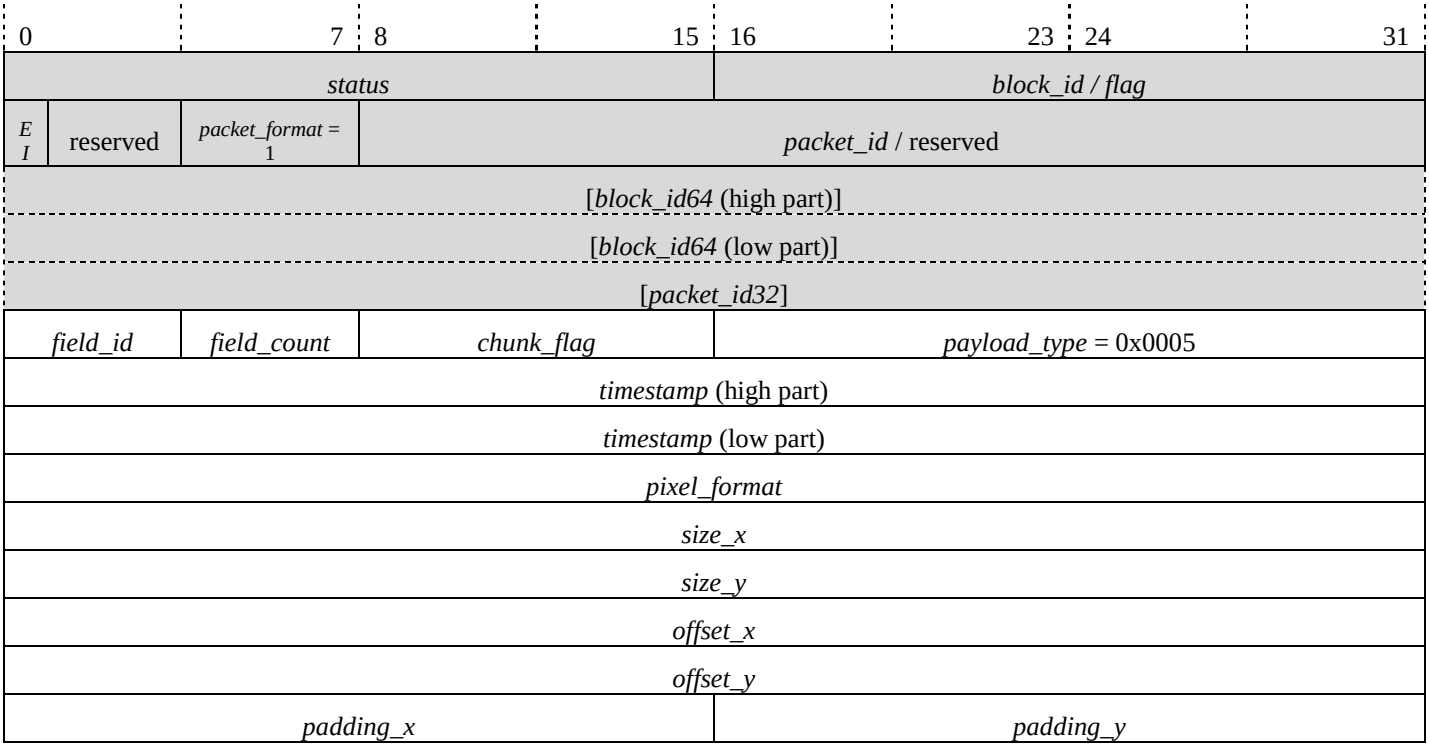


Figure 25-14: Extended Chunk Data Leader Packet (deprecated)

EXTENDED CHUNK DATA LEADER PACKET		
field_id	4 bits	See Image payload type.
field_count	4 bits	See Image payload type.
chunk_flag	8 bits	bit 0 to 6 – Reserved. Set to 0 on transmission, ignore on reception. bit 7 (lsb) – Image Chunk: When set, indicates if an image chunk is present in the block.
payload_type	16 bits	Refer to Table 25-1.
timestamp	64 bits	See Image payload type.
pixel_format	32 bits	See Image payload type.
size_x	32 bits	See Image payload type.
size_y	32 bits	See Image payload type.
offset_x	32 bits	See Image payload type.
offset_y	32 bits	See Image payload type.
padding_x	16 bits	See Image payload type.
padding_y	16 bits	See Image payload type.

The *pixel_format*, *size_x*, *size_y*, *offset_x*, *offset_y*, *padding_x* and *padding_y* fields of the leader are defined as per Section 25.2.1 when the image chunk bit of the *chunk_flag* field is set. Otherwise, they are set to 0.

25.6.2 Extended Chunk Data Payload Packet (deprecated)

[CR-311s] If Extended Chunk Data payload is supported, its Data Payload packet MUST follow the layout of Figure 25-12. [CR24-16s]

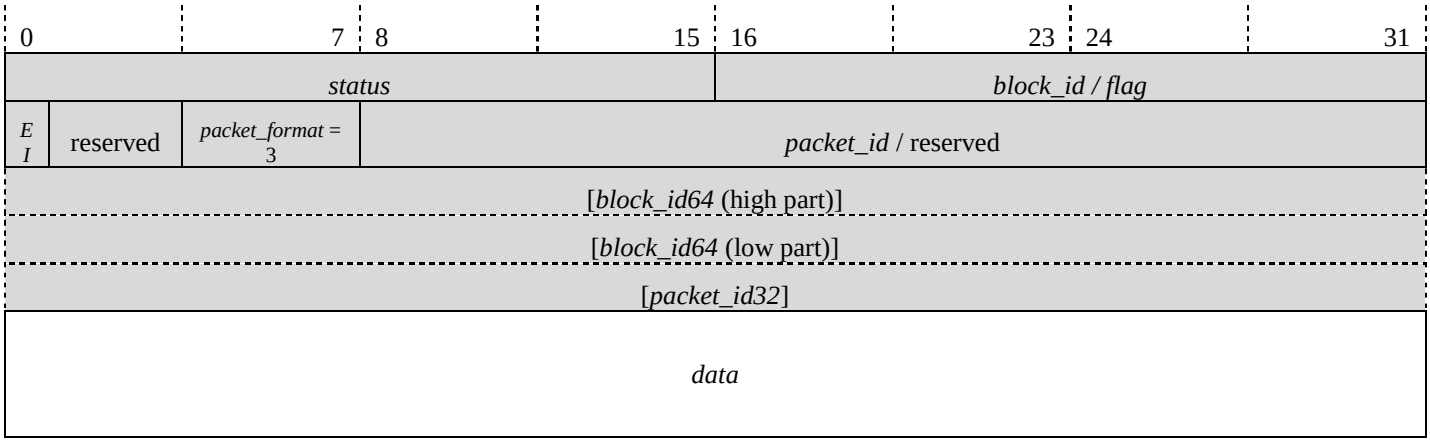


Figure 25-15: Extended Chunk Data Payload Packet (deprecated)

EXTENDED CHUNK DATA PAYLOAD PACKET		
data	up to packet size	Chunk data. For extended chunk data payload packets, the IP Header + UDP Header + GVSP Header + data must be equal to the packet size specified in the Stream Channel Packet Size register of the GVSP transmitter. The only exception is the last Data Payload packet which may be smaller than the specified packet size. However, it is also possible to pad the last data packet so that all data payload packets are exactly the same size.

25.6.3 Extended Chunk Data Trailer Packet (deprecated)

[CR-312s] If Extended Chunk Data Payload is supported, its Data Trailer packet MUST follow the layout of Figure 25-16. [CR24-32s]

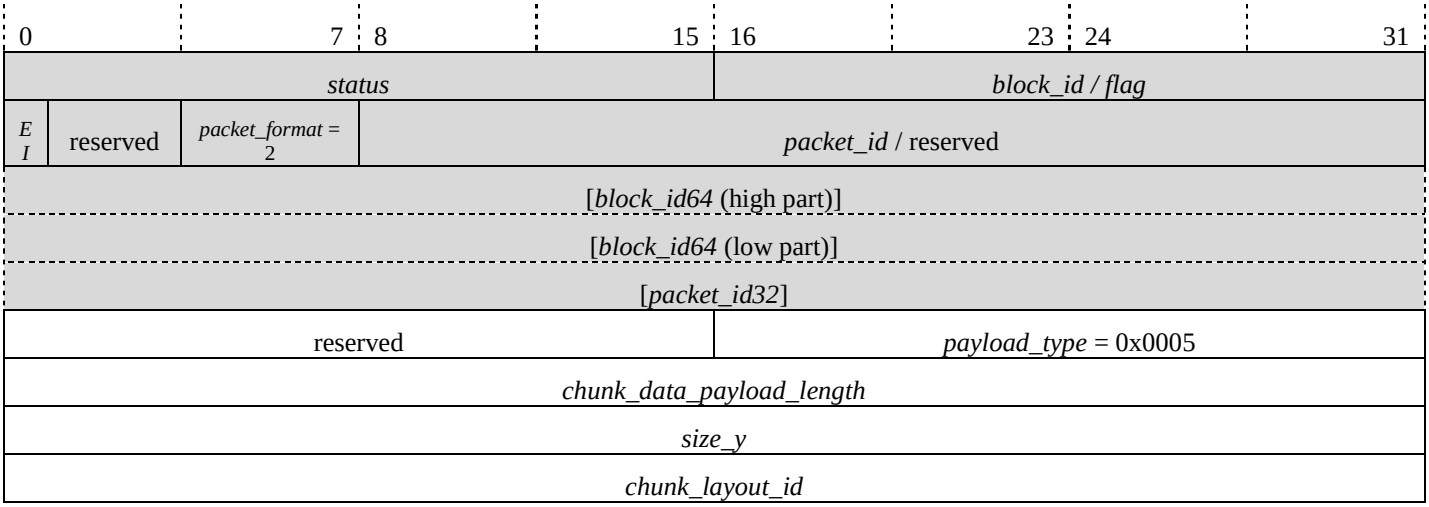


Figure 25-16: Extended Chunk Data Trailer Packet (deprecated)

CHUNK DATA TRAILER PACKET		
reserved	16 bits	Set to 0 on transmission, ignore on reception.
payload_type	16 bits	Refer to Table 25-1.
chunk_data_payload_length	32 bits	See section 25.1- Extended Chunk Mode.
size_y	32 bits	See Image Payload Type. When an image chunk is present in the block, this field is the same as listed under Image Payload Type. If image chunk is not present, this field is set to 0.
chunk_layout_id	32 bits	See section 25.1- Extended Chunk Mode.

25.7 JPEG Payload Type

This section describes a GVSP payload type for the ITU Rec. T.81, better known as JPEG. This JPEG payload type is inspired from RFC2435.

25.7.1 JPEG Principles

A JPEG encoder is use to encode (i.e. compress) a “raw” image to produce a JPEG-compliant compressed data stream. The format of the data stream is defined by the JPEG standard.

The figure below presents a JPEG compressed data stream as depicted in ITU-T Rec. T.81. A JPEG compressed data stream begins with a Start Of Image (SOI) marker, contains one compressed image and ends with an End Of Image (EOI) marker.

A frame begins with a frame header and contains one or more scans. A scan is a single pass through the data for one or more of the components in an image. A frame header may be preceded by one or more table specification or miscellaneous marker segments as specified in the JPEG standard.

A scan begins with a scan header and contains one or more entropy-coded data segments. Each scan header may be preceded by one or more table-specification or miscellaneous marker segments. If restart is not enabled, there is only one entropy-coded segment (the one labeled "last"), and no restart markers are present. If restart is enabled, the number of entropy-coded segments is defined by the size of the image and the defined restart interval. In this case, a restart marker follows each entropy-coded segment except the last one. Restart markers in the JPEG data denote a point where a decoder should reset its state.

An entropy-coded segment is comprised of a sequence of minimum entropy-coded units.

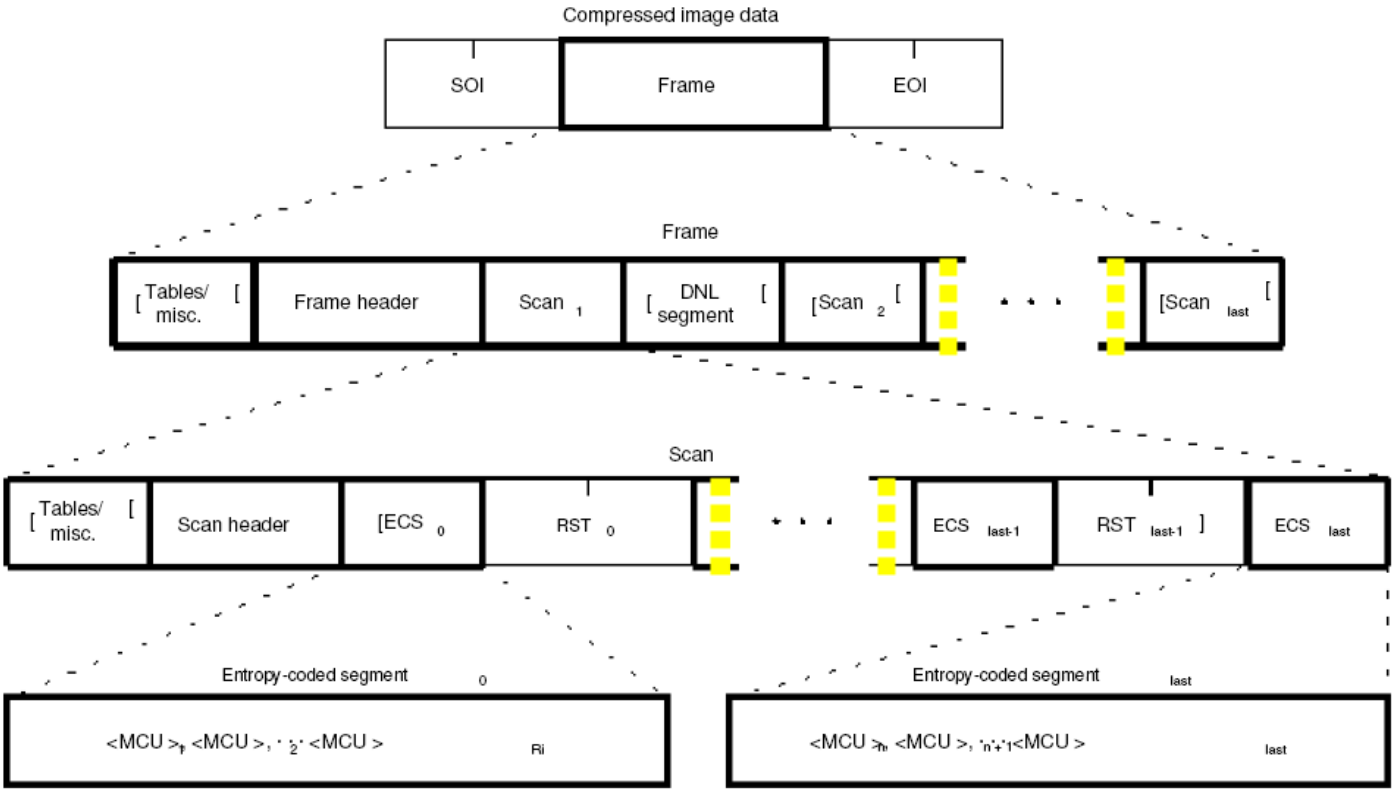


Figure 25-17 : JPEG Compressed Data Stream Format

25.7.2 JPEG Implementation for GVSP

A JPEG compressed data stream includes most of the information required to decode and display a compressed image. This specification defines a GVSP payload format that enables a GVSP receiver to decode and display the image encoded without knowing about the characteristics of the video encoder. An image encoded in a JPEG compressed data stream is transported over a single GVSP data block.

Note: This specification does not prevent a GVSP transmitter to transport a JPEG File Interchange Format (JFIF) file in a GVSP data block. This is because a JFIF file is only different from a JPEG compressed data stream by the presence of the APP0 marker right after the SOI marker.

The JPEG payload type is only supported with GigE Vision Extended ID mode (i.e. *EI* field of the GVSP packet must be set to 1).

[CR-313st] If JPEG payload type is supported, then the transmitter MUST use the Extended ID mode for the stream channels associated to the JPEG payload type (including All-in Transmission).

25.7.3 JPEG Data Leader Packet

[CR-314s] If JPEG payload is supported, its Data Leader packet MUST follow the layout of Figure 25-18.

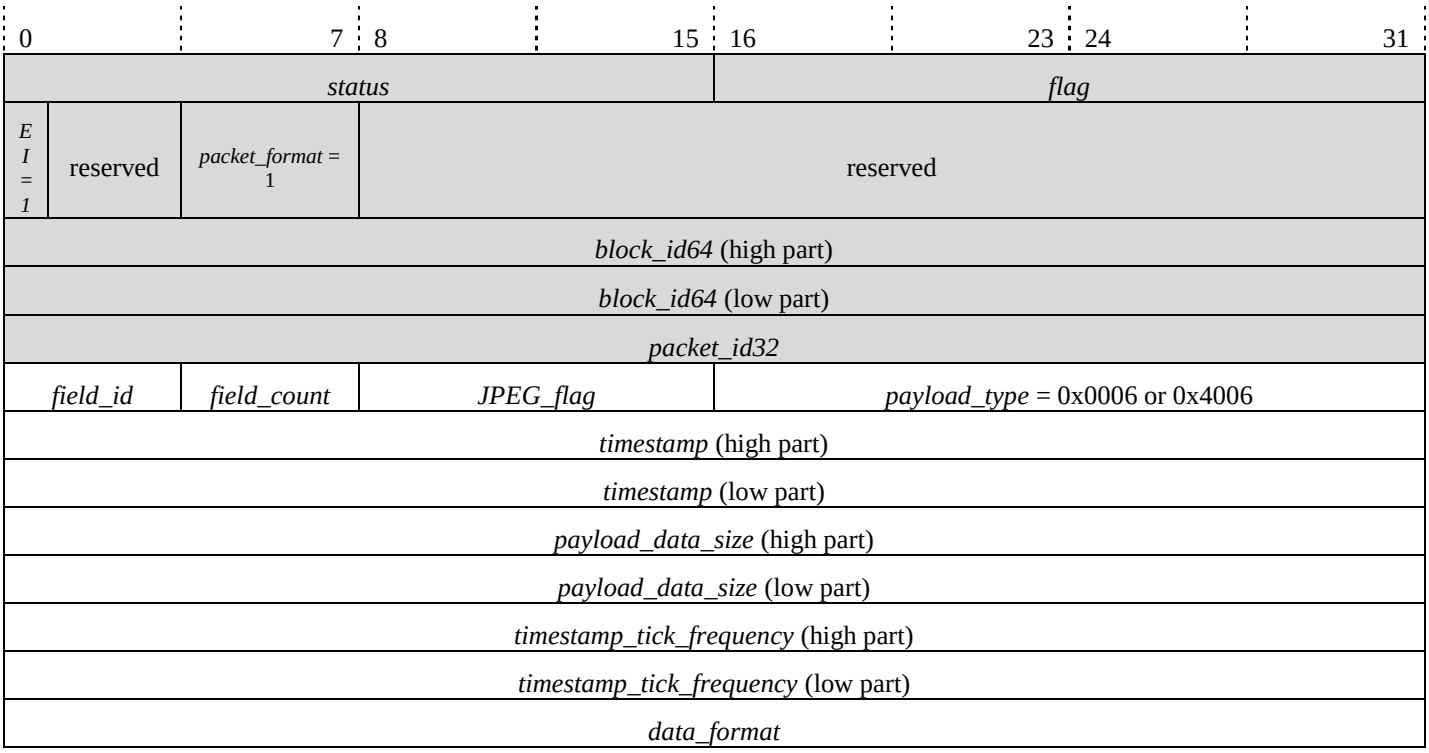


Figure 25-18: JPEG Data Leader Packet

JPEG DATA LEADER PACKET		
EI	1 bit	Extended ID. Must be set to 1 for JPEG payload type.
field_id	4 bits	See Image payload type.
field_count	4 bits	See Image payload type.
JPEG_flag	8 bits	bit 0 to 6 – Reserved. Set to 0 on transmission, ignore on reception. bit 7 (lsb) – color space selection: When clear, indicates that the *data_format* field contains a GigE Vision pixel format. When set, indicates that the *data_format* field uses a valid EnumCS color space, as defined by JPEG 2000.

payload_type	16 bits	Refer to Table 25-1.
timestamp	64 bits	Indicates the presentation time of the image contained in the data block. It is expressed in timestamp tick periods (1 / timestamp tick frequency).
payload_data_size	64 bits	Maximum size of the compressed data stream in bytes. It should be noted that this value is likely not representing the actual size of the GVSP block payload data but the upper limit of the amount of data that will be transmitted in a given block. This is because a GVSP transmitter will likely not know the maximum size of the compressed data when sending the leader down the link.
timestamp_tick_frequency	64 bits	The timestamp clock rate in Hz. Rates below 1000 Hz should not be used in order to provide enough accuracy. This field enables a GVSP receiver to compute the actual presentation time of the compressed image from the *timestamp* field.
data_format	32 bits	Content of this field depends on the color space selection bit. For GigE Vision pixel format, this field represents a valid GigE Vision Pixel Format value. For EnumCS color space, this field specifies the color space of the image. Valid values are defined by the JPEG 2000 EnumCS values. JPEG 2000 EnumCS values provide support for monochrome, sRGB and sYCC color spaces. This is defined by ITU-T Rec. T.800. However, Part 2 of the JPEG 2000 standard (ITU-T Rec. T.801) defines additional EnumCS values. All these values are valid for GigE Vision. **Note**: The JPEG standard text does not specify color space. The compressed data stream can include a number of color components. Monochrome image would use one while YCbCr would use three. It is nevertheless needed to specify more than the number of components in order to be able to display a video feed. Hence the need to include this field.

25.7.4 JPEG Data Payload Packet

[CR-315s] If JPEG payload is supported, its Data Payload packet MUST follow layout of Figure 25-19.

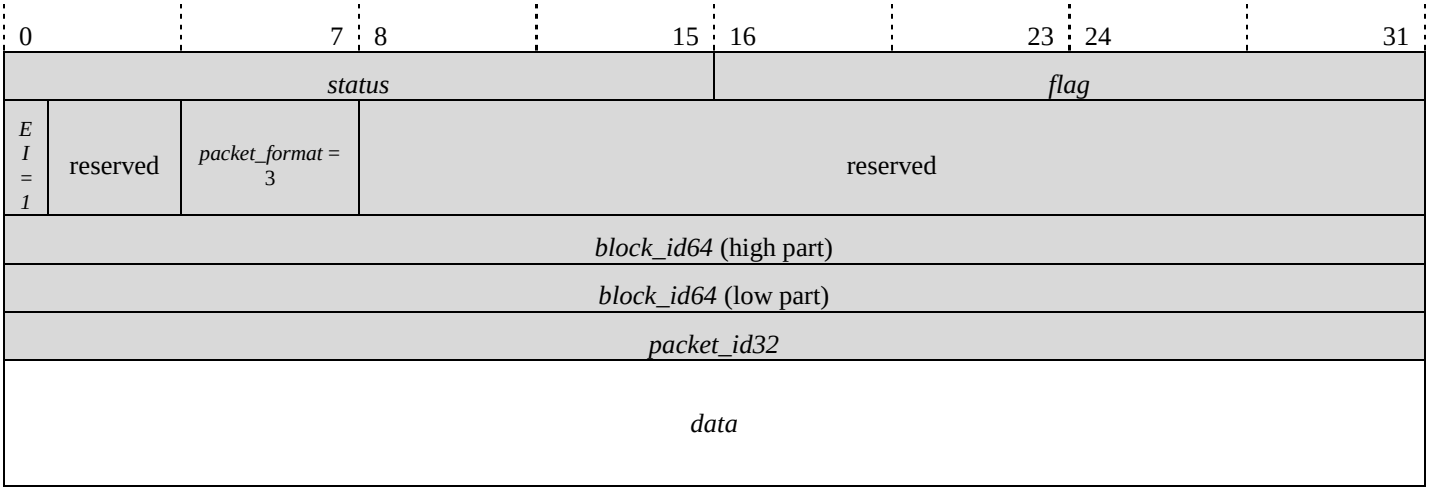


Figure 25-19: JPEG Data Payload Packet

JPEG DATA PAYLOAD PACKET		
EI	1 bit	Extended ID. Must be set to 1 for JPEG payload type.
data	up to packet size	Compressed data stream. The IP header + UDP header + GVSP header + data must be equal to the packet size specified in the Stream Channel Packet Size register of the GVSP transmitter. The only exception is the last Data Payload packet which may be smaller than the specified packet size. However, it is also possible to pad the last data packet so that all data payload packets are exactly the same size. The End Of Image (EOI) JPEG marker is used by the GVSP receiver to identify the end of the compressed data stream.

25.7.5 JPEG Data Trailer Packet

[CR-316s] If JPEG payload is supported, its Data Trailer packet MUST follow the layout of Figure 25-20.

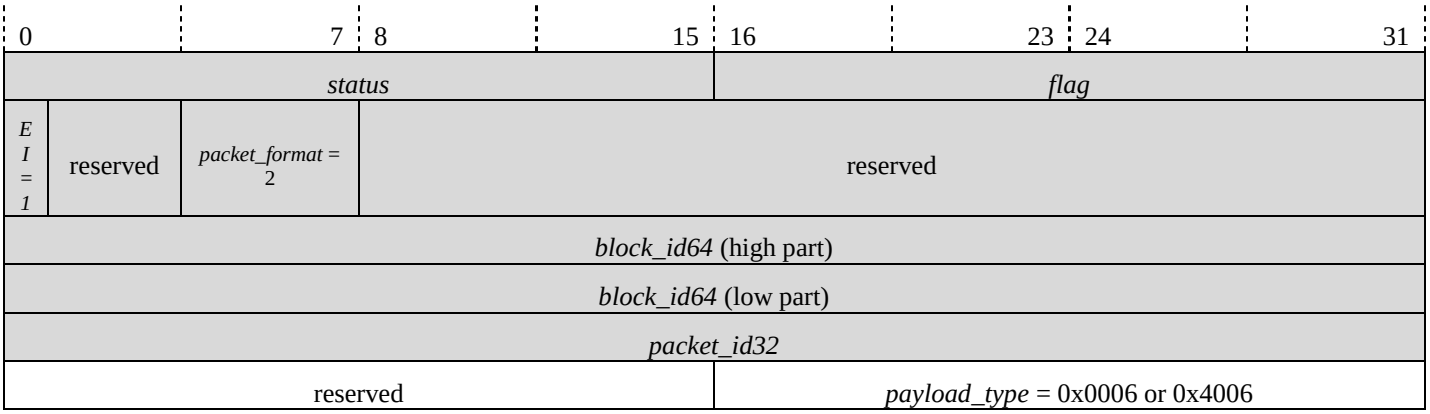


Figure 25-20: JPEG Data Trailer Packet

JPEG DATA TRAILER PACKET		
EI	1 bit	Extended ID. Must be set to 1 for JPEG payload type.
reserved	16 bits	Set to 0 on transmission, ignore on reception.
payload_type	16 bits	Refer to Table 25-1.

25.7.6 JPEG All-in Packet

[CR-317s] When All-in transmission mode is supported, if JPEG payload is supported, then its All-in packet MUST regroup the JPEG Data Leader, JPEG Data Trailer and JPEG Data Payload according to Figure 24-7.

25.8 JPEG 2000 Payload Type

This section describes a GVSP payload type for the ITU-T Rec. T.800, better known as JPEG 2000. This JPEG 2000 payload type is inspired from RFC5371.

25.8.1 JPEG 2000 Principles

A JPEG 2000 encoder is used to encode (compress) a “raw” image to produce a JPEG 2000 compliant codestream. The format of this codestream is defined by the JPEG 2000 standard.

The figure below presents a JPEG 2000 codestream. A JPEG 2000 codestream is structured from the main header, beginning with the Start Of Codestream (SOC) marker, one or more tiles, and the End Of Codestream (EOC) marker to indicate the end of the codestream. Each tile consists of a tile-part header that starts with the Start Of Tile (SOT) marker and ends with a Start Of Data (SOD) marker, and bit stream (a series of JPEG 2000 packets).

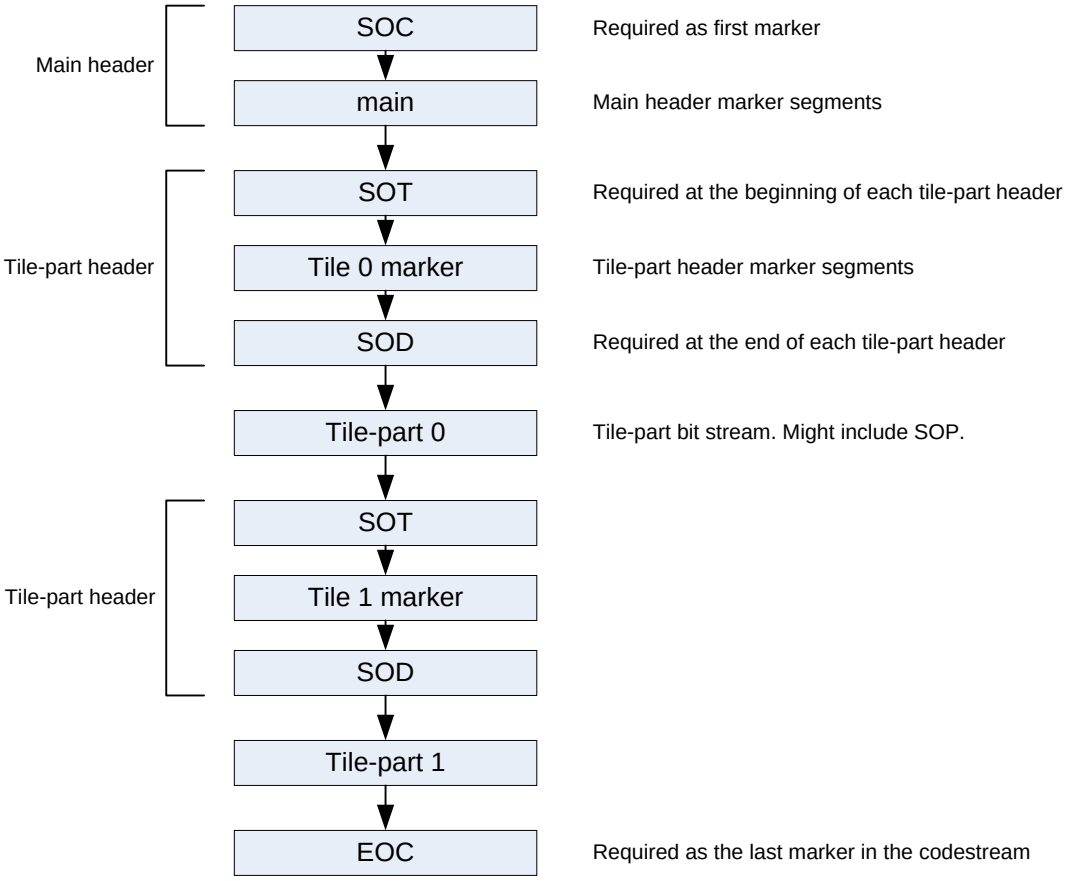


Figure 25-21 : Basic Construction of the JPEG 2000 Codestream

It should be noted that a tile is defined as a rectangular region of an image that is transformed and encoded separately. Also one should not confuse the JPEG 2000 packets referenced in the paragraph above with IP packets being use to transport a JPEG 2000 codestream over Ethernet.

25.8.2 JPEG 2000 Implementation for GVSP

A JPEG 2000 codestream includes most of the information required to decode and display a compressed image. This specification defines a GVSP payload format that enables a GVSP receiver to decode and display the image encoded without knowing about the characteristics of the video encoder. In other words, a JPEG 2000 GVSP payload video feed includes all the information required to decode and display an encoded image. It does not rely on the use of capability announcement and offer/answer protocols. In a GigE Vision system, it is assumed that a higher level management entity is responsible to configure the system so the video receivers can decode the video provided by the relevant video source. Therefore, the lack of support for a capability announcement and offer/answer protocols is not perceived as an issue since the current GigE Vision system level model for the transport of uncompressed video is not relying on such mechanisms. Nevertheless this specification does not prevent one to use such protocols if deemed desirable.

A JPEG 2000 codestream containing the data of one compressed image is transported as one GVSP data block.

The JPEG 2000 payload type is only supported with GigE Vision Extended ID mode (i.e. *EI* field of the GVSP packet must be set to 1).

[CR-318st] If JPEG 2000 payload type is supported, then the transmitter MUST use the Extended ID mode for the stream channels associated to the JPEG 2000 payload type (including All-in Transmission).

25.8.3 JPEG 2000 Data Leader Packet

The data leader packet includes information that is not available from in the JPEG 2000 codestream.

[CR-319s] If JPEG 2000 payload is supported, its Data Leader packet MUST follow the layout of Figure 25-22.

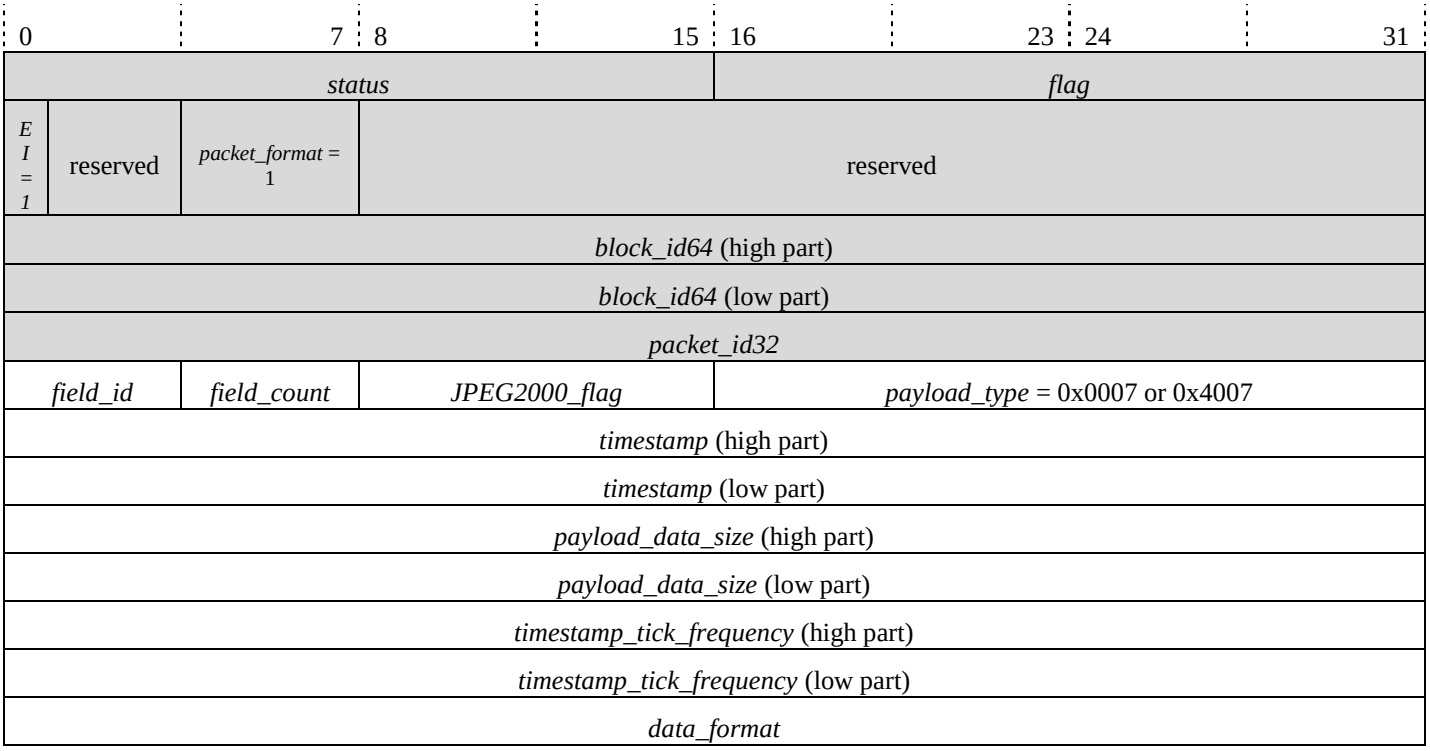


Figure 25-22: JPEG 2000 Data Leader Packet

JPEG 2000 DATA LEADER PACKET		
EI	1 bit	Extended ID. Must be set to 1 for JPEG 2000 payload type.
field_id	4 bits	See Image payload type.
field_count	4 bits	See Image payload type.

JPEG2000_flag	8 bits	bit 0 to 6 – Reserved. Set to 0 on transmission, ignore on reception. bit 7 (lsb) – color space selection: When clear, indicates that the *data_format* field contains a GigE Vision pixel format. When set, indicates that the *data_format* field uses a valid EnumCS color space, as defined by JPEG 2000.
payload_type	16 bits	Refer to Table 25-1.
timestamp	64 bits	See JPEG payload type.
payload_data_size	64 bits	See JPEG payload type.
timestamp_tick_frequency	64 bits	See JPEG payload type.
data_format	32 bits	See JPEG payload type.

25.8.4 JPEG 2000 Data Payload Packet

[CR-320s] If JPEG payload is supported, its Data Payload packet MUST follow layout of Figure 25-23.

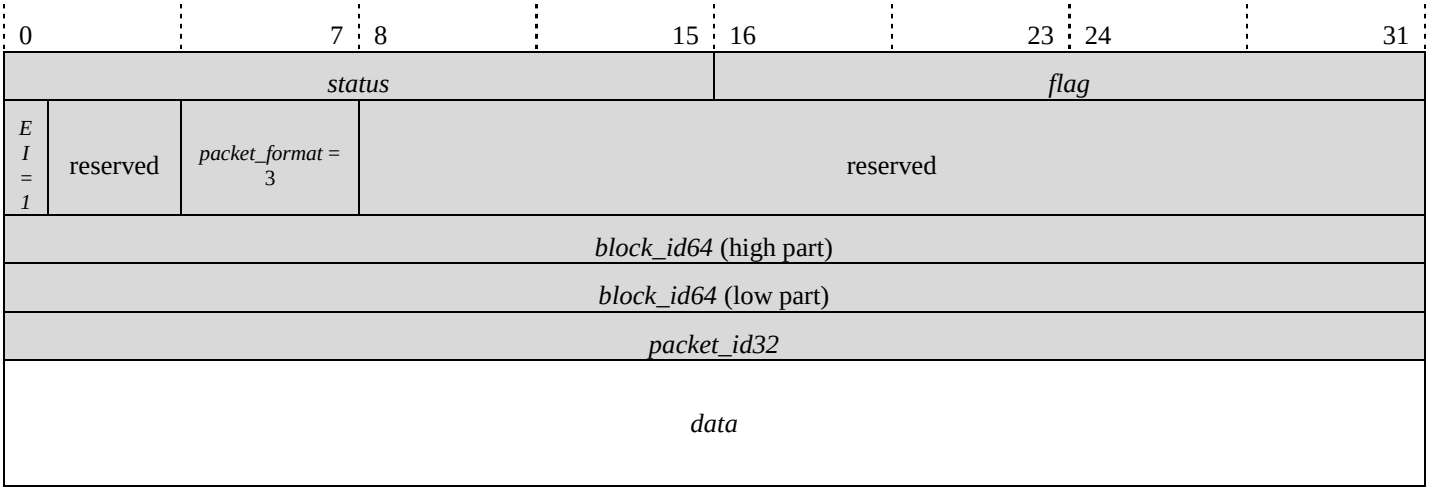


Figure 25-23: JPEG 2000 Data Payload Packet

JPEG 2000 DATA PAYLOAD PACKET		
EI	1 bit	Extended ID. Must be set to 1 for JPEG 2000 payload type.
data	up to packet size	Codestream data. The IP header + UDP header + GVSP header + data must be equal to the packet size specified in the Stream Channel Packet Size register of the GVSP transmitter. The only exception is the last Data Payload packet which may be smaller than the specified packet size. However, it is also possible to pad the last data packet so that all data payload packets are exactly the same size. The End Of Codestream (EOC) JPEG 2000 marker is used by the GVSP receiver to identify the end of the codestream.

25.8.5 JPEG 2000 Data Trailer Packet

[CR-321s] If JPEG 2000 payload is supported, its Data Trailer packet MUST follow the layout of Figure 25-24.

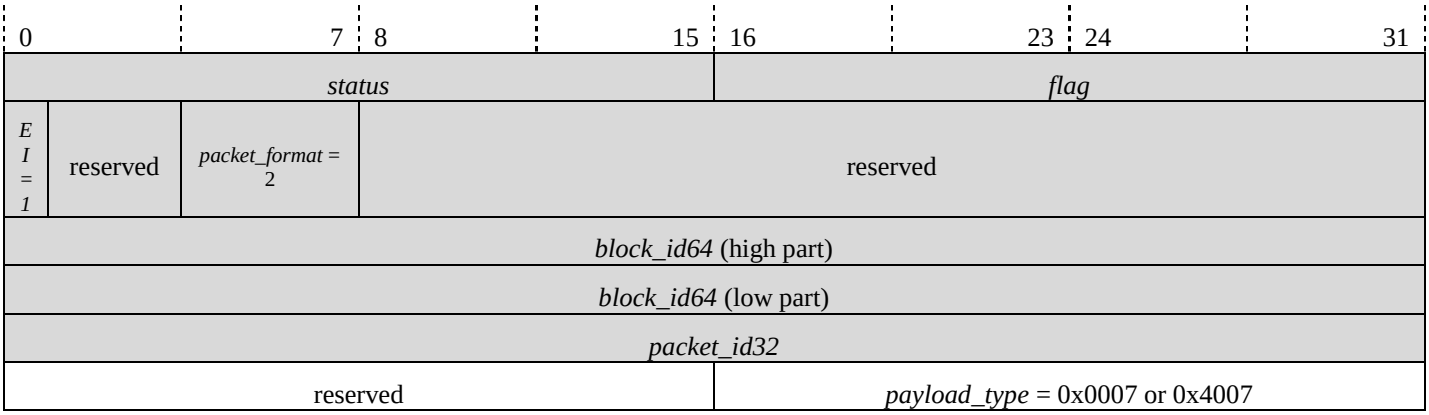


Figure 25-24: JPEG 2000 Data Trailer Packet

JPEG 2000 DATA TRAILER PACKET		
EI	1 bit	Extended ID. Must be set to 1 for JPEG 2000 payload type.
reserved	16 bits	Set to 0 on transmission, ignore on reception.
payload_type	16 bits	Refer to Table 25-1.

25.8.6 JPEG 2000 All-in Packet

[CR-322s] When All-in transmission mode is supported, if JPEG 2000 payload is supported, then its All-in packet MUST regroup the JPEG 2000 Data Leader, JPEG 2000 Data Trailer and JPEG 2000 Data Payload according to Figure 24-7.

25.9 H.264 Payload Type

This section describes a GVSP payload type for the ITU-T Rec. H.264, also known as MPEG-4 Part 10/MPEG-4 AVC. This H.264 payload type is inspired from RFC3984.

25.9.1 H.264 Principles

The H.264 specification defines two conceptual layers: a Video Coding Layer (VCL) and a Network Abstraction Layer (NAL).

The VCL contains the signal processing functionality of the codec. It defines mechanisms such as transform, quantization, motion compensated prediction and a loop filter. It follows the general concepts of most inter-frame video codecs i.e. a macroblock-based encoder that uses inter picture prediction with motion compensation and transform coding of the residual signal. A macroblock is a 16x16 block of luma

samples and two corresponding blocks of chroma samples of a picture that has three sample arrays, or a 16x16 block of samples of a monochrome picture or a picture that is coded using three separate color planes. The VCL encoder outputs slices: a bit stream that contains the macroblock data of an integer number of macroblocks and the information of the slice header (containing the spatial address of the first macroblock in the slice, the initial quantization parameter and similar information). Macroblocks in slices are generally arranged in scan order unless a different macroblock allocation is specified by using the Flexible Macroblock Ordering (FMO) syntax. In-picture prediction is used only within a slice.

The Network Abstraction Layer (NAL) encoder encapsulates the slice output of the VCL encoder into Network Abstraction Layer Units (NAL units), which are suitable for transmission over packet networks or use in packet oriented multiplex environments.

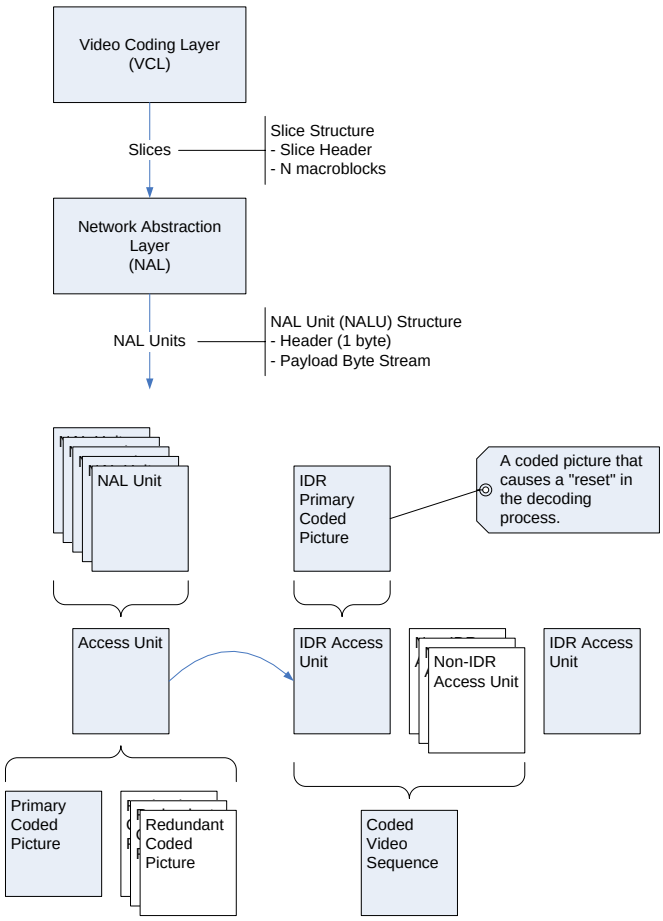


Figure 25-25 : H.264 Model

The H.264 standard defines an access unit as a set of NAL units always containing a primary coded picture. A primary coded picture contains all the macroblocks of a picture (image). In addition to the primary coded picture, an access unit may also contain one or more redundant coded pictures or other NAL units not containing slices or slice data partitions of a coded picture. The content of a redundant coded picture may be used by the decoding process for a bit stream that contains errors or losses.

A coded video sequence is a sequence of access units that consists, in decoding order, of an instantaneous decoding refresh (IDR) access unit followed by zero or more non-IDR access units including all subsequent access units up to but not including any subsequent IDR access unit. An IDR access unit is an access unit in which the primary coded picture is an IDR picture. An IDR picture is a coded picture containing only slices with I or SI slice types that causes a "reset" in the decoding process. After the decoding of an IDR picture, all following coded pictures in decoding order can be decoded without inter-prediction from any picture decoded prior to the IDR picture.

Note: There are apparently commercial video sequences that do not include IDR access units. These sequences commence with an I slice that does not cause a “reset” in the decoding process. Such sequences violate the H.264 standard but they exist and decoders generally cope with them.

25.9.2 H.264 Implementation for GVSP

This specification defines that the data block transports one access unit. This enables the transport of a number of NAL units in one block to optimize transport overhead. A GVSP data payload packet can carry a single NAL unit, more than one NAL unit or a fragment of one NAL unit. This is done in order to minimize GVSP transport overhead by optimizing the number of GVSP data payload packets being used to transport an access unit. The ramification of this scheme is that not all GVSP data payload packets may carry the same amount of data. This is different from most of the other payload types where only the last data payload packet might be smaller. As such, a H.264 receiver must be able to accommodate a gap after a NAL unit. This gap extends up to the start of the next packet. The receiver knows the location of the next packet after the end of a NAL unit using the default packet size provided by the SCPSx register. It parses the NAL data and skips those gaps.

This specification defines a method for a receiver to decode and display an H.264 encoded GVSP stream without having to know anything about the capabilities of the encoder. It does not rely on the implementation of capability announcement and offer/answer models. The information that is not available in the H.264 access unit is included in the GVSP data block. This may lead to the waste of some Ethernet bandwidth since additional information is to be delivered in each block as opposed to being auto negotiated before a streaming session is initiated.

The H.264 payload type is only supported with GigE Vision Extended ID mode (i.e. *EI* field of the GVSP packet must be set to 1).

[CR-323st] If H.264 payload type is supported, then the transmitter MUST use the Extended ID mode for the stream channels associated to the H.264 payload type (including All-in Transmission).

25.9.3 H.264 Data Leader Packet

The data leader packet includes information that is not available from in the H.264 access unit. It should be noted that the timestamp is not included since it is to be included in the data payload packets.

[CR-324s] If H.264 payload is supported, its Data Leader packet MUST follow the layout of Figure 25-26.

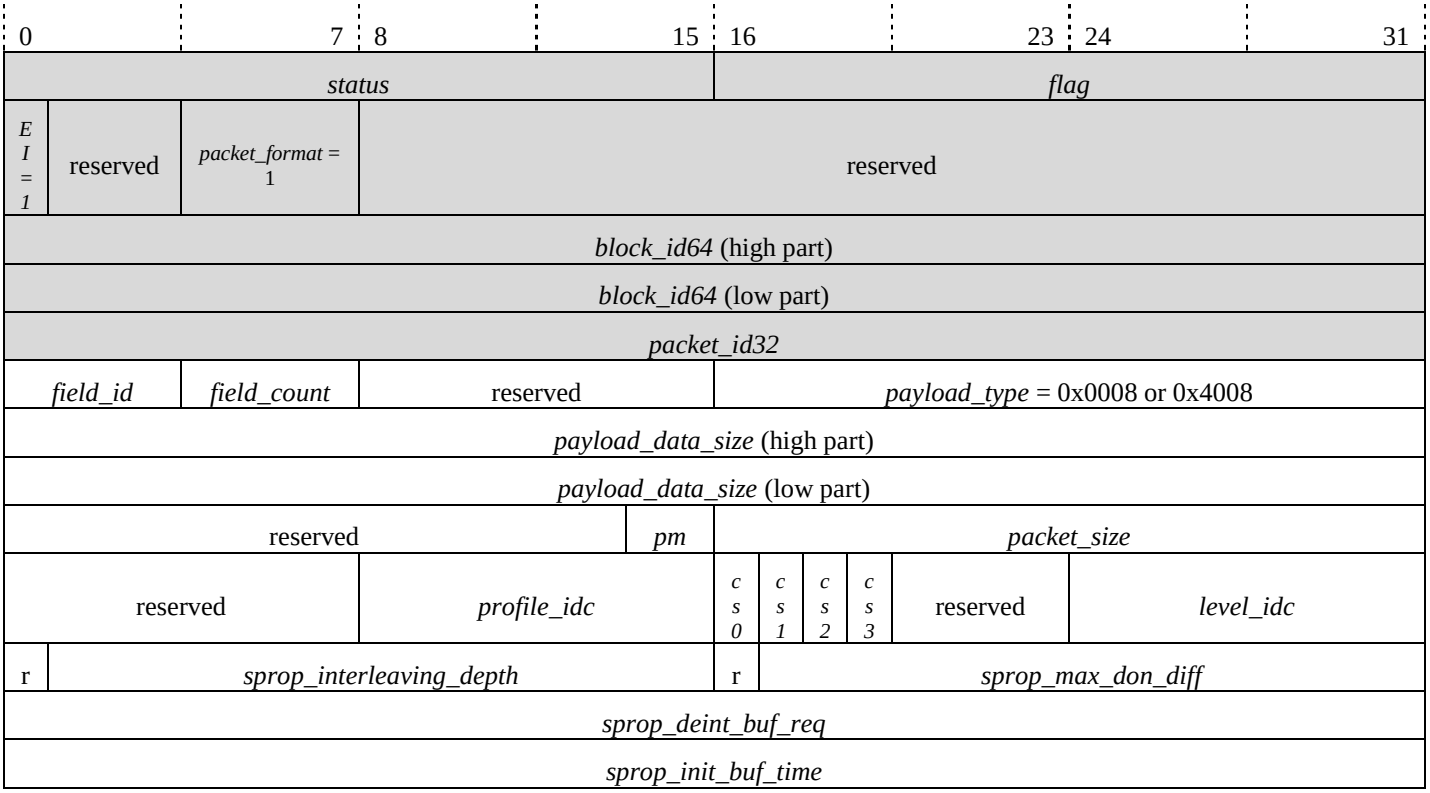


Figure 25-26: H.264 Data Leader Packet

H.264 DATA LEADER PACKET		
EI	1 bit	Extended ID. Must be set to 1 for H.264 payload type.
field_id	4 bits	See Image payload type.
field_count	4 bits	See Image payload type.
reserved	8 bits	Set to 0 on transmission, ignore on reception.
payload_type	16 bits	Refer to Table 25-1.
payload_data_size	64 bits	Maximum size of the access unit in bytes. It should be noted that this value is likely not representing the actual size of the GVSP block payload data but the upper limit of the amount of data that will be transmitted in a given block. This is because a GVSP transmitter will likely not know the maximum size of the compressed data when sending the leader down the link.
reserved	14 bits	Set to 0 on transmission, ignore on reception.

packetization_mode (pm)	2 bits	This parameter signals the properties of an H.264 GVSP payload type. When the value of packetization-mode is equal to zero, the single NAL mode, as defined in section 6.2 of RFC3984, is used. When the value of packetization-mode is equal to one, the non-interleaved mode, as defined in section 6.3 of RFC3984, is used. When the value of packetization-mode is equal to two, the interleaved mode, as defined in section 6.4 of RFC3984, is used. Other values are reserved.
packet_size	16 bits	The packet size specified in the Stream Channel Packet Size register of the GVSP transmitter. This information is provided to help the receiver to perform packet reordering since data payload packets may be of different size.
reserved	8 bits	Set to 0 on transmission, ignore on reception.
profile_idc	8 bits	profile_idc sequence parameter set data attribute as defined by H.264.
cs0	1 bit	constrant_set0_flag sequence parameter set data attribute as defined by H.264.
cs1	1 bit	constrant_set1_flag sequence parameter set data attribute as defined by H.264.
cs2	1 bit	constrant_set2_flag sequence parameter set data attribute as defined by H.264.
cs3	1 bit	constrant_set3_flag sequence parameter set data attribute as defined by H.264.
reserved	4 bits	Set to 0 on transmission, ignore on reception.
level_idc	8 bits	level_idc sequence parameter set data as defined by H.264.
reserved (r)	1 bit	Set to 0 on transmission, ignore on reception.
sprop_interleaving_depth	15 bits	This parameter may be used to signal the properties of a NAL unit stream. This parameter is not applicable if the value of *packetization_mode* is equal to zero or one. Otherwise, it specifies the maximum number of VCL NAL units that precede any VCL NAL unit in the NAL unit stream in transmission order and follow the VCL NAL unit in decoding order as per RFC3984.
reserved (r)	1 bit	Set to 0 on transmission, ignore on reception.
sprop_max_don_diff	15 bits	This parameter may be used to signal the properties of a NAL unit stream. This parameter is not applicable if the value of *packetization_mode* is equal to zero or one. Otherwise, *sprop_max_don_diff* is an integer in the range of 0 to 32767 and it is used in the NAL unit de-interleaving process as per RFC3984.
sprop_deint_buf_req	32 bits	This parameter may be used to signal the properties of a NAL unit stream. This parameter is not applicable if the value of *packetization_mode* is equal to zero or one. Otherwise, *sprop_deint_buf_req* signals the required size of the de-interleaving buffer for the NAL unit stream as per RFC3984.
sprop_init_buf_time	32 bits	This parameter may be used to signal the properties of a NAL unit stream. This parameter is not applicable if the value of *packetization_mode* is equal to zero or one. Otherwise, the parameter signals the initial buffering time that a receiver must buffer before starting decoding to recover the NAL unit decoding order from the transmission order as per RFC3984.

Note: H.264 is unlikely to carry interlaced video. As such, *field_id* and *field_count* are likely to be 0 for this payload type.

25.9.4 H.264 Data Payload Packet

[CR-325s] If H.264 payload is supported, its Data Payload packet MUST follow layout of Figure 25-27.

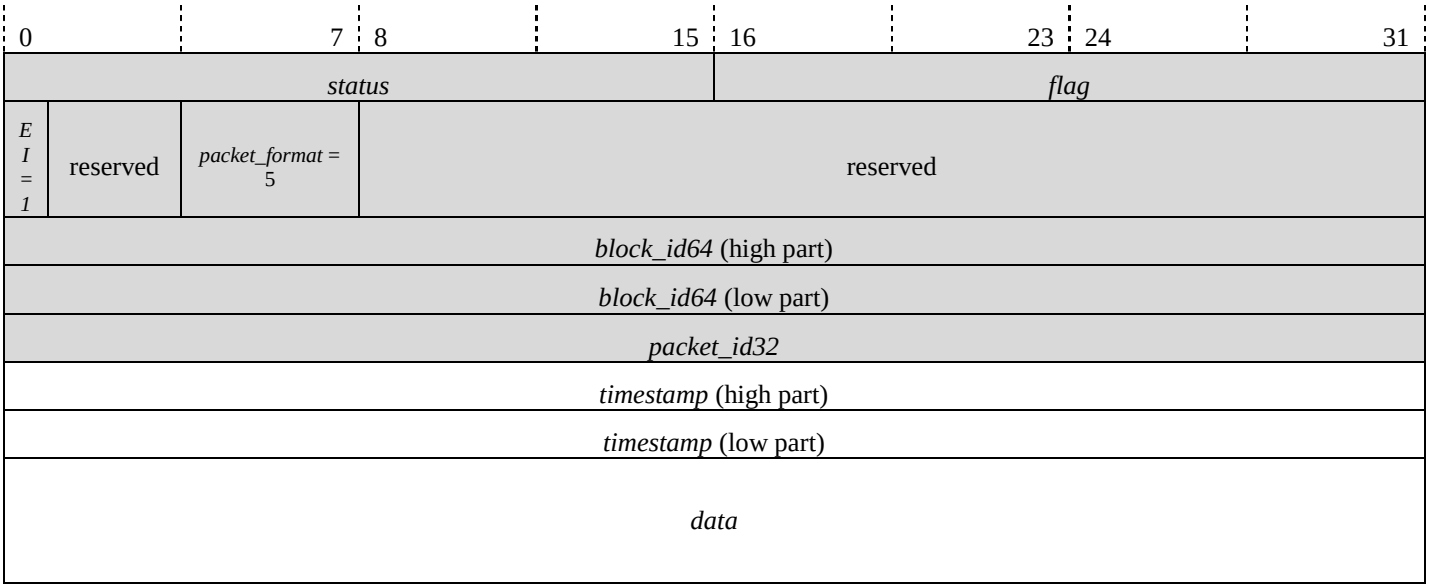


Figure 25-27: H.264 Data Payload Packet

H.264 DATA PAYLOAD PACKET		
EI	1 bit	Extended ID. Must be set to 1 for H.264 payload type.
timestamp	64 bits	Indicates the sampling timestamp of the content of the packet. It needs to be expressed in units of a 90 kHz clock rate to be compliant with the H.264 standard. Note: For H.264, the *timestamp* field is not in the data leader, but part of the data payload since one access unit can include data from more than one frame.
data	up to packet size	Access unit data. Includes a single NAL unit, more than one NAL unit or a fragment of one NAL unit. The IP header + UDP header + GVSP header (including the timestamp high and low parts) + data must be equal or less than the packet size specified in the Stream Channel Packet Size register of the GVSP transmitter. However, it is also possible to pad the data packet so that all data payload packets are exactly the same size.

The following figures present the layout of the *data* field of the H.264 data payload packets as per the encapsulation method provided in RFC3984. For more details, refer to this RFC and to the H.264 standard.

Single NAL Unit Packet

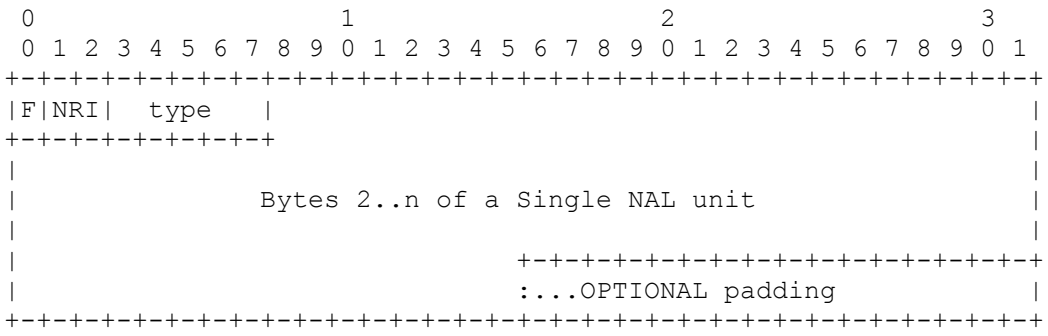


Figure 25-28 : Data Payload Format for Single NAL Unit Packet

Note that the first byte (*F*, *NRI* and *type* fields) is the 1-byte NAL unit header from Figure 25-25. The *type* field specifies the NAL unit payload type.

Aggregation Packets

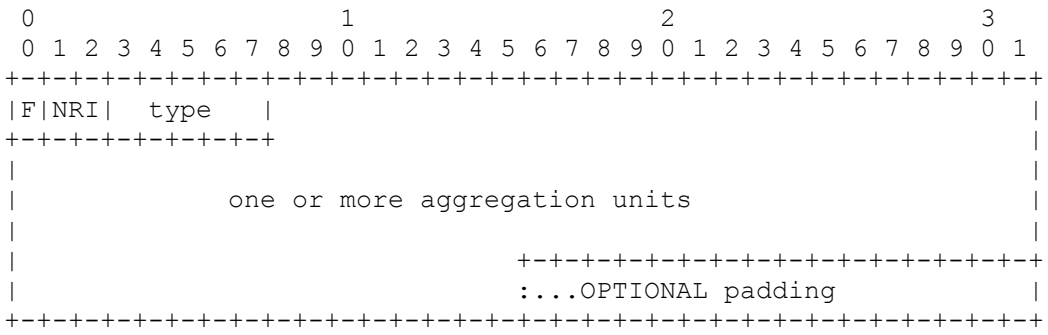


Figure 25-29 : Data Payload Format for Aggregation Packets

Single-Time Aggregation Packets (STAPs)

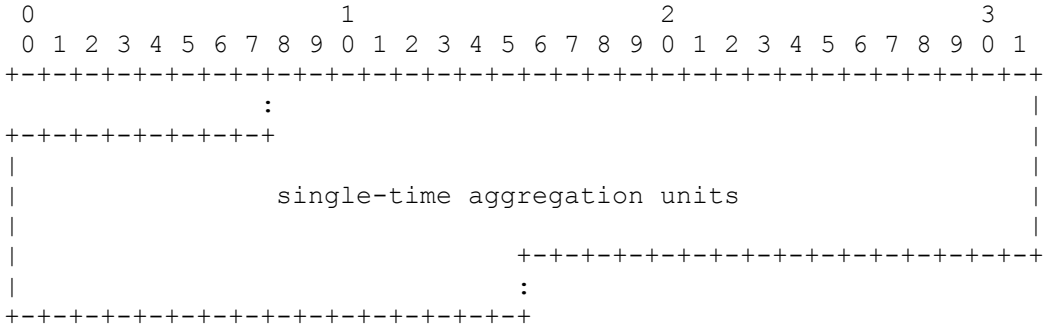


Figure 25-30 : Data Payload Format for STAP-A

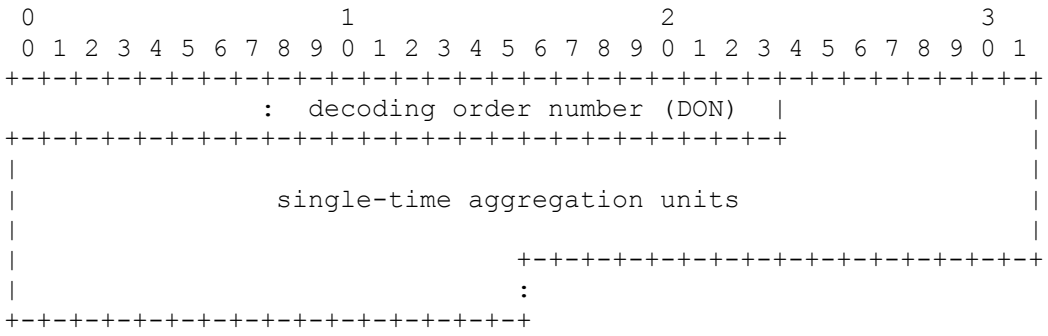


Figure 25-31 : Data Payload Format for STAP-B

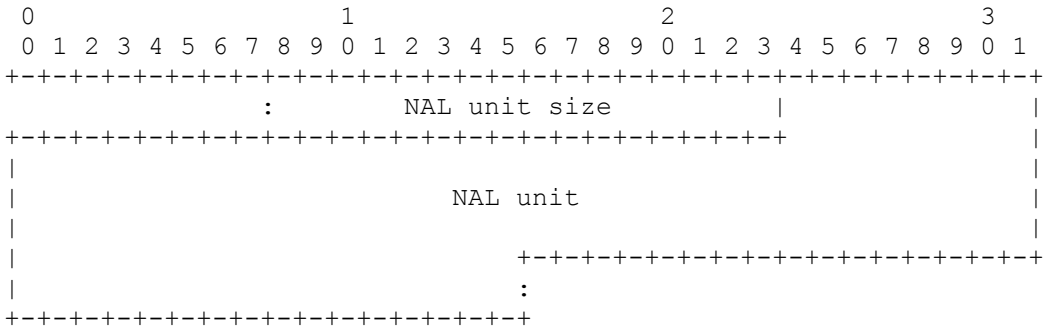


Figure 25-32 : Structure for Single-time Aggregation Unit

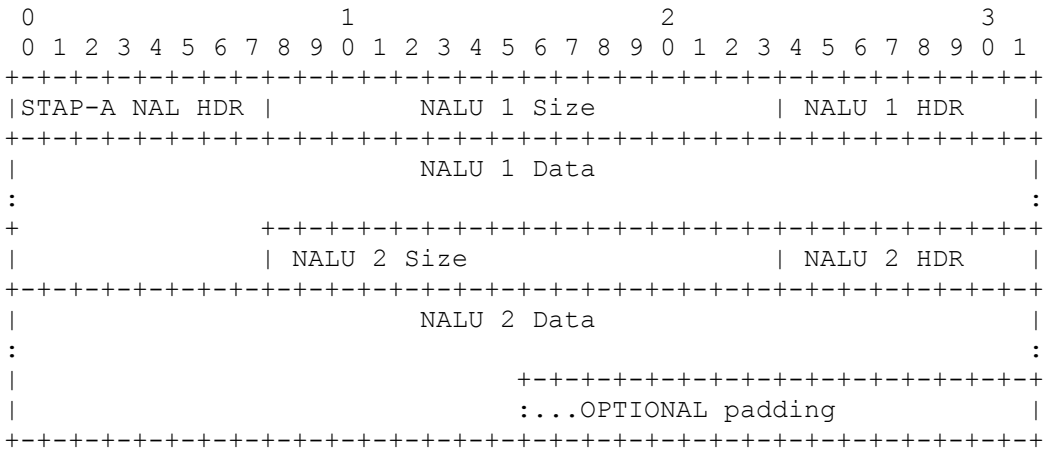


Figure 25-33 : An example of a data payload including a STAP-A and two single-time aggregation units

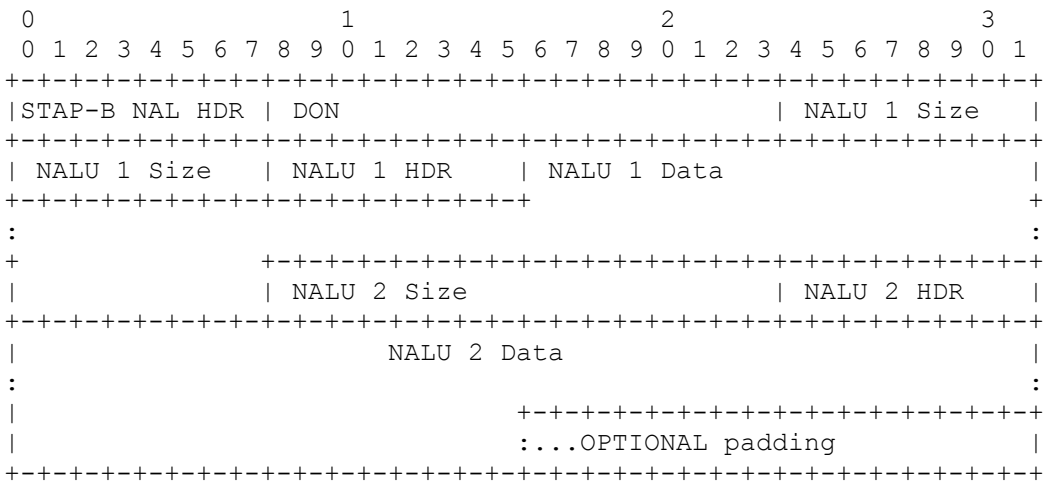


Figure 25-34 : An example of a data payload including a STAP-B and two single-tim aggregation units

Multi-Time Aggregation Packets (MTAPs)

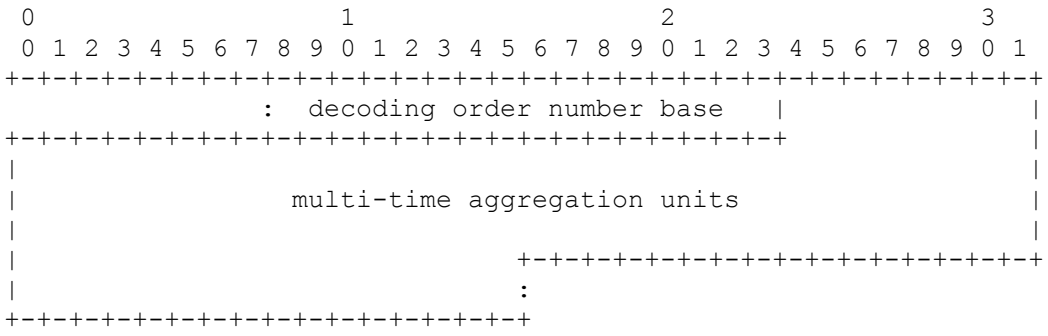


Figure 25-35 : NAL Unit Payload Format for MTAPs

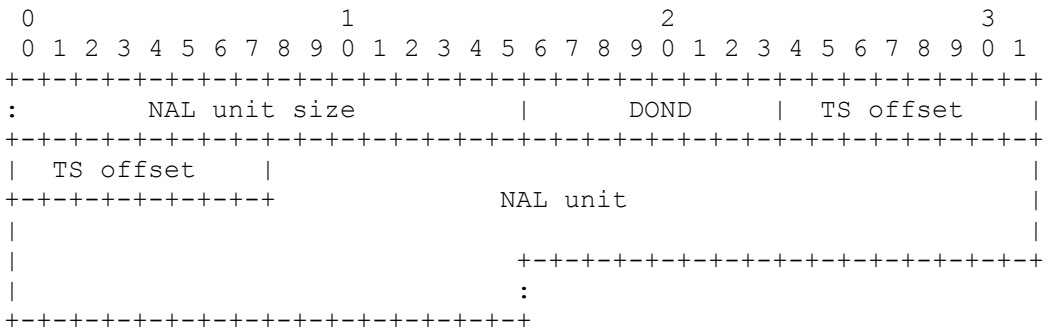


Figure 25-36 : Multi-time Aggregation Unit for MTAP16

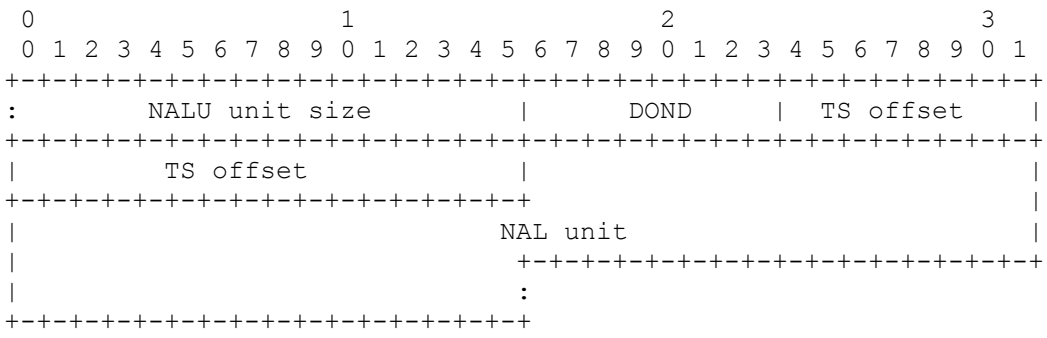


Figure 25-37 : Multi-time Aggregation Unit for MTAP24

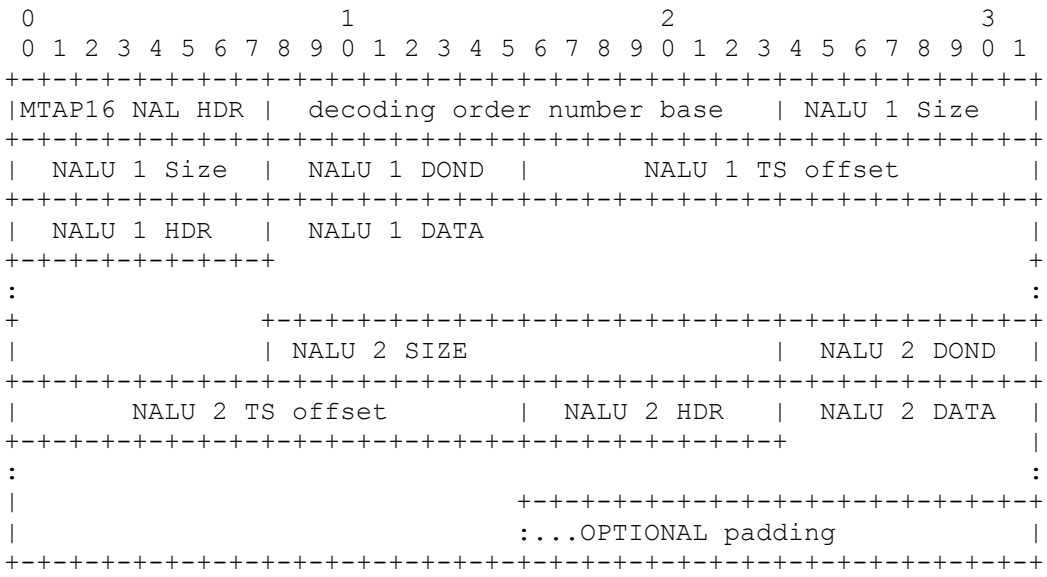


Figure 25-38 : A data payload including a multi-time aggregation packet of type MTAP16 and two multi-time aggregation units

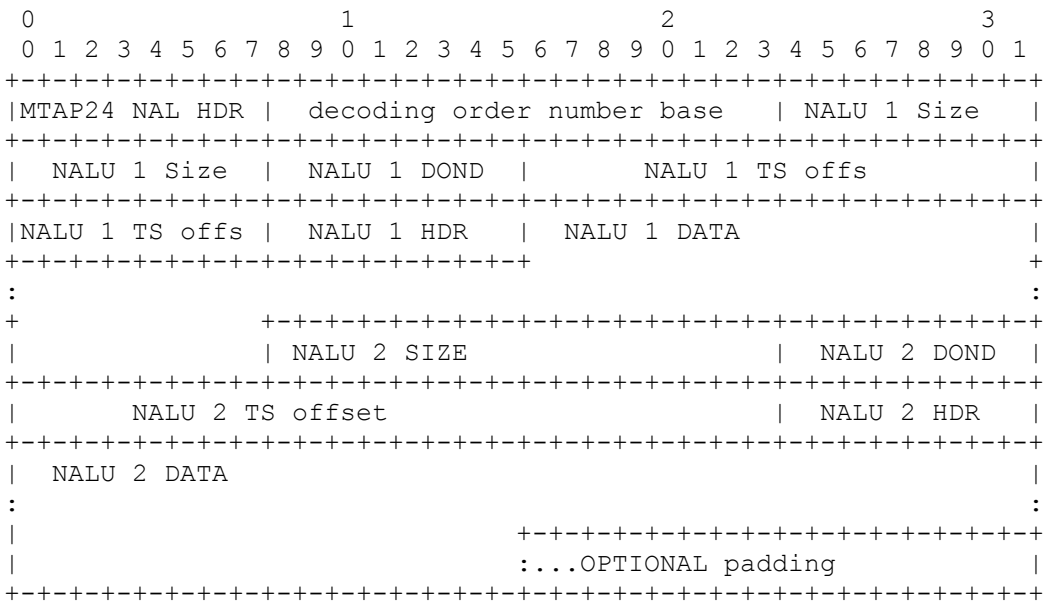


Figure 25-39 : A data payload including a multi-time aggregation packet of type MTAP24 and two multi-time aggregation units

Fragmentation Units (FUs)

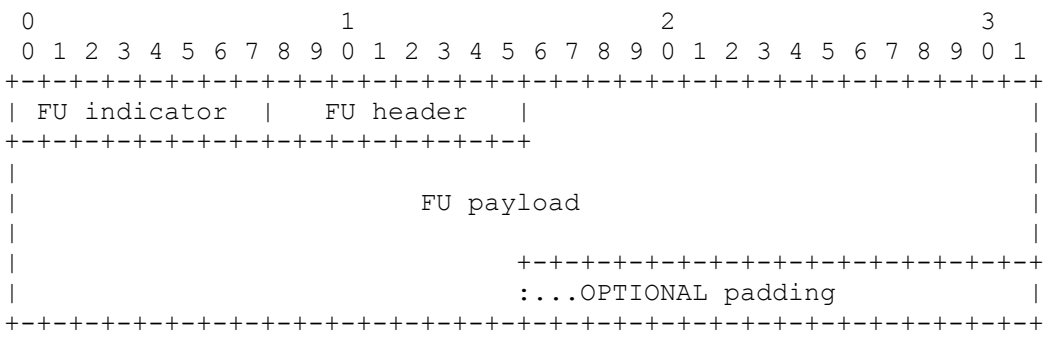


Figure 25-40 : Data Payload Format for FU-A

The FU indicator in the previous figure is a NAL unit byte header. The FU header includes bits to identify the start and the end of a fragment of a NAL unit.

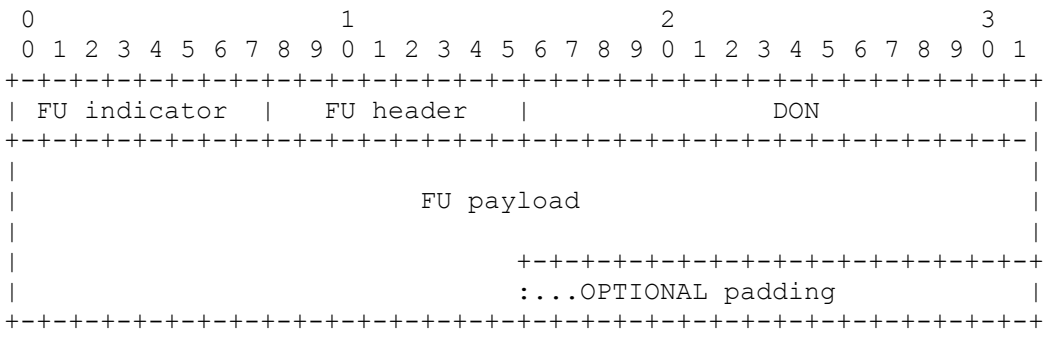


Figure 25-41 : Data Payload Format for FU-B

25.9.5 H.264 Data Trailer Packet

[CR-326s] If H.264 payload is supported, its Data Trailer packet MUST follow the layout of Figure 25-42.

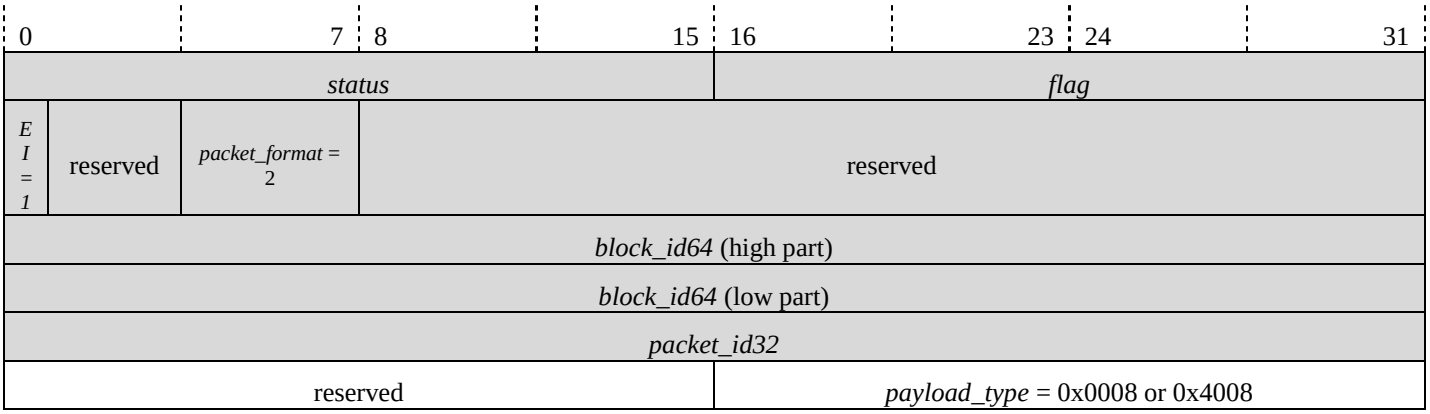


Figure 25-42: H.264 Data Trailer Packet

H.264 DATA TRAILER PACKET		
EI	1 bit	Extended ID. Must be set to 1 for H.264 payload type.
reserved	16 bits	Set to 0 on transmission, ignore on reception.
payload_type	16 bits	Refer to Table 25-1.

25.9.6 H.264 All-in Packet

[CR-327s] When All-in transmission mode is supported, if H.264 payload is supported, then its All-in packet MUST regroup the H.264 Data Leader, H.264 Data Trailer and H.264 Data Payload according to Figure 24-7.

25.10 Multi-zone Image Payload Type

As indicated previously, the Image payload type available since version 1.0 of this specification deals with raster-scan transmission of the image data. This is an ideal fit for cameras where the sensor outputs the data on a single tap. Otherwise, the camera must first reconstruct the image in raster-scan format before transmission. This could introduce some additional latency and create peaks of transmission bandwidth. To address this situation, GigE Vision version 2.0 introduces the optional Multi-zone Image payload type.

The intention of the Multi-zone Image payload type is to improve the latency characteristics for the transport of image data coming from multiple taps sensors by dividing the image into horizontal bands. Each of these horizontal bands is called a “zone”. The transmitter is allowed to start the transmission of data from a zone as soon as the zone’s next packet is ready.

Multi-zone Image is especially useful to support sensors featuring a number of taps read top-down and a number of others read bottom-up.

25.10.1 Multi-zone Image Principles

The main objective of the Multi-zone Image payload type is to improve the support of multi-tap sensors with taps starting at different vertical locations within the image. For instance, a 2-tap sensor could have one tap starting at the top of the image and a second tap starting from the bottom. By dividing the image into horizontal zones (see Figure 25-43 below), the camera is able to start the transmission of data from each of these zones earlier, reducing overall latency and internal image buffering.

Note: A camera hosting a sensor with horizontal taps needs to reconstruct the image lines locally in its on-board memory since it is not permitted to divide image in vertical bands for multi-zone image payload type.

[CR-328st] If Multi-zone Image payload type is supported, then the transmitter MUST create the zones by slicing the full image into horizontal bands. It is not permitted to have vertical bands.

Zones are numbered sequentially from 0 to N-1, with no gap, starting at the top of the image (where N is the total number of zones in the image).

[CR-329st] If Multi-zone Image payload type is supported, then the zones MUST be indexed, starting from the top, with a zone ID of 0 and increment moving toward the bottom of the image until the last zone. It is not permitted to leave a gap in the zone ID sequence.

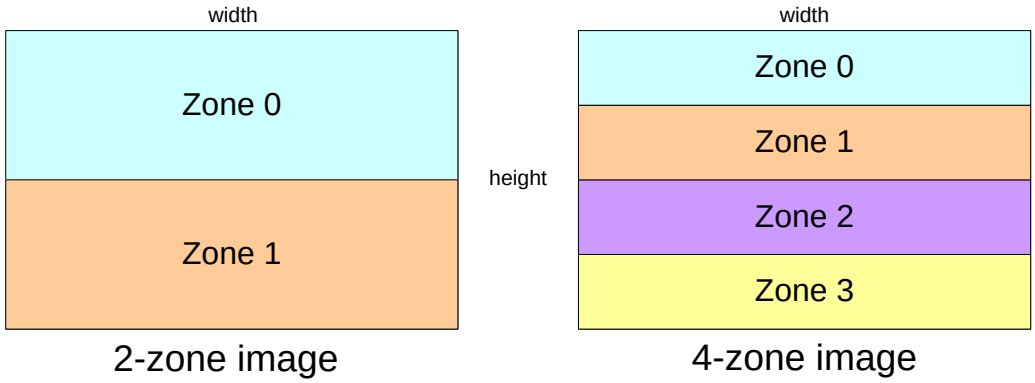


Figure 25-43 : Two and Four Zone Images

Each zone might have a different height: this allows a ROI to be positioned anywhere on the full size image and to still associate a sensor vertical tap to each zone. The transmitter typically iterates (i.e. round-robin) between zones when transmitting data, but not necessarily. Packets within a zone are transmitted sequentially top-down or bottom-up. This helps the receiver predict the order of arrival of image data within a zone.

[CR-330st] If Multi-zone Image payload type is supported, then the transmitter MUST transmit the packets associated to each zone sequentially, either top-down or bottom-up.

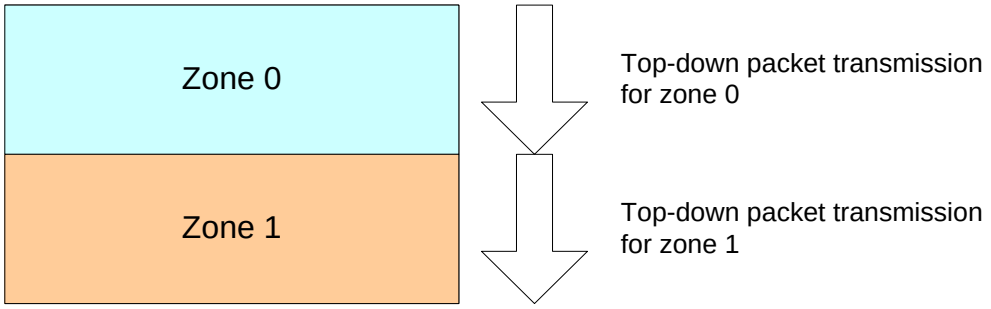


Figure 25-44 : Top-down Only Packet Transmission

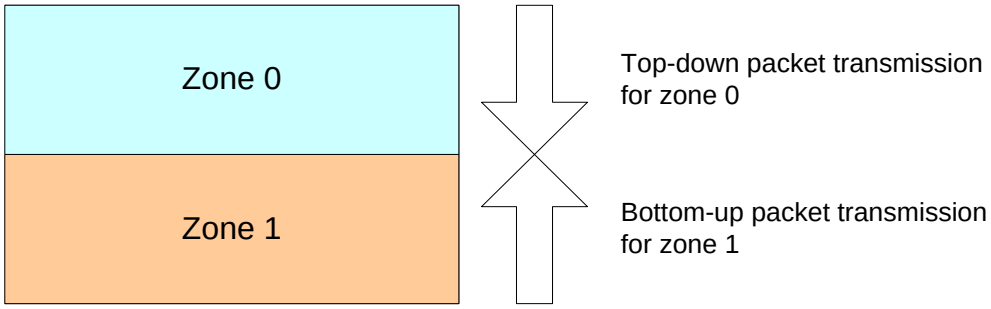


Figure 25-45 : Top-down and Bottom-up Packet Transmission

The content of each data packet is raster-scan to facilitate the copy of data on the receiver side.

[CR-331st] If Multi-zone Image payload type is supported, then the transmitter MUST present the pixel content of the Multi-zone Image data payload packet in raster-scan format.

A zone is typically created by grouping sensor taps on the same horizontal line together (i.e. horizontal taps) and separating the sensor taps coming from different lines of the image (i.e. vertical taps) into different zones. Typically, there is one zone associated to each of the sensor vertical tap (or group of horizontal taps) present in the ROI from which data is transmitted. The burden to re-order the pixel data sequentially from left to right from those horizontal taps sitting on the same line is left to the transmitter.

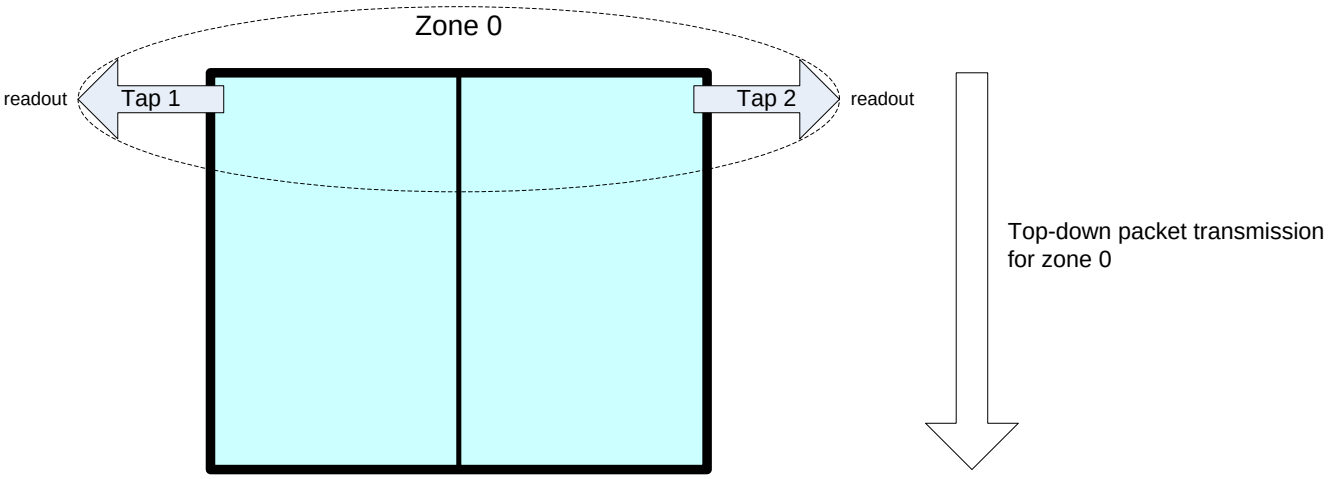


Figure 25-46 : Sensor with 2 Horizontal Taps Grouped into One Zone (Top-down Transmission)

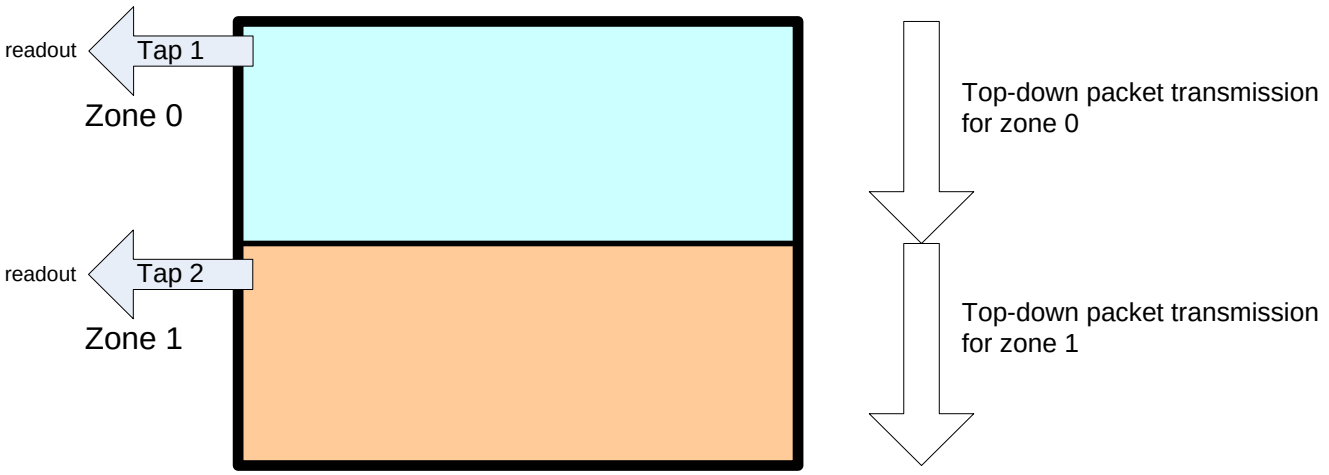


Figure 25-47 : Sensor with 2 Vertical Taps Leading to Two Zones (Top-down Transmission)

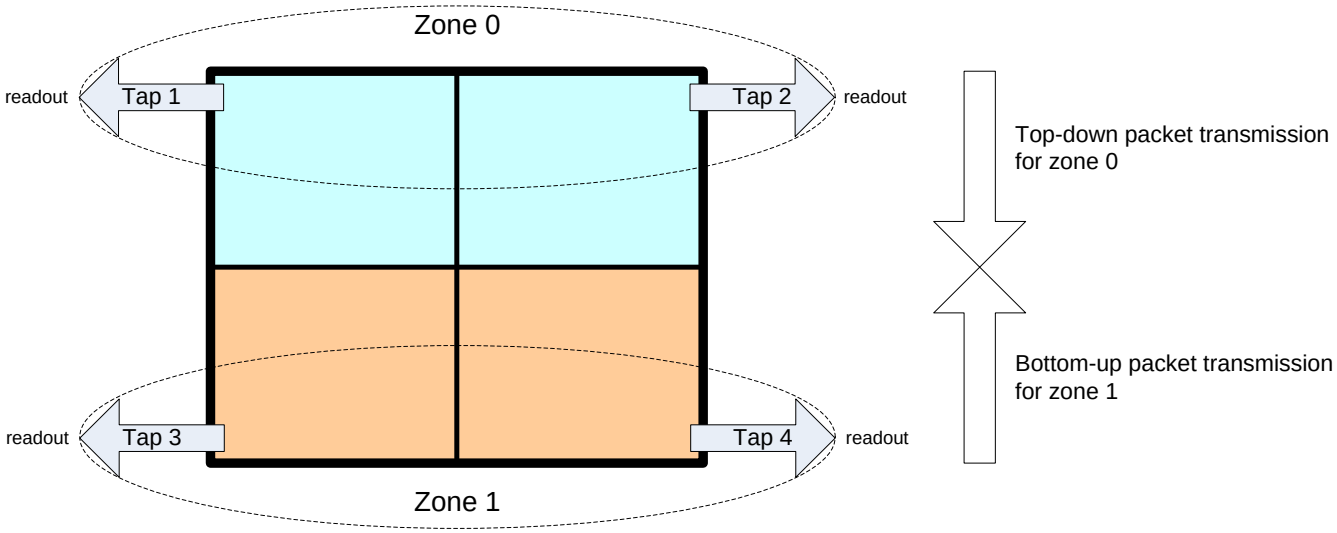


Figure 25-48 : 4-tap Sensor Leading to Two Zones (Top-down and Bottom-up Transmission)

It is worth highlighting that data from a given vertical tap may be transmitted as part of an adjacent zone. For example, let's assume a situation where two sensor taps across the vertical direction contribute to a given ROI and two zones are used to transport the data over the stream channel.

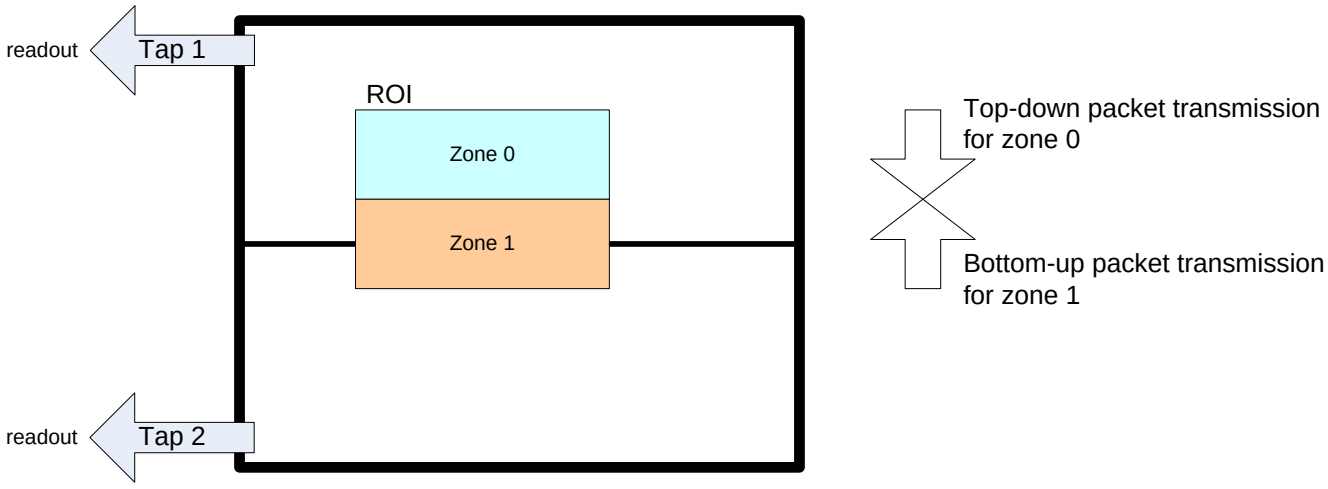


Figure 25-49 : 2-tap Sensor with ROI

In this case, some data from the top sensor tap may be carried as part of the bottom zone (zone ID 1) in the same packet as data coming from the bottom sensor tap. That can be done in order to maximize transmission's efficiency by including data of two adjacent taps in a given zone. It is technically possible for a transmitter to use more zones than the number of vertical taps presents in the ROI but there is no anticipated technical advantage to do so.

25.10.2 Multi-zone Image Implementation for GVSP

In GVSP, Multi-zone Image payload type provides an address offset at which the data must be deposited in the destination buffer. Contrary to the packet ID, this address offset does not need to be sequential hence allowing image data to be sent "out of order" (i.e. switching between zones) to the destination buffer. Note that it is important to maintain sequential packet IDs to simplify the packet resend logic at the receiver. Even though data can be transmitted bottom-up, the corresponding packet ID still increments. This decoupling of packet ID from the data address offset provides a better encapsulation of the transport layer as it is not bound to the image data location. Within a zone, the data must be sent sequentially (i.e. in-order), top-down or bottom-up.

GigE Vision allows up to 32 zones (numbered from 0 to N-1) to be defined on a given stream. The number of zones and their direction is reported in the data leader packet and in the SCZx and SCZDx bootstrap registers. This enables Multi-zone Image data blocks to be self-described.

[CR-332st] If Multi-zone Image payload type is supported, then the transmitter MUST not provide more than 32 zones per stream channel.

Note: As indicted in section 25.1, the Extended Chunk realization of the Multi-zone Image payload type is achieved by introducing an additional top-down zone to carry all the chunks after the Multi-zone Image.

The *end_of_zone* and *zone_direction* fields are present in the Multi-zone Image data payload and data leader packets respectively to help the GVSP receiver simplify the way it manages incoming data packets. But a receiver can ignore them and simply use the *address_offset* field and the UDP packet size to copy data to image buffer.

A feature in the XML device description file can enable the application to configure GVSP transmitters not to change the number of zones and their direction during block acquisition and transmission. This enables an application and its associated GVSP receivers to pre-allocate resources to optimize GVSP de-encapsulation.

The last data packet of a zone is tagged with an *end_of_zone* flag, with the possible exception of variable image length where the transmitter might not know at transmission time that this is the last packet of the zone. This last data packet of a zone might be smaller than the other data packets. It must not be padded to avoid padding data to be copied over actual image data.

[CR-333st] If Multi-zone Image payload type is supported, then all data payload packets MUST use the size of the SCPSx bootstrap register except for the last data payload packet of a zone which can be smaller.

[CR-334st] If Multi-zone Image payload type is supported, then the transmitter MUST not pad the last data payload packet associated to a given zone.

Note: The above conditional requirement is necessary to prevent the last packet of a zone to overwrite data that belongs to another adjacent zone. In some situations, it might be necessary for the last grouped or packed pixel of a zone to include pixel information from the first pixel of an adjacent zone to avoid "line padding". This is because GigE Vision only permits "image padding" as defined in the Pixel Format Naming Convention. Refer to section 26.2 for additional information.

The Multi-zone Image payload type is only supported with GigE Vision Extended ID mode (i.e. *EI* field of the GVSP packet must be set to 1).

[CR-335st] If Multi-zone Image payload type is supported, then the transmitter MUST use the Extended ID mode for the stream channels associated to the Multi-zone Image payload type (including All-in Transmission).

It is mandatory for the transmitter to send packets in order within a zone (top-down or bottom up). This enables a GVSP receiver to predict the order of arrival of image data within a zone. Also, a GVSP transmitter typically iterates in round robin fashion between zones until all zones are fully transmitted but this behavior is not mandatory.

A GVSP transmitter can elect to use the Multi-zone Image payload type to transmit a single top-down raster-scan image. In this case, the following conditions apply:

1. Number of zones = 1
2. Zone direction = top-down
3. Zone ID = 0
4. End-of-zone = don't care
5. *address_offset* = successive address of the first pixel of each packet

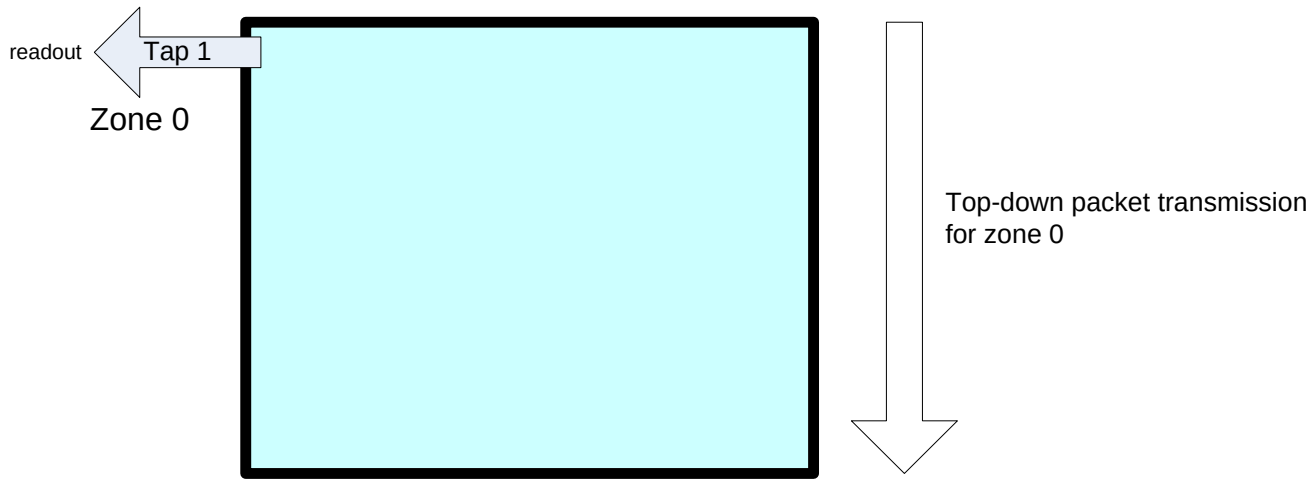


Figure 25-50 : Raster-scan Image Using Multi-zone Image Payload Type

As indicated in section 25.1, if Extended Chunk mode is used, then the previous example will require 2 zones: the first zone to carry the raster-scan image (i.e. the first chunk); the second zone for the chunk tags of the first chunk as well as for all additional chunks.

25.10.3 Multi-zone Image Data Leader Packet

The data leader of this payload type reports the number of zones present in the block (up to 32 zones) and the transmission direction of each zone (top-down or bottom-up). The transmission direction of each zone is independent of each other.

[CR-336s] If Multi-zone Image payload is supported, its Data Leader packet MUST follow layout of Figure 25-51, which reuses many fields from the Image payload type.

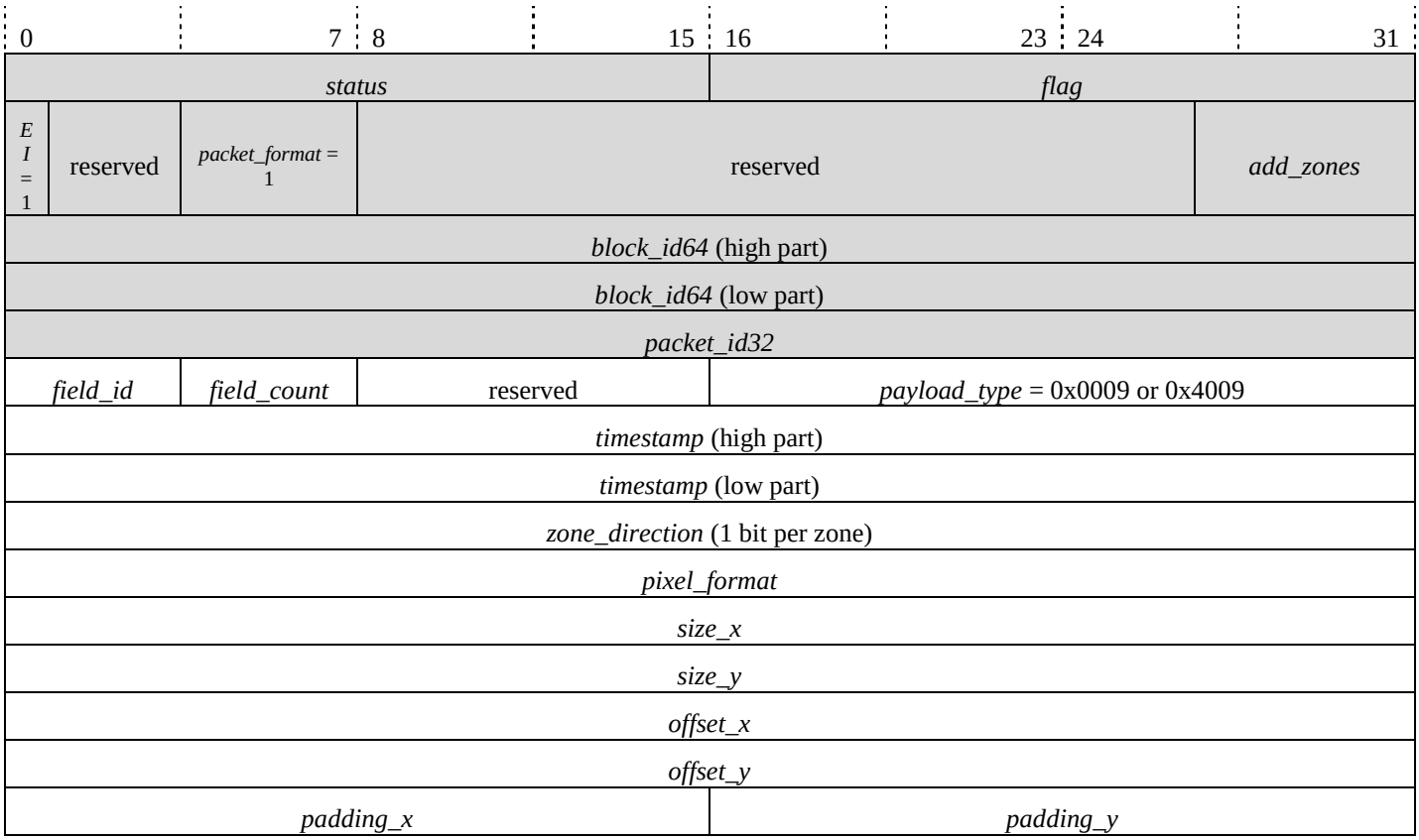


Figure 25-51: Multi-zone Image Data Leader Packet

MULTI-ZONE IMAGE DATA LEADER PACKET		
EI	1 bit	Extended ID. Must be set to 1 for Multi-zone Image payload type.
reserved	3 bits	Set to 0 on transmission, ignore on reception.
packet_format	4 bits	See section 24.2 - Data Block Packet Header
reserved	19 bits	Set to 0 on transmission, ignore on reception.

add_zone	5 bits	Reports the number of additional zones in the block. The number of zones (horizontal bands) is equal to the value of this field plus one (0 indexed value).
field_id	4 bits	See Image payload type.
field_count	4 bits	See Image payload type.
reserved	8 bits	Set to 0 on transmission, ignore on reception.
payload_type	16 bits	Refer to Table 25-1.
timestamp	64 bits	See Image payload type.
zone_direction	32 bits	Reports the transmission direction of each zone. bit 0 (msb) - Direction of zone 0. Reports the transmission direction of zone ID 0. When set to zero, the zone is transmitted top-down. Otherwise, it is transmitted bottom up. bit 1 - Direction of zone 1. Reports the transmission direction of zone ID 1. When set to zero, the zone is transmitted top-down. Otherwise, it is transmitted bottom up. … bit 31 (lsb) - Direction of zone 31. Reports the transmission direction of zone ID 31. When set to zero, the zone is transmitted top-down. Otherwise, it is transmitted bottom up.
pixel_format	32 bits	See Image payload type.
size_x	32 bits	See Image payload type.
size_y	32 bits	See Image payload type.
offset_x	32 bits	See Image payload type.
offset_y	32 bits	See Image payload type.
padding_x	16 bits	See Image payload type.
padding_y	16 bits	See Image payload type.

25.10.4 Multi-zone Image Data Payload Packet

It should be noted that the Multi-zone Image payload type uses a specific packet format for its image data payload packets. This allows to uniquely identify Multi-zone Image data payload packets by looking at the packet header.

[CR-337s] If Multi-zone Image payload is supported, its Data Payload packet MUST follow layout of Figure 25-52.

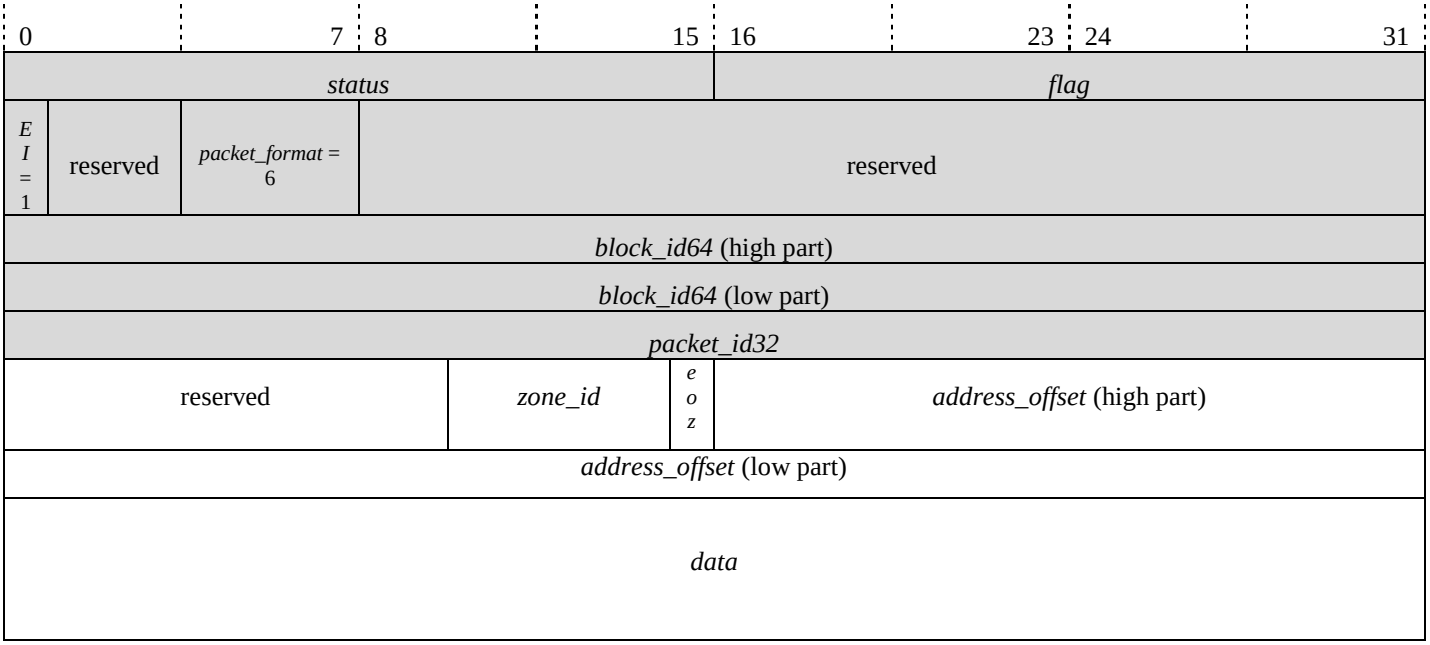


Figure 25-52: Multi-zone Image Data Payload Packet

MULTI-ZONE IMAGE DATA PAYLOAD PACKET		
EI	1 bit	Extended ID. Must be set to 1 for Multi-zone Image payload type.
reserved	10 bits	Set to 0 on transmission, ignore on reception.
zone_id	5 bits	Indicates the ID of the zone from which the image data is coming from in this packet. *zone_id* is set to 0 for the top zone of the block, it is set to one for the second zone from the top and so on.
end_of_zone (eoz)	1 bit	When more than one zone is present, set to one by the transmitter when it knows this is the last data payload packet of a zone. For a single zone image, this field is don't care. For the variable image height scenario where the GVSP transmitter does not know this is the last packet of a zone, it is acceptable for the GVSP transmitter to leave this bit cleared even though it should have been one. In this case, the *size_y* field of the data trailer reports the actual image height.
address_offset	48 bits	Provides the absolute address (in bytes) of the data of the data payload packet from the start of the block.
data	up to packet size	Image data formatted as specified in the Data Leader *pixel_format* field. For Data Payload packets, the IP Header + UDP Header + GVSP Header + data must be equal to the packet size specified in the Stream Channel Packet Size register of the GVSP transmitter. The only exception is the last Data Payload packet of a zone which may be smaller than the specified packet size. However, the last data packet of a zone must **not** be padded.

25.10.5 Multi-zone Image Data Trailer Packet

[CR-338s] If Multi-zone Image payload is supported, its Data Trailer packet MUST follow the layout of Figure 25-53.

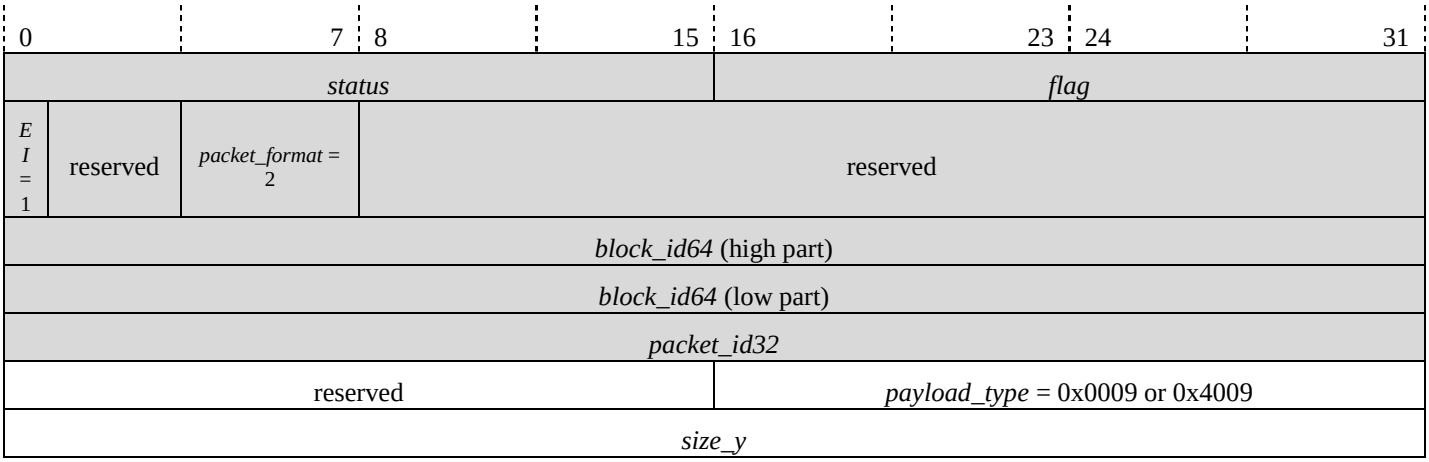


Figure 25-53: Multi-zone Image Data Trailer Packet

IMAGE DATA TRAILER PACKET		
EI	1 bit	Extended ID. Must be set to 1 for Multi-zone Image payload type.
reserved	16 bits	Set to 0 on transmission, ignore on reception.
payload_type	16 bits	Refer to Table 25-1.
size y	32 bits	See Image payload type.

25.10.6 Multi-zone Image All-in Packet

By definition, Multi-zone Image cannot support All-in Packet since they require at least one data packet per zone. Therefore, it is not possible to use All-in Transmission mode for Multi-zone Image payload type. Use Image payload type if All-in Packet is needed for images.

25.10.7 Multi-zone Image Examples

The following examples illustrate the realization of Multi-zone Image for different scenarios.

Example 1: 2 top-down zones with no ROI

This first example shows a scenario with 2 vertical taps, both top-down. Two zones (Zone 0 and Zone 1) are needed. The first zone starts at the top of the image while the second starts in the middle. Transmission of data packets is round-robin between the two zones. Within a packet, pixels are always transmitted from pixel A to pixel Z (see figure below) in raster-scan format. Note that the last data packet of each zone, tagged with the *end_of_zone* flag, might be smaller and must not be padded with any additional padding bytes. Also, considerations should be given to avoid line padding (see section 26.2). Therefore, the last pixel

of a zone might need to be combined with the first pixel of the adjacent zone for grouped or packed pixel formats.

Nominal image size	16 columns x 8 lines
Pixel size	1 byte
Number of zones	2
Packet size (payload)	24 bytes
Zone 0	top-down, 16 x 4
Zone 1	top-down, 16 x 4

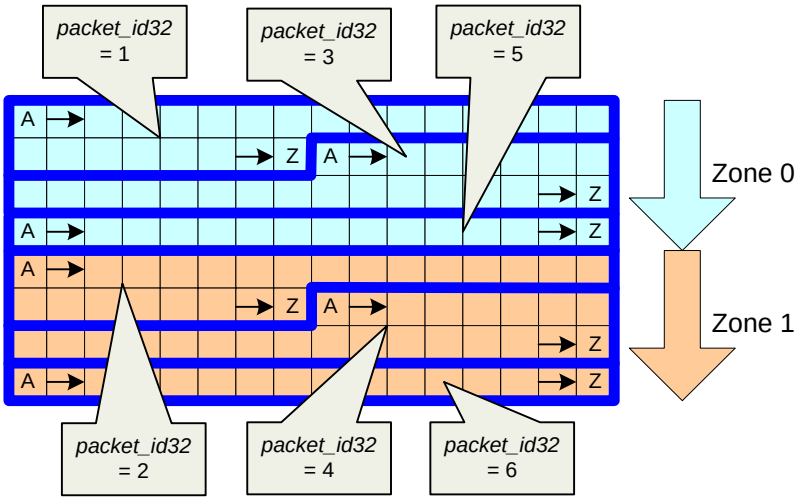


Figure 25-54 : 2 Top-down Zones with no ROI

packet_id32	*zone_id*	*address_offset*	*eoz*	packet size for payload
1	0	0	0	24
2	1	64	0	24
3	0	24	0	24
4	1	88	0	24
5	0	48	1	16
6	1	112	1	16

The previous table shows that the packet size might be smaller than its nominal value when *end_of_zone (eoz)* is set to 1.

Example 2: Top-down and bottom-up zones with no ROI

The second example illustrates a slightly more complex case with 2 vertical taps. The first tap starts is top-down while the second is bottom-up. It can be seen that the last packet of each zone will meet in the middle of the image.

Nominal image size	16 columns x 8 lines
Pixel size	1 byte
Number of zones	2
Packet size (payload)	24 bytes
Zone 0	top-down, 16 x 4
Zone 1	bottom-up, 16 x 4

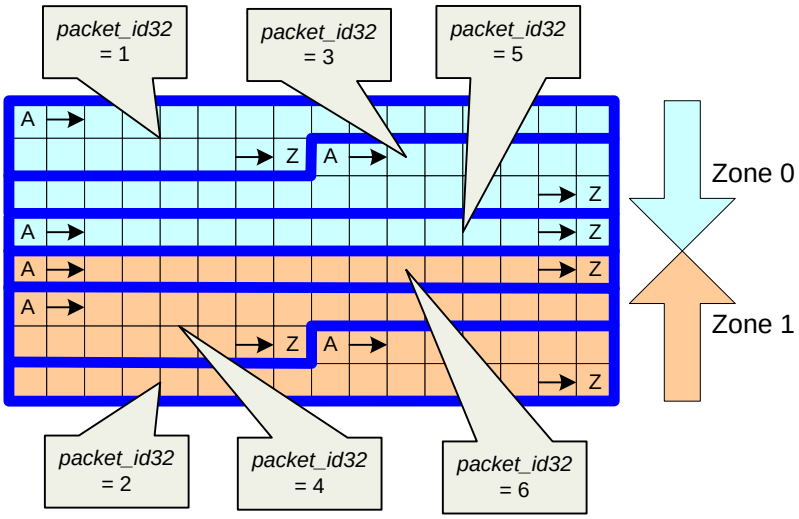


Figure 25-55 : Top-down and Bottom-up Zones with no ROI

packet_id32	*zone_id*	*address_offset*	*eoz*	packet size for payload
1	0	0	0	24
2	1	104	0	24
3	0	24	0	24
4	1	80	0	24
5	0	48	1	16
6	1	64	1	16

Example 3: Top-down and bottom-up with ROI

This last example demonstrates a complex situation where a region of interest is defined. This ROI is not symmetrical with the vertical tap. Hence the first zone has fewer lines than the second. The sequence of packet transmission thus does not follow a round-robin scheme, as the camera would transmit data packet as soon as they are available for a given zone.

Nominal ROI size	14 columns x 7 lines
Pixel size	2 byte
Number of zones	2
Packet size (payload)	48 bytes
Zone 0	top-down, 14 x 2
Zone 1	bottom-up, 14 x 5

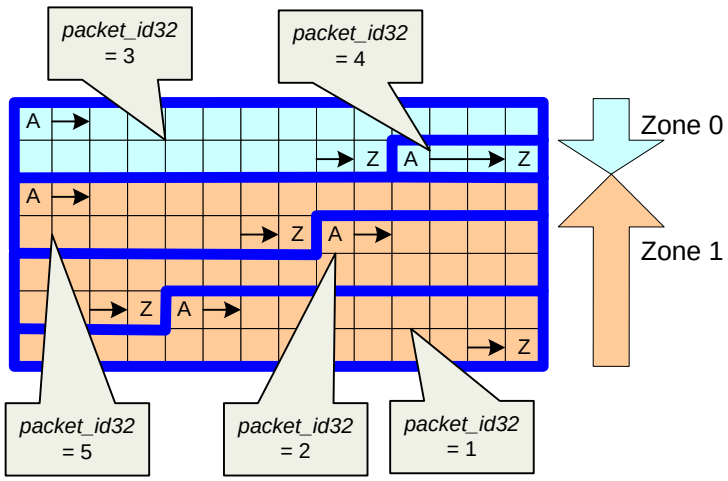


Figure 25-56 : Top-down and Bottom-up with ROI

packet_id32	*zone_id*	*address_offset*	*eoz*	packet size for payload
1	1	148	0	48
2	1	100	0	48
3	0	0	0	48
4	0	48	1	8
5	1	56	1	44

This example illustrates that the data packets can arrive from different zones in any order (not necessarily round-robin), as long as they are sequential within a given zone (top-down or bottom-up). The *packet_id32* is always incrementing to enable the quick detection of a missing packet.

25.11 Device-specific Payload Type

This payload type can be used to transfer device-specific information.

25.11.1 Device-specific Data Leader Packet

[CR-339s] If Device-specific payload is supported, its Data Leader packet MUST follow the layout of Figure 25-57. [CR24-9s]

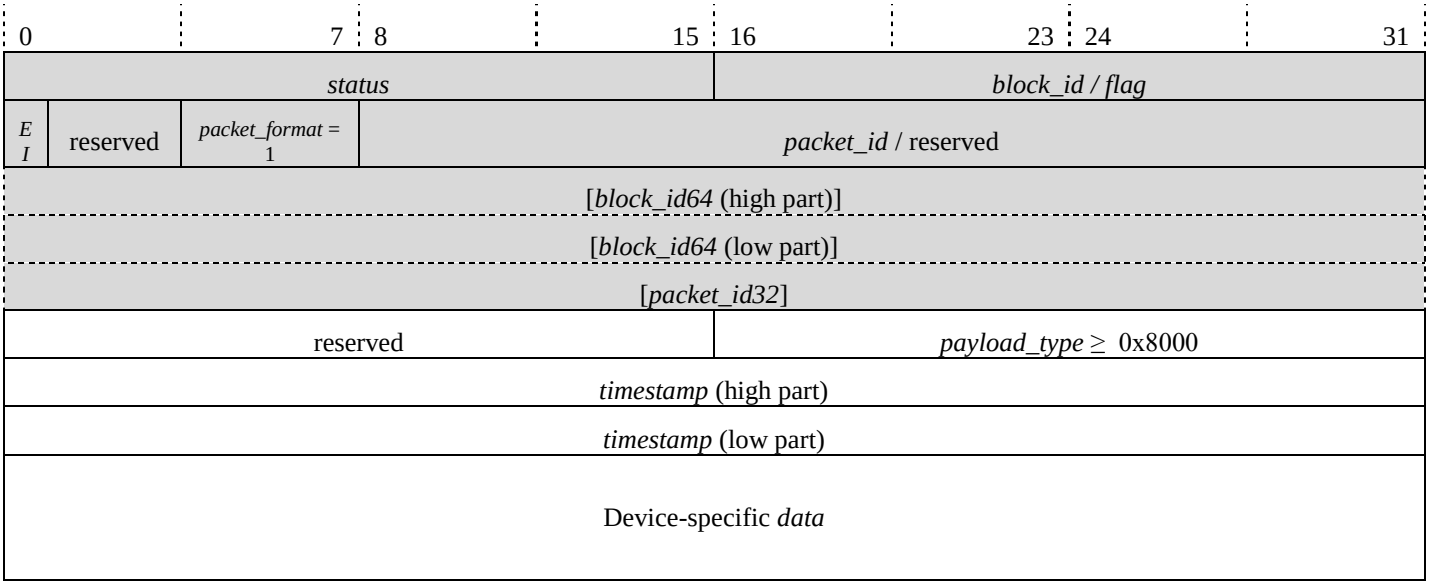


Figure 25-57: Device-specific Data Leader Packet

DEVICE-SPECIFIC DATA LEADER PACKET		
reserved	16 bits	Set to 0 on transmission, ignore on reception.
payload_type	16 bits	Must be ≥ 0x8000.
timestamp	64 bits	See Image payload type.
data	up to packet size	Manufacturer-specific. Not defined by this standard.

25.11.2 Device-specific Data Payload Packet

[CR-340s] If Device-specific payload is supported, its Data Payload packet MUST follow the layout of Figure 25-58. [CR24-17s]

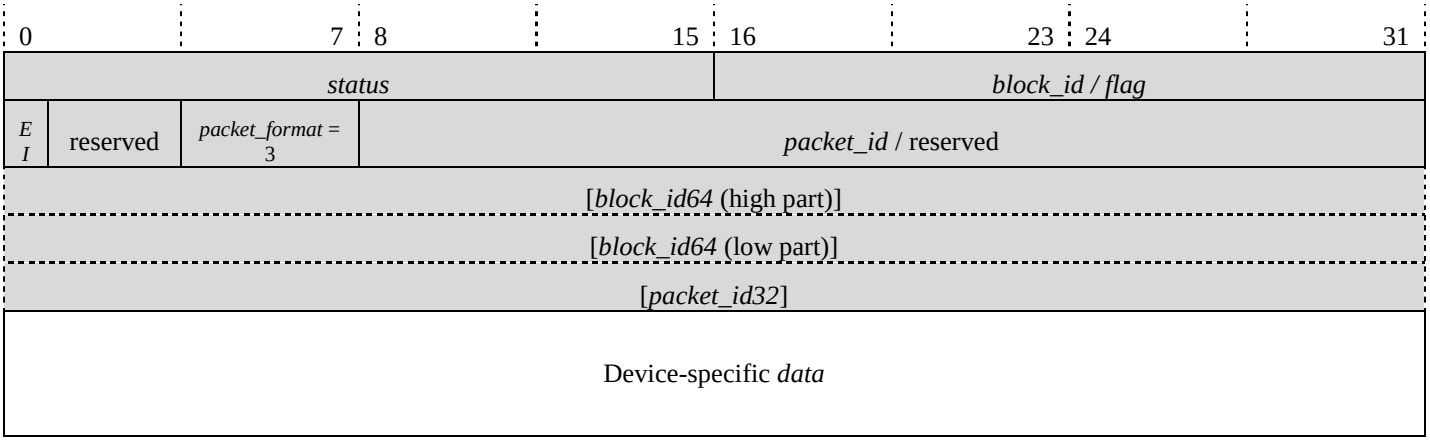


Figure 25-58: Device-specific Data Payload Packet

DATA PAYLOAD PACKET		
data	up to packet size	Device-specific data. For Data Payload packets, the IP Header + UDP Header + GVSP Header + data must be equal to the packet size specified in the Stream Channel Packet Size register of the GVSP transmitter. The only exception is the last Data Payload packet which may be smaller than the specified packet size. However, it is also possible to pad the last data packet so that all data payload packets are exactly the same size.

25.11.3 Device-specific Data Trailer Packet

[CR-341s] If Device-specific payload is supported, its Data Trailer packet MUST follow the layout of Figure 25-59. [CR24-26s]

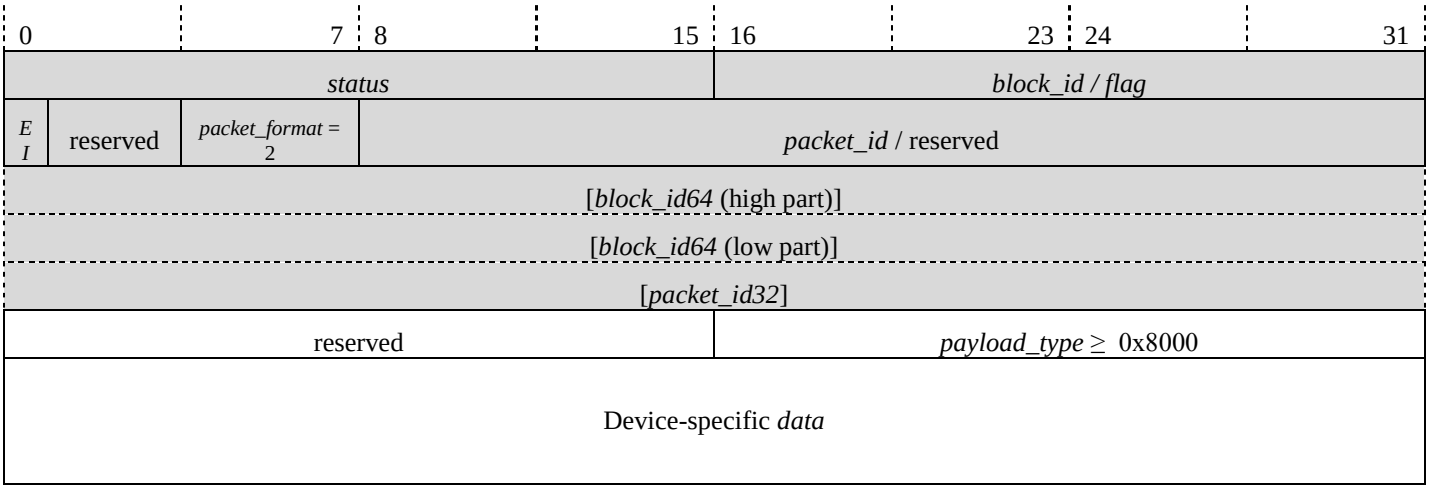


Figure 25-59: Device-specific Data Trailer Packet

DATA PAYLOAD PACKET		
reserved	16 bits	Set to 0 on transmission, ignore on reception.
payload_type	16 bits	Refer to Table 25-1.
data	up to packet size	Manufacturer-specific. Not defined by this standard.

25.11.4 Device-specific All-in Packet

[CR-342s] When All-in transmission mode is supported, if Device-specific payload is supported, then its All-in packet MUST regroup the Device-specific Data Leader, Device-specific Data Trailer and Device-specific Data Payload according to Figure 24-7.

26 Pixel Layouts

This section describes the various pixel and image layouts supported by GigE Vision. As of GigE Vision version 2.0, these layouts are based on the AIA Pixel Format Naming Convention.

Note: The figures in this section are illustrated using little-endian convention with byte 0 on the right. This is in-line with the Pixel Format Naming Convention. Thus msb (bit 7) is on the left of each byte while the lsb (bit 0) is on the right.

[O-343st] GVSP transmitters providing pixel depths greater than 8 bits SHOULD also provide an 8-bit transfer mode to facilitate display of images by GVSP receivers. This mode is realized by truncating the least significant bits. [O25-1st]

The mechanism to select the pixel depth and format is supplied in the XML device description file.

26.1 Pixel Alignment

[R-344st] In order to minimize pixel processing time on the GVSP receiver side, pixel data using multiple-byte in the payload data packets MUST be little-endian by default. [R25-2st]

A GVSP transmitter may implement a converter to translate pixels from little-endian to big-endian. This converter can be activated using the SCPSx bootstrap register (*pixel_endianess* field) when supported. The *big_and_little_endian_supported* field of SCCx register indicates if this converter is supported.

In the following figures, Byte 0 is sent first on the wire, followed by Byte 1 and so on for little-endian mode.

26.2 Line and Image Boundaries

GigE Vision uses **Image Padding** as defined in the Pixel Format Naming Convention. Image Padding requires that no artificial padding is inserted at the end of a line.

Therefore, for some grouped and packed pixel formats, it is possible that pixels from different lines are combined in the same bytes. As an example, imagine an image with an odd width (ex: Width = 641 pixels). If pixels are combined in pair (due to packing), then the last pixel of the first line will be combined with the first pixel of the second line, as illustrated below.

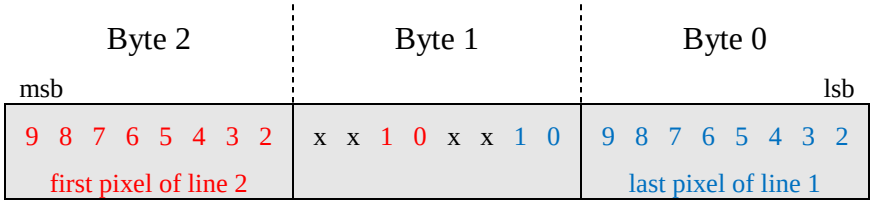


Figure 26-1 : Mono10Packed at an Odd Line Boundary

Another example is when Mono1p is used and the image width is not a multiple of 8 pixels. It takes 8 pixels to fill one byte using this pixel format, so the last pixel of a line might not align with a byte boundary.

Note: To avoid this grouping of pixels from different line, a camera might elect to use a coarser increment for the Width standard feature (see section 29.6).

26.3 Pixel Formats

This section illustrates the various pixel formats natively supported by GVSP. These pixel formats are based on the Pixel Format Naming Convention document, where each pixel format is defined by 5 characteristics:

1. Components and Location
2. # bits
3. Sign indicator (optional)
4. Packing Style (optional)
5. Interface-specific (optional)

[O-345s] A GVSP transmitter or a GVSP receiver SHOULD support a subset of the pixel formats listed in this section. [O25-3s]

Custom pixel formats can be realized by following the rules listed in the Pixel Format Naming Convention for 32-bit pixel format ID values. This is achieved by setting the most significant bit of the *pixel_format* value to 1 and using the remaining 31 bits as manufacturer-specific. Custom pixel formats should only be used when no standard pixel format exists as they prevent interoperability.

26.3.1 Mono1p

This pixel format defines a 1-bit monochrome, unsigned, 8 pixels packed in one byte format. Refer to the Pixel Format Naming Convention for additional information. Last data byte of the total image payload must be padded to the nearest 8-bit boundary. Hence, no padding is put at the end of each line of the image if image width is not a multiple of 8 pixels.

[CR-346s] If supported, “Mono1p” pixel format MUST respect the Pixel Format Naming Convention and MUST follow the layout below.

	Description	Value
Color/Mono	GVSP_PIX_MONO	0x01000000
Bits per pixel	1 bit	0x00010000
Pixel ID	Id of pixel	0x00000037
Pixel Format	Mono1p	0x01010037
Legacy Name	non-existent for GigE Vision 1.x	N/A

Byte 0

msb lsb
Y_{18} .. Y_{11}
Y_{1-16} .. Y_{19}
…
Y_{28} .. Y_{21}
…

26.3.2 Mono2p

This pixel format defines a 2-bit monochrome, unsigned, 4 pixels packed in one byte format. Refer to the Pixel Format Naming Convention for additional information. Last data byte of the total image payload must be padded to the nearest 8 bits. Hence, no padding is put at the end of each line of the image if image width is not a multiple of 4 pixels.

[CR-347s] If supported, “Mono2p” pixel format MUST respect the Pixel Format Naming Convention and MUST follow the layout below.

	Description	Value
Color/Mono	GVSP_PIX_MONO	0x01000000
Bits per pixel	2 bits	0x00020000
Pixel ID	Id of pixel	0x00000038
Pixel Format	Mono2p	0x01020038
Legacy Name	non-existent for GigE Vision 1.x	N/A

Byte 0

msb lsb
Y_{14} .. Y_{11}
Y_{18} .. Y_{15}
…
Y_{24} .. Y_{21}
…

26.3.3 Mono4p

This pixel format defines a 4-bit monochrome, unsigned, 2 pixels packed in one byte format. Refer to the Pixel Format Naming Convention for additional information. Last data byte of the total image payload must be padded to the nearest 8 bits. Hence, no padding is put at the end of each line of the image if image width is not a multiple of 2 pixels.

[CR-348s] If supported, "Mono4p" pixel format MUST respect the Pixel Format Naming Convention and MUST follow the layout below.

	Description	Value
Color/Mono	GVSP_PIX_MONO	0x01000000
Bits per pixel	4 bits	0x00040000
Pixel ID	Id of pixel	0x00000039
Pixel Format	Mono4p	0x01040039
Legacy Name	non-existent for GigE Vision 1.x	N/A

Byte 0

msb	lsb
Y_{12} .. Y_{11}	
Y_{14} .. Y_{13}	
…	
Y_{22} .. Y_{21}	
…	

26.3.4 Mono8

This pixel format defines an 8-bit monochrome, unsigned, unpacked format. Refer to the Pixel Format Naming Convention for additional information.

[CR-349s] If supported, "Mono8" pixel format MUST respect the Pixel Format Naming Convention and follow the MUST layout below. [CR25-4s]

	Description	Value
Color/Mono	GVSP_PIX_MONO	0x01000000
Bits per pixel	8 bits	0x00080000
Pixel ID	Id of pixel	0x00000001
Pixel Format	Mono8	0x01080001
Legacy Name	Mono8 in GigE Vision 1.x	no change

Byte 0

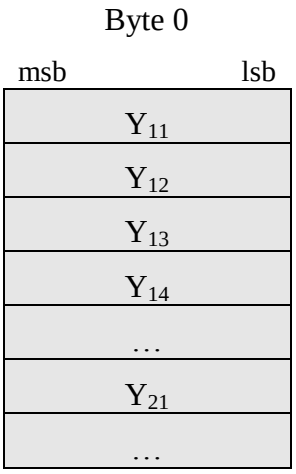
msb lsb
Y_{11}
Y_{12}
Y_{13}
Y_{14}
…
Y_{21}
…

26.3.5 Mono8s

This pixel format defines an 8-bit monochrome, signed, unpacked format. Refer to the Pixel Format Naming Convention for additional information.

[CR-350s] If supported, “Mono8s” pixel format MUST respect the Pixel Format Naming Convention and MUST follow the layout below. [CR25-5s]

	Description	Value
Color/Mono	GVSP_PIX_MONO	0x01000000
Bits per pixel	8 bits	0x00080000
Pixel ID	Id of pixel	0x00000002
Pixel Format	Mono8s	0x01080002
Legacy Name	Mono8Signed in GigE Vision 1.x	deprecated

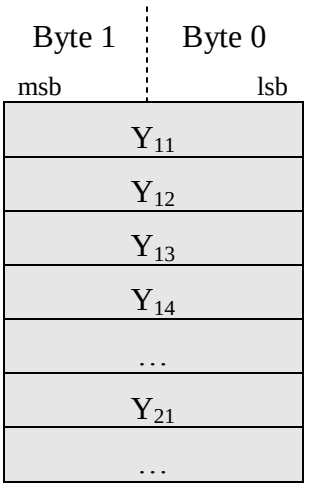


26.3.6 Mono10

This pixel format defines a 10-bit monochrome, unsigned, unpacked format. Refer to the Pixel Format Naming Convention for additional information.

[CR-351s] If supported, “Mono10” pixel format MUST respect the Pixel Format Naming Convention and MUST follow the layout below. [CR25-6s]

	Description	Value
Color/Mono	GVSP_PIX_MONO	0x01000000
Bits per pixel	16 bits	0x00100000
Pixel ID	Id of pixel	0x00000003
Pixel Format:	Mono10	0x01100003
Legacy Name	Mono10 in GigE Vision 1.x	no change

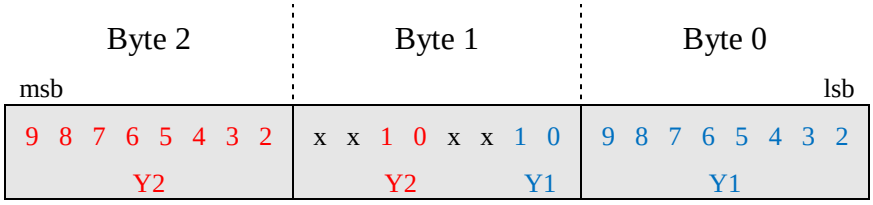


26.3.7 Mono10Packed

This pixel format defines a 10-bit monochrome, unsigned, GigE Vision-specific packed format.

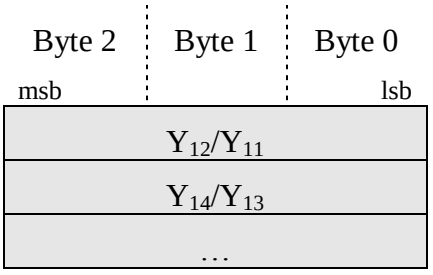
Note: This pixel format **does not** respect the Pixel Format Naming Convention as this format was introduced with GigE Vision 1.0 (prior to creation of the Pixel Format Naming Convention).

With this GigE Vision packing style, two 10-bit pixels are packed into 24 bits as illustrated below (little-endian layout). Main difference with the grouped style of Pixel Format Naming Convention is the ordering of the bytes (the grouped byte is the last one for the Pixel Format Naming Convention).



[CR-352s] If supported, "Mono10Packed" pixel format MUST follow the layout below. [CR25-7s]

	Description	Value
Color/Mono	GVSP_PIX_MONO	0x01000000
Bits per pixel	12 bits	0x000C0000
Pixel ID	Id of pixel	0x00000004
Pixel Format	Mono10Packed	0x010C0004



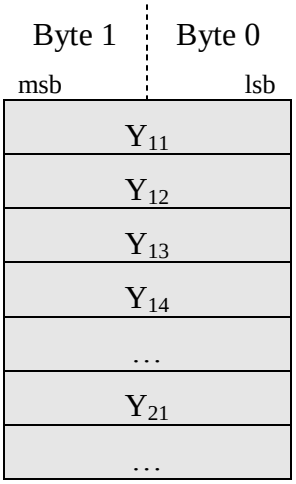
Note that the first pixel of the second line will not start aligned on a 3-byte boundary when the image width contains an odd number of pixels.

26.3.8 Mono12

This pixel format defines a 12-bit monochrome, unsigned, unpacked format. Refer to the Pixel Format Naming Convention for additional information.

[CR-353s] If supported, "Mono12" pixel format MUST respect the Pixel Format Naming Convention and MUST follow the layout below. [CR25-8s]

	Description	Value
Color/Mono	GVSP_PIX_MONO	0x01000000
Bits per pixel	16 bits	0x00100000
Pixel ID	Id of pixel	0x00000005
Pixel Format	Mono12	0x01100005
Legacy Name	Mono12 in GigE Vision 1.x	no change

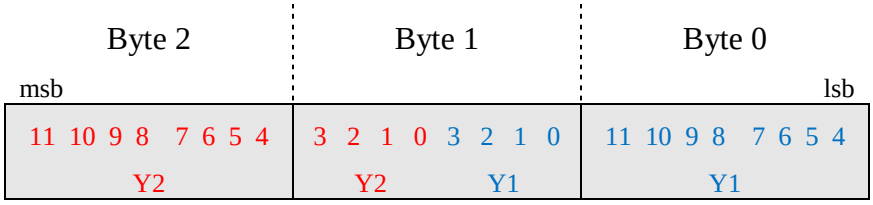


26.3.9 Mono12Packed

This pixel format defines a 12-bit monochrome, unsigned, GigE Vision-specific packed format.

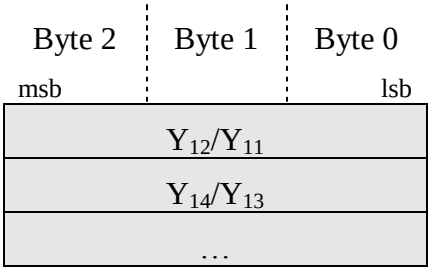
Note: This pixel format **does not** respect the Pixel Format Naming Convention as this format was introduced with GigE Vision 1.0 (prior to creation of the Pixel Format Naming Convention).

With this GigE Vision packing style, two 12-bit pixels are packed into 24 bits as illustrated below (little-endian layout). Main difference with the grouped style of Pixel Format Naming Convention is the ordering of the bytes (the grouped byte is the last one for the Pixel Format Naming Convention).



[CR-354s] If supported, "Mono12Packed" pixel format MUST follow the layout below. [CR25-9s]

	Description	Value
Color/Mono	GVSP_PIX_MONO	0x01000000
Bits per pixel	12 bits	0x000C0000
Pixel ID	Id of pixel	0x00000006
Pixel Format	Mono12Packed	0x010C0006



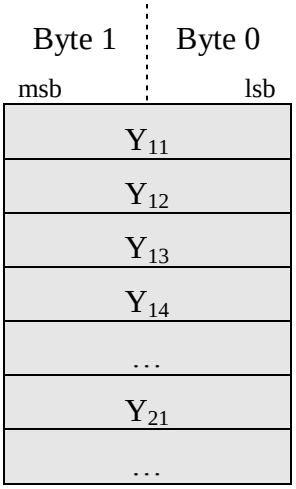
Note that the first pixel of the second line will not start aligned on a 3-byte boundary when the image width contains an odd number of pixels.

26.3.10 Mono14

This pixel format defines a 14-bit monochrome, unsigned, unpacked format. Refer to the Pixel Format Naming Convention for additional information.

[CR-355s] If supported, "Mono14" pixel format MUST respect the Pixel Format Naming Convention and MUST follow the layout below. [CR25-41s]

	Description	Value
Color/Mono	GVSP_PIX_MONO	0x01000000
Bits per pixel	16 bits	0x00100000
Pixel ID	Id of pixel	0x00000025
Pixel Format	Mono14	0x01100025
Legacy Name	Mono14 in GigE Vision 1.x	no change

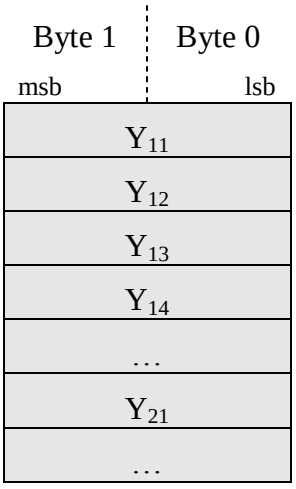


26.3.11 Mono16

This pixel format defines a 16-bit monochrome, unsigned, unpacked format. Refer to the Pixel Format Naming Convention for additional information.

[CR-356s] If supported, “Mono16” pixel format MUST respect the Pixel Format Naming Convention and MUST follow the layout below. [CR25-10s]

	Description	Value
Color/Mono	GVSP_PIX_MONO	0x01000000
Bits per pixel	16 bits	0x00100000
Pixel ID	Id of pixel	0x00000007
Pixel Format	Mono16	0x01100007
Legacy Name	Mono16 in GigE Vision 1.x	no change



26.3.12 BayerGR8

This pixel format defines an 8-bit GRBG (green-red-blue-green) Bayer pattern, unsigned, unpacked format. Refer to the Pixel Format Naming Convention for additional information.

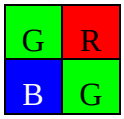


Figure 26-2 : BayerGR 2x2 tile

[CR-357st] All Bayer pixel formats (PixelFormat feature) MUST represent the Bayer pattern of the full image (with no ROI). [CR25-11st]

The *pixel_format* field of the GVSP data leader contains the actual Bayer pixel format of the full image. When a ROI is activated, the receiver can look at the *offset_x* and *offset_y* fields to determine if it needs to adjust the Bayer tile for decoding, relative to its upper left corner.

Note: The GVSP data must be self-described to enable any receiver to decode the image stream. Therefore features which modify the raw image data in a way that a different pixel format needs to be used for decoding must make sure that the *pixel_format* field and the PixelFormat feature are set accordingly.

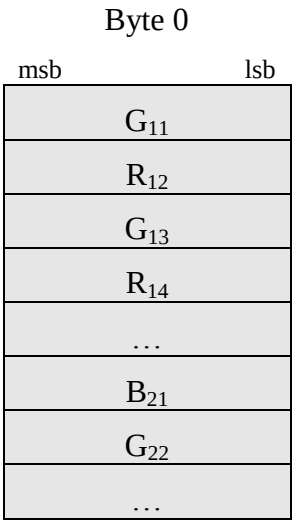
In order for the PixelFormat feature to match the *pixel_format* field of the Image Data Leader, it is recommended to align the offset of any area of interest to the size of the Bayer tile. This is typically implemented by having an increment of 2 pixels on the OffsetX and OffsetY features of the GenICam™ Standard Features Naming Convention.

Additionally, when implementing image flip (ReverseX and ReverseY features of the GenICam Standard Features Naming Convention), the device should internally compensate for the mirror effect taking place on the Bayer tile by shifting image capture by 1 pixel in the direction of the flip.

The above principles also apply to any color filter array (CFA) pattern.

[CR-358s] If supported, “BayerGR8” pixel format MUST respect the Pixel Format Naming Convention and MUST follow the layout below. [CR25-12s]

	Description	Value
Color/Mono	GVSP_PIX_MONO	0x01000000
Bits per pixel	8 Bit	0x00080000
Pixel ID	Id of pixel	0x00000008
Pixel Format	BayerGR8	0x01080008
Legacy Name	BayerGR8 in GigE Vision 1.x	no change



26.3.13 BayerRG8

This pixel format defines an 8-bit RGGB (red-green-green-blue) Bayer pattern, unsigned, unpacked format. Refer to the Pixel Format Naming Convention for additional information.

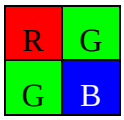


Figure 26-3 : BayerRG 2x2 tile

[CR-359s] If supported, "BayerRG8" pixel format MUST respect the Pixel Format Naming Convention and MUST follow the layout below. [CR25-13s]

	Description	Value
Color/Mono	GVSP_PIX_MONO	0x01000000
Bits per pixel	8 Bit	0x00080000
Pixel ID	Id of pixel	0x00000009
Pixel Format	BayerRG8	0x01080009
Legacy Name	BayerRG8 in GigE Vision 1.x	no change

Byte 0

msb	lsb
R_{11}	
G_{12}	
R_{13}	
G_{14}	
…	
G_{21}	
B_{22}	
…	

26.3.14 BayerGB8

This pixel format defines an 8-bit GBRG (green-blue-red-green) Bayer pattern, unsigned, unpacked format. Refer to the Pixel Format Naming Convention for additional information.

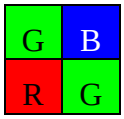


Figure 26-4 : BayerGB 2x2 tile

[CR-360s] If supported, "BayerGB8" pixel format MUST respect the Pixel Format Naming Convention and MUST follow the layout below. [CR25-14s]

	Description	Value
Color/Mono	GVSP_PIX_MONO	0x01000000
Bits per pixel	8 bits	0x00080000
Pixel ID	Id of pixel	0x0000000A
Pixel Format	BayerGB8	0x0108000A
Legacy Name	BayerGB8 in GigE Vision 1.x	no change

Byte 0

msb	lsb
G_{11}	
B_{12}	
G_{13}	
B_{14}	
…	
R_{21}	
G_{22}	
…	

26.3.15 BayerBG8

This pixel format defines an 8-bit BGGR (blue-green-green-red) Bayer pattern, unsigned, unpacked format. Refer to the Pixel Format Naming Convention for additional information.

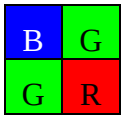
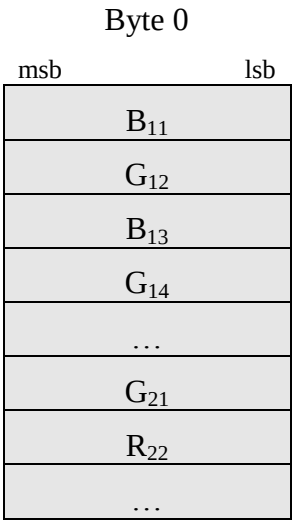


Figure 26-5 : BayerBG 2x2 tile

[CR-361s] If supported, "BayerBG8" pixel format MUST respect the Pixel Format Naming Convention and MUST follow the layout below. [CR25-15s]

	Description	Value
Color/Mono	GVSP_PIX_MONO	0x01000000
Bits per pixel	8 bits	0x00080000
Pixel ID	Id of pixel	0x0000000B
Pixel Format	BayerBG8	0x0108000B
Legacy Name	BayerBG8 in GigE Vision 1.x	no change

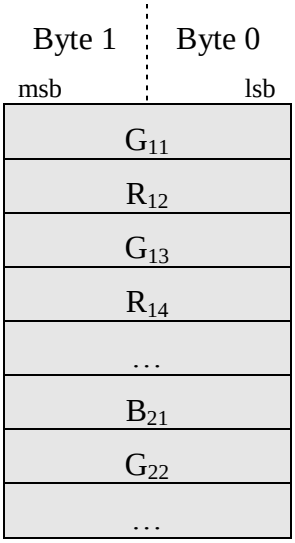


26.3.16 BayerGR10

This pixel format defines a 10-bit GRBG (green-red-blue-green) Bayer pattern, unsigned, unpacked format. Refer to the Pixel Format Naming Convention for additional information.

[CR-362s] If supported, "BayerGR10" pixel format MUST respect the Pixel Format Naming Convention and MUST follow the layout below. [CR25-16s]

	Description	Value
Color/Mono	GVSP_PIX_MONO	0x01000000
Bits per pixel	16 bits	0x00100000
Pixel ID	Id of pixel	0x0000000C
Pixel Format	BayerGR10	0x0110000C
Legacy Name	BayerGR10 in GigE Vision 1.x	no change

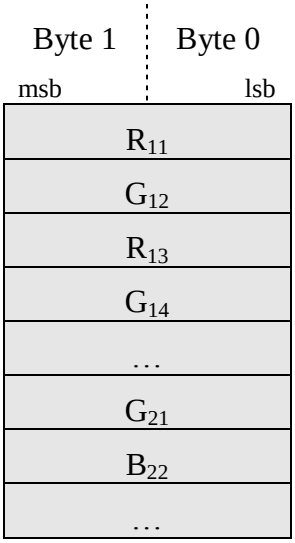


26.3.17 BayerRG10

This pixel format defines a 10-bit RGGB (red-green-green-blue) Bayer pattern, unsigned, unpacked format. Refer to the Pixel Format Naming Convention for additional information.

[CR-363s] If supported, "BayerRG10" pixel format MUST respect the Pixel Format Naming Convention and MUST follow the layout below. [CR25-17s]

	Description	Value
Color/Mono	GVSP_PIX_MONO	0x01000000
Bits per pixel	16 bits	0x00100000
Pixel ID	Id of pixel	0x0000000D
Pixel Format	BayerRG10	0x0110000D
Legacy Name	BayerRG10 in GigE Vision 1.x	no change

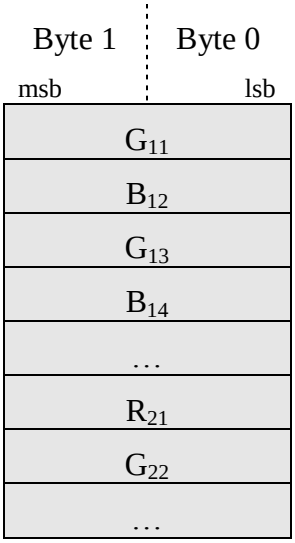


26.3.18 BayerGB10

This pixel format defines a 10-bit GBRG (green-blue-red-green) Bayer pattern, unsigned, unpacked format. Refer to the Pixel Format Naming Convention for additional information.

[CR-364s] If supported, "BayerGB10" pixel format MUST respect the Pixel Format Naming Convention and MUST follow the layout below. [CR25-18s]

	Description	Value
Color/Mono	GVSP_PIX_MONO	0x01000000
Bits per pixel	16 bits	0x00100000
Pixel ID	Id of pixel	0x0000000E
Pixel Format	BayerGB10	0x0110000E
Legacy Name	BayerGB10 in GigE Vision 1.x	no change

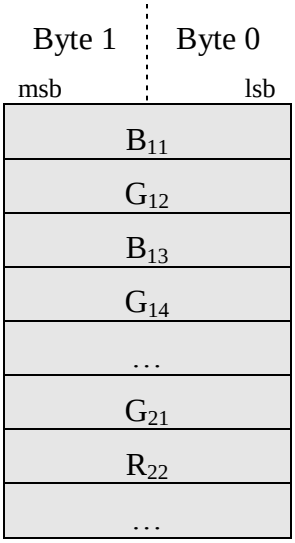


26.3.19 BayerBG10

This pixel format defines a 10-bit BGGR (blue-green-green-red) Bayer pattern, unsigned, unpacked format. Refer to the Pixel Format Naming Convention for additional information.

[CR-365s] If supported, "BayerBG10" pixel format MUST respect the Pixel Format Naming Convention and MUST follow the layout below. [CR25-19s]

	Description	Value
Color/Mono	GVSP_PIX_MONO	0x01000000
Bits per pixel	16 bits	0x00100000
Pixel ID	Id of pixel	0x0000000F
Pixel Format	BayerBG10	0x0110000F
Legacy Name	BayerBG10 in GigE Vision 1.x	no change

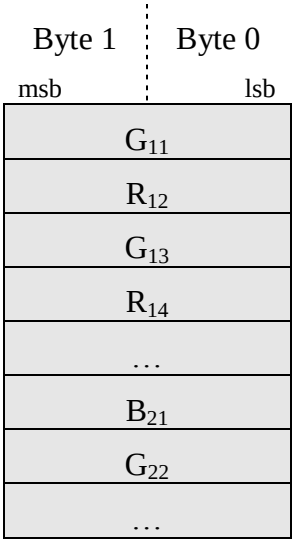


26.3.20 BayerGR12

This pixel format defines a 12-bit GRBG (green-red-blue-green) Bayer pattern, unsigned, unpacked format. Refer to the Pixel Format Naming Convention for additional information.

[CR-366s] If supported, "BayerGR12" pixel format MUST respect the Pixel Format Naming Convention and MUST follow the layout below. [CR25-20s]

	Description	Value
Color/Mono	GVSP_PIX_MONO	0x01000000
Bits per pixel	16 bits	0x00100000
Pixel ID	Id of pixel	0x00000010
Pixel Format	BayerGR12	0x01100010
Legacy Name	BayerGR12 in GigE Vision 1.x	no change

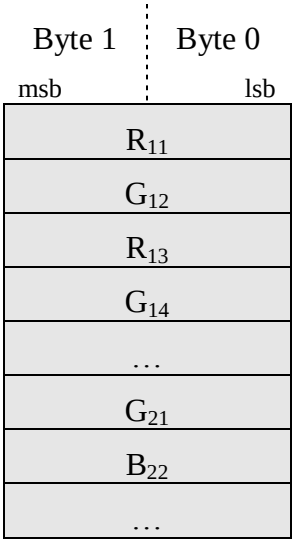


26.3.21 BayerRG12

This pixel format defines a 12-bit RGGB (red-green-green-blue) Bayer pattern, unsigned, unpacked format. Refer to the Pixel Format Naming Convention for additional information.

[CR-367s] If supported, "BayerRG12" pixel format MUST respect the Pixel Format Naming Convention and MUST follow the layout below. [CR25-21s]

	Description	Value
Color/Mono	GVSP_PIX_MONO	0x01000000
Bits per pixel	16 bits	0x00100000
Pixel ID	Id of pixel	0x00000011
Pixel Format	BayerRG12	0x01100011
Legacy Name	BayerRG12 in GigE Vision 1.x	no change

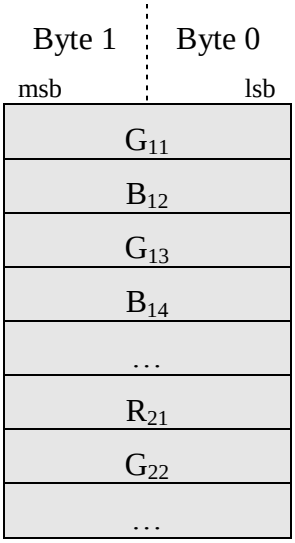


26.3.22 BayerGB12

This pixel format defines a 12-bit GBRG (green-blue-red-green) Bayer pattern, unsigned, unpacked format. Refer to the Pixel Format Naming Convention for additional information.

[CR-368s] If supported, "BayerGB12" pixel format MUST respect the Pixel Format Naming Convention and MUST follow the layout below. [CR25-22s]

	Description	Value
Color/Mono	GVSP_PIX_MONO	0x01000000
Bits per pixel	16 bits	0x00100000
Pixel ID	Id of pixel	0x00000012
Pixel Format	BayerGB12	0x01100012
Legacy Name	BayerGB12 in GigE Vision 1.x	no change

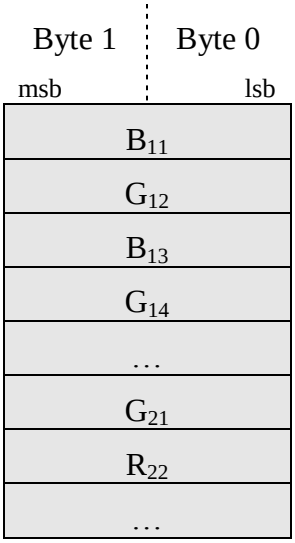


26.3.23 BayerBG12

This pixel format defines a 12-bit BGGR (blue-green-green-red) Bayer pattern, unsigned, unpacked format. Refer to the Pixel Format Naming Convention for additional information.

[CR-369s] If supported, "BayerBG12" pixel format MUST respect the Pixel Format Naming Convention and MUST follow the layout below. [CR25-23s]

	Description	Value
Color/Mono	GVSP_PIX_MONO	0x01000000
Bits per pixel	16 bits	0x00100000
Pixel ID	Id of pixel	0x00000013
Pixel Format	BayerBG12	0x01100013
Legacy Name	BayerBG12 in GigE Vision 1.x	no change

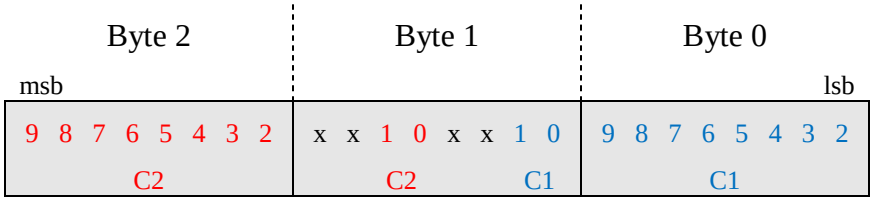


26.3.24 BayerGR10Packed

This pixel format defines a 10-bit GRBG (green-red-blue-green) Bayer pattern, unsigned, GigE Vision-specific packed format.

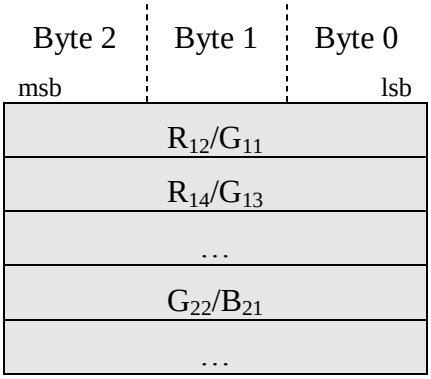
Note: This pixel format **does not** respect the Pixel Format Naming Convention as this format was introduced with GigE Vision 1.0 (prior to creation of the Pixel Format Naming Convention).

With this GigE Vision packing style, two 10-bit color components (C1 and C2) are packed into 24 bits as illustrated below (little-endian layout). Main difference with the grouped style of Pixel Format Naming Convention is the ordering of the bytes (the grouped byte is the last one for the Pixel Format Naming Convention).



[CR-370s] If supported, "BayerGR10Packed" pixel format MUST follow the layout below. [CR25-42s]

	Description	Value
Color/Mono	GVSP_PIX_MONO	0x01000000
Bits per pixel	12 bits	0x000C0000
Pixel ID	Id of pixel	0x00000026
Pixel Format	BayerGR10Packed	0x010C0026



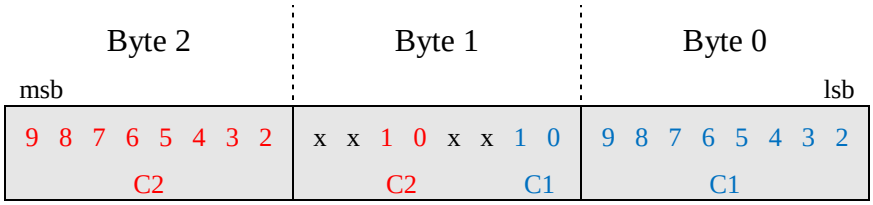
The above is packed in the same way as Mono10Packed.

26.3.25 BayerRG10Packed

This pixel format defines a 10-bit RGGB (red-green-green-blue) Bayer pattern, unsigned, GigE Vision-specific packed format.

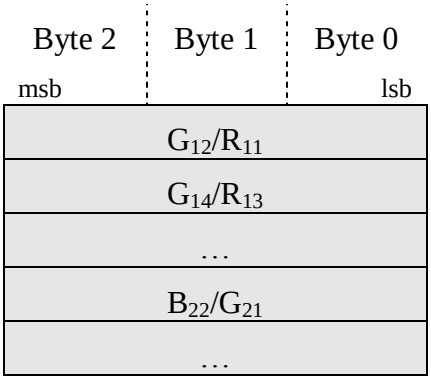
Note: This pixel format **does not** respect the Pixel Format Naming Convention as this format was introduced with GigE Vision 1.0 (prior to creation of the Pixel Format Naming Convention).

With this GigE Vision packing style, two 10-bit color components (C1 and C2) are packed into 24 bits as illustrated below (little-endian layout). Main difference with the grouped style of Pixel Format Naming Convention is the ordering of the bytes (the grouped byte is the last one for the Pixel Format Naming Convention).



[CR-371s] If supported, "BayerRG10Packed" pixel format MUST follow the layout below. [CR25-43s]

	Description	Value
Color/Mono	GVSP_PIX_MONO	0x01000000
Bits per pixel	12 bits	0x000C0000
Pixel ID	Id of pixel	0x00000027
Pixel Format	BayerRG10Packed	0x010C0027



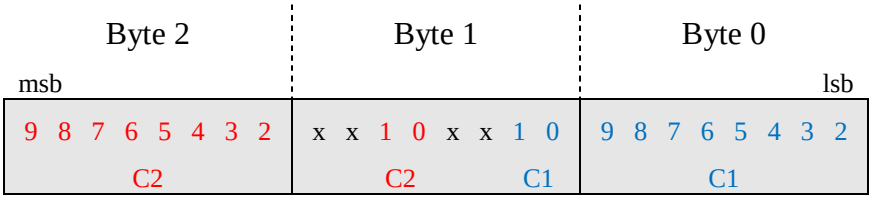
The above is packed in the same way as Mono10Packed.

26.3.26 BayerGB10Packed

This pixel format defines a 10-bit GBRG (green-blue-red-green) Bayer pattern, unsigned, GigE Vision-specific packed format.

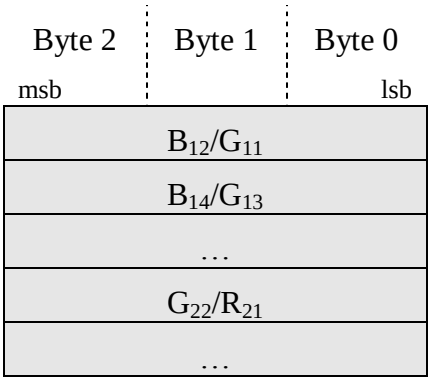
Note: This pixel format **does not** respect the Pixel Format Naming Convention as this format was introduced with GigE Vision 1.0 (prior to creation of the Pixel Format Naming Convention).

With this GigE Vision packing style, two 10-bit color components (C1 and C2) are packed into 24 bits as illustrated below (little-endian layout). Main difference with the grouped style of Pixel Format Naming Convention is the ordering of the bytes (the grouped byte is the last one for the Pixel Format Naming Convention).



[CR-372s] If supported, "BayerGB10Packed" pixel format MUST follow the layout below. [CR25-44s]

	Description	Value
Color/Mono	GVSP_PIX_MONO	0x01000000
Bits per pixel	12 bits	0x000C0000
Pixel ID	Id of pixel	0x00000028
Legacy Name	BayerGB10Packed	0x010C0028



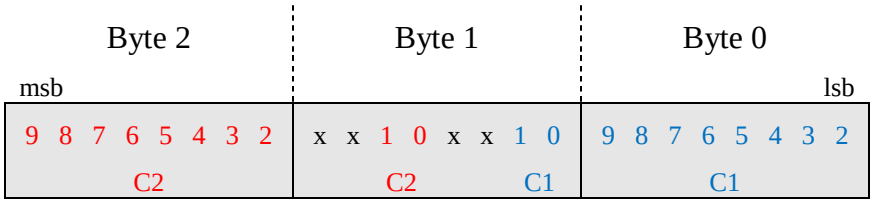
The above is packed in the same way as Mono10Packed.

26.3.27 BayerBG10Packed

This pixel format defines a 10-bit BGGR (blue-green-green-red) Bayer pattern, unsigned, GigE Vision-specific packed format.

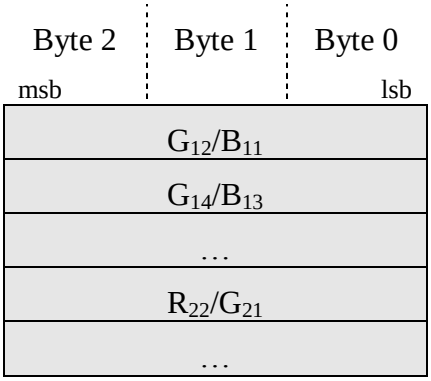
Note: This pixel format **does not** respect the Pixel Format Naming Convention as this format was introduced with GigE Vision 1.0 (prior to creation of the Pixel Format Naming Convention).

With this GigE Vision packing style, two 10-bit color components (C1 and C2) are packed into 24 bits as illustrated below (little-endian layout). Main difference with the grouped style of Pixel Format Naming Convention is the ordering of the bytes (the grouped byte is the last one for the Pixel Format Naming Convention).



[CR-373s] If supported, "BayerBG10Packed" pixel format MUST respect the Pixel Format Naming Convention and MUST follow the layout below. [CR25-45s]

	Description	Value
Color/Mono	GVSP_PIX_MONO	0x01000000
Bits per pixel	12 bits	0x000C0000
Pixel ID	Id of pixel	0x00000029
Pixel Format	BayerBG10Packed	0x010C0029



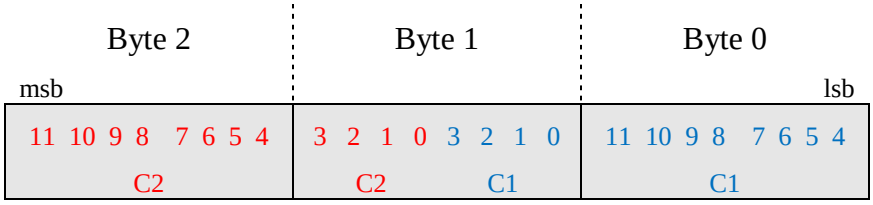
The above is packed in the same way as Mono10Packed.

26.3.28 BayerGR12Packed

This pixel format defines a 12-bit GRBG (green-red-blue-green) Bayer pattern, unsigned, GigE Vision-specific packed format.

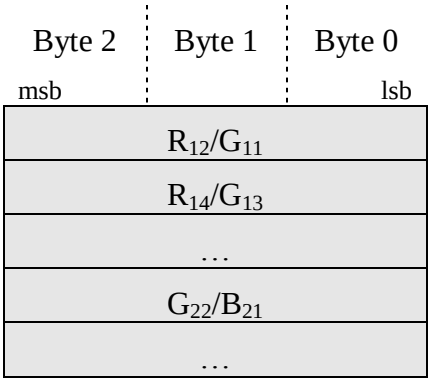
Note: This pixel format **does not** respect the Pixel Format Naming Convention as this format was introduced with GigE Vision 1.0 (prior to creation of the Pixel Format Naming Convention).

With this GigE Vision packing style, two 12-bit color components (C1 and C2) are packed into 24 bits as illustrated below (little-endian layout). Main difference with the grouped style of Pixel Format Naming Convention is the ordering of the bytes (the grouped byte is the last one for the Pixel Format Naming Convention).



[CR-374s] If supported, "BayerGR12Packed" pixel format MUST follow the layout below. [CR25-46s]

	Description	Value
Color/Mono	GVSP_PIX_MONO	0x01000000
Bits per pixel	12 bits	0x000C0000
Pixel ID	Id of pixel	0x0000002A
Pixel Format	BayerGR12Packed	0x010C002A



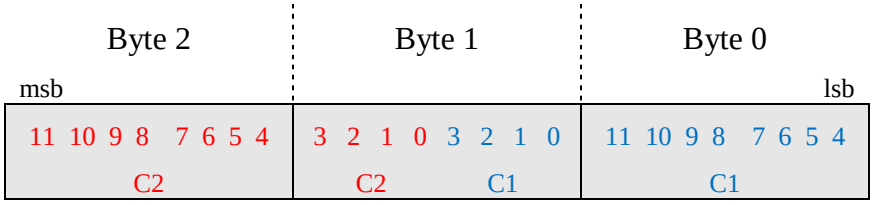
The above is packed in the same way as Mono12Packed.

26.3.29 BayerRG12Packed

This pixel format defines a 12-bit RGGB (red-green-green-blue) Bayer pattern, unsigned, GigE Vision-specific packed format.

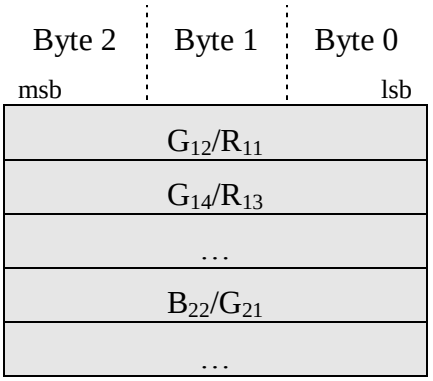
Note: This pixel format **does not** respect the Pixel Format Naming Convention as this format was introduced with GigE Vision 1.0 (prior to creation of the Pixel Format Naming Convention).

With this GigE Vision packing style, two 12-bit color components (C1 and C2) are packed into 24 bits as illustrated below (little-endian layout). Main difference with the grouped style of Pixel Format Naming Convention is the ordering of the bytes (the grouped byte is the last one for the Pixel Format Naming Convention).



[CR-375s] If supported, "BayerRG12Packed" pixel format MUST respect the Pixel Format Naming Convention and MUST follow the layout below. [CR25-47s]

	Description	Value
Color/Mono	GVSP_PIX_MONO	0x01000000
Bits per pixel	12 bits	0x000C0000
Pixel ID	Id of pixel	0x0000002B
Legacy Name	BayerRG12Packed	0x010C002B



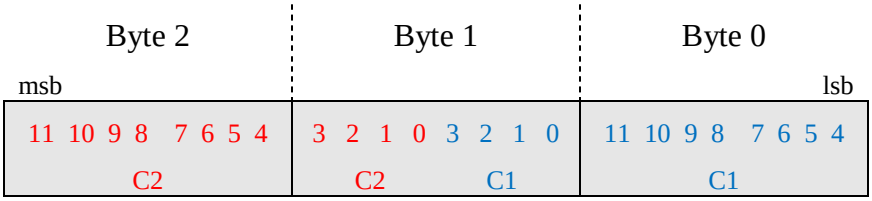
The above is packed in the same way as Mono12Packed.

26.3.30 BayerGB12Packed

This pixel format defines a 12-bit GBRG (green-blue-red-green) Bayer pattern, unsigned, GigE Vision-specific packed format.

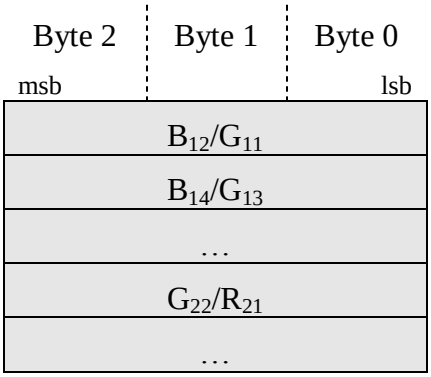
Note: This pixel format **does not** respect the Pixel Format Naming Convention as this format was introduced with GigE Vision 1.0 (prior to creation of the Pixel Format Naming Convention).

With this GigE Vision packing style, two 12-bit color components (C1 and C2) are packed into 24 bits as illustrated below (little-endian layout). Main difference with the grouped style of Pixel Format Naming Convention is the ordering of the bytes (the grouped byte is the last one for the Pixel Format Naming Convention).



[CR-376s] If supported, "BayerGB12Packed" pixel format MUST follow the layout below. [CR25-48s]

	Description	Value
Color/Mono	GVSP_PIX_MONO	0x01000000
Bits per pixel	12 bits	0x000C0000
Pixel ID	Id of pixel	0x0000002C
Pixel Format	BayerGB12Packed	0x010C002C



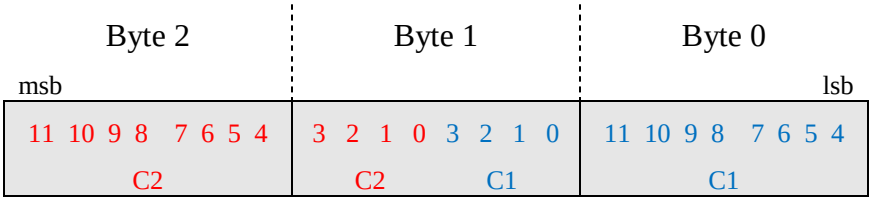
The above is packed in the same way as Mono12Packed.

26.3.31 BayerBG12Packed

This pixel format defines a 12-bit BGGR (blue-green-green-red) Bayer pattern, unsigned, GigE Vision-specific packed format.

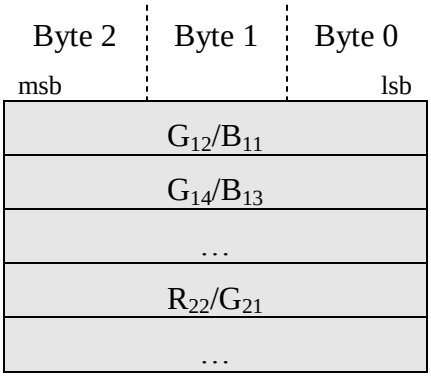
Note: This pixel format **does not** respect the Pixel Format Naming Convention as this format was introduced with GigE Vision 1.0 (prior to creation of the Pixel Format Naming Convention).

With this GigE Vision packing style, two 12-bit color components (C1 and C2) are packed into 24 bits as illustrated below (little-endian layout). Main difference with the grouped style of Pixel Format Naming Convention is the ordering of the bytes (the grouped byte is the last one for the Pixel Format Naming Convention).



[CR-377s] If supported, "BayerBG12Packed" pixel format MUST follow the layout below. [CR25-49s]

	Description	Value
Color/Mono	GVSP_PIX_MONO	0x01000000
Bits per pixel	12 bits	0x000C0000
Pixel ID	Id of pixel	0x0000002D
Pixel Format	BayerBG12Packed	0x010C002D



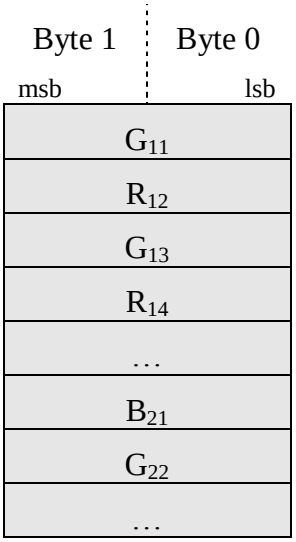
The above is packed in the same way as Mono12Packed.

26.3.32 BayerGR16

This pixel format defines a 16-bit GRBG (green-red-blue-green) Bayer pattern, unsigned, unpacked format. Refer to the Pixel Format Naming Convention for additional information.

[CR-378s] If supported, "BayerGR16" pixel format MUST respect the Pixel Format Naming Convention and MUST follow the layout below. [CR25-50s]

	Description	Value
Color/Mono	GVSP_PIX_MONO	0x01000000
Bits per pixel	16 bits	0x00100000
Pixel ID	Id of pixel	0x0000002E
Pixel Format	BayerGR16	0x0110002E
Legacy Name	BayerGR16 in GigE Vision 1.x	no change

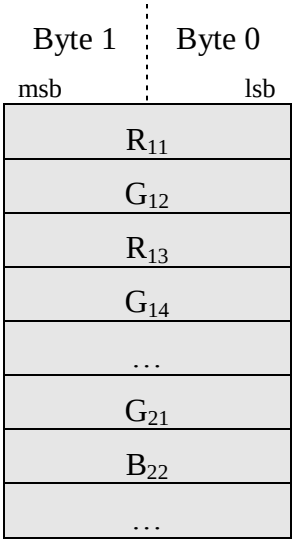


26.3.33 BayerRG16

This pixel format defines a 16-bit RGGB (red-green-green-blue) Bayer pattern, unsigned, unpacked format. Refer to the Pixel Format Naming Convention for additional information.

[CR-379s] If supported, "BayerRG16" pixel format MUST respect the Pixel Format Naming Convention and MUST follow the layout below. [CR25-51s]

	Description	Value
Color/Mono	GVSP_PIX_MONO	0x01000000
Bits per pixel	16 bits	0x00100000
Pixel ID	Id of pixel	0x0000002F
Pixel Format	BayerRG16	0x0110002F
Legacy Name	BayerRG16 in GigE Vision 1.x	no change

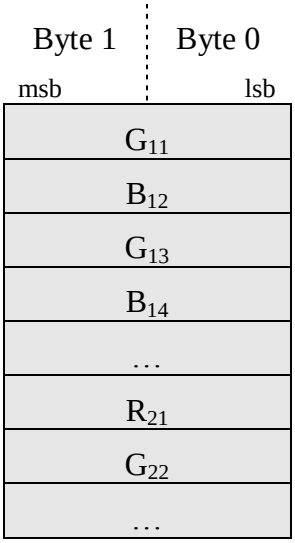


26.3.34 BayerGB16

This pixel format defines a 16-bit GBRG (green-blue-red-green) Bayer pattern, unsigned, unpacked format. Refer to the Pixel Format Naming Convention for additional information.

[CR-380s] If supported, "BayerGB16" pixel format MUST respect the Pixel Format Naming Convention and MUST follow the layout below. [CR25-52s]

	Description	Value
Color/Mono	GVSP_PIX_MONO	0x01000000
Bits per pixel	16 bits	0x00100000
Pixel ID	Id of pixel	0x00000030
Pixel Format	BayerGB16	0x01100030
Legacy Name	BayerGB16 in GigE Vision 1.x	no change

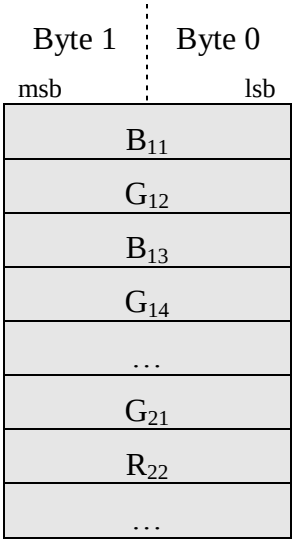


26.3.35 BayerBG16

This pixel format defines a 16-bit BGGR (blue-green-green-red) Bayer pattern, unsigned, unpacked format. Refer to the Pixel Format Naming Convention for additional information.

[CR-381s] If supported, "BayerBG16" pixel format MUST respect the Pixel Format Naming Convention and MUST follow the layout below. [CR25-53s]

	Description	Value
Color/Mono	GVSP_PIX_MONO	0x01000000
Bits per pixel	16 bits	0x00100000
Pixel ID	Id of pixel	0x00000031
Pixel Format	BayerBG16	0x01100031
Legacy Name	BayerBG16 in GigE Vision 1.x	no change

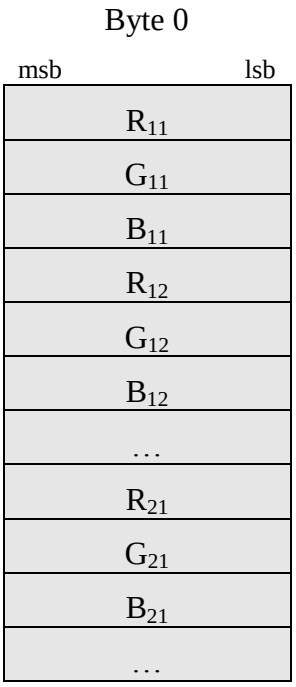


26.3.36 RGB8

This pixel format defines an 8-bit RGB (red-green-blue), unsigned, unpacked format. Refer to the Pixel Format Naming Convention for additional information.

[CR-382s] If supported, "RGB8" pixel format MUST respect the Pixel Format Naming Convention and MUST follow the layout below. [CR25-24s]

	Description	Value
Color/Mono	GVSP_PIX_COLOR	0x02000000
Bits per pixel	24 bits	0x00180000
Pixel ID	Id of pixel	0x00000014
Pixel Format	RGB8	0x02180014
Legacy Name	RGB8Packed in GigE Vision 1.x	deprecated

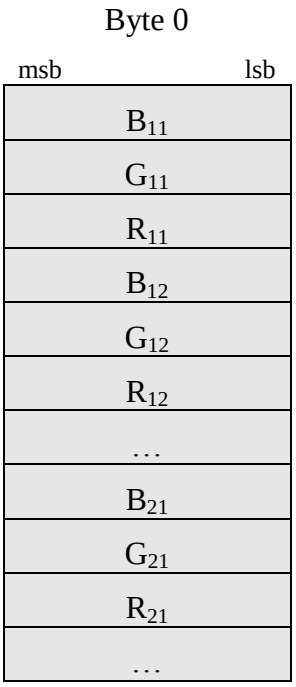


26.3.37 BGR8

This pixel format defines an 8-bit BGR (blue-green-red), unsigned, unpacked format. Refer to the Pixel Format Naming Convention for additional information.

[CR-383s] If supported, "BGR8" pixel format MUST respect the Pixel Format Naming Convention and MUST follow the layout below. [CR25-25s]

	Description	Value
Color/Mono	GVSP_PIX_COLOR	0x02000000
Bits per pixel	24 bits	0x00180000
Pixel ID	Id of pixel	0x00000015
Pixel Format	BGR8	0x02180015
Legacy Name	BGR8Packed in GigE Vision 1.x	deprecated



26.3.38 RGBa8

This pixel format defines an 8-bit RGBa (red-green-blue-alpha), unsigned, unpacked format. Refer to the Pixel Format Naming Convention for additional information. The content of the alpha component is manufacturer-specific.

[CR-384s] If supported, "RGBa8" pixel format MUST respect the Pixel Format Naming Convention and MUST follow the layout below. [CR25-26s]

	Description	Value
Color/Mono	GVSP_PIX_COLOR	0x02000000
Bits per pixel	32 bits	0x00200000
Pixel ID	Id of pixel	0x00000016
Pixel Format	RGBa8	0x02200016
Legacy Name	RGBA8Packed in GigE Vision 1.x	deprecated

Byte 0

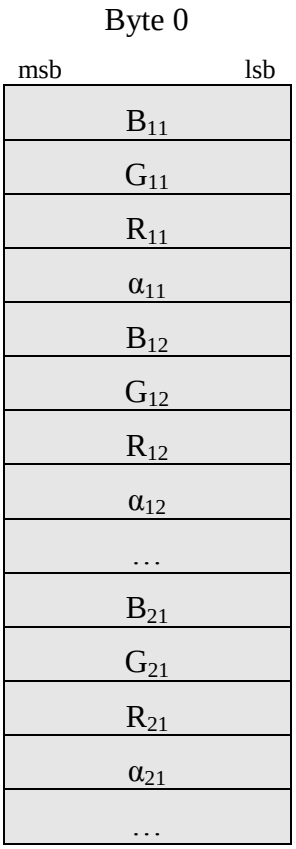
msb lsb
R_{11}
G_{11}
B_{11}
α_{11}
R_{12}
G_{12}
B_{12}
α_{12}
…
R_{21}
G_{21}
B_{21}
α_{21}
…

26.3.39 BGRa8

This pixel format defines an 8-bit BGRa (blue-green-red-alpha), unsigned, unpacked format. Refer to the Pixel Format Naming Convention for additional information. The content of the alpha component is manufacturer-specific.

[CR-385s] If supported, “BGRa8” pixel format MUST respect the Pixel Format Naming Convention and MUST follow the layout below. [CR25-27s]

	Description	Value
Color/Mono	GVSP_PIX_COLOR	0x02000000
Bits per pixel	32 bits	0x00200000
Pixel ID	Id of pixel	0x00000017
Pixel Format	BGRa8	0x02200017
Legacy Name	BGRA8Packed in GigE Vision 1.x	deprecated

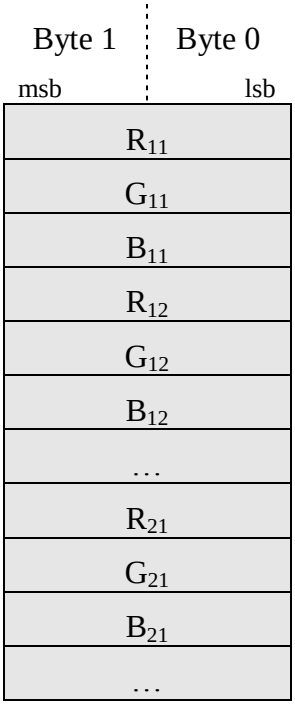


26.3.40 RGB10

This pixel format defines a 10-bit RGB (red-green-blue), unsigned, unpacked format. Refer to the Pixel Format Naming Convention for additional information.

[CR-386s] If supported, "RGB10" pixel format MUST respect the Pixel Format Naming Convention and MUST follow the layout below. [CR25-28s]

	Description	Value
Color/Mono	GVSP_PIX_COLOR	0x02000000
Bits per pixel	48 bits	0x00300000
Pixel ID	Id of pixel	0x00000018
Pixel Format	RGB10	0x02300018
Legacy Name	RGB10Packed in GigE Vision 1.x	deprecated

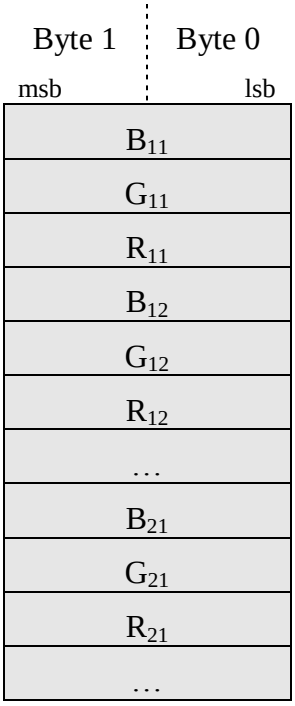


26.3.41 BGR10

This pixel format defines a 10-bit BGR (blue-green-red), unsigned, unpacked format. Refer to the Pixel Format Naming Convention for additional information.

[CR-387s] If supported, "BGR10" pixel format MUST respect the Pixel Format Naming Convention and MUST follow the layout below. [CR25-29s]

	Description	Value
Color/Mono	GVSP_PIX_COLOR	0x02000000
Bits per pixel	48 bits	0x00300000
Pixel ID	Id of pixel	0x00000019
Pixel Format	BGR10	0x02300019
Legacy Name	BGR10Packed in GigE Vision 1.x	deprecated

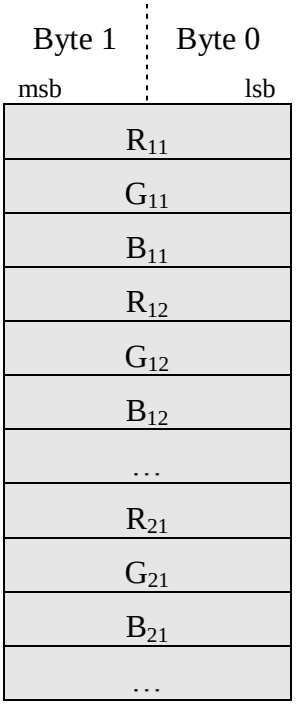


26.3.42 RGB12

This pixel format defines a 12-bit RGB (red-green-blue), unsigned, unpacked format. Refer to the Pixel Format Naming Convention for additional information.

[CR-388s] If supported, "RGB12" pixel format MUST respect the Pixel Format Naming Convention and MUST follow the layout below. [CR25-30s]

	Description	Value
Color/Mono	GVSP_PIX_COLOR	0x02000000
Bits per pixel	48 bits	0x00300000
Pixel ID	Id of pixel	0x0000001A
Pixel Format	RGB12	0x0230001A
Legacy Name	RGB12Packed in GigE Vision 1.x	deprecated

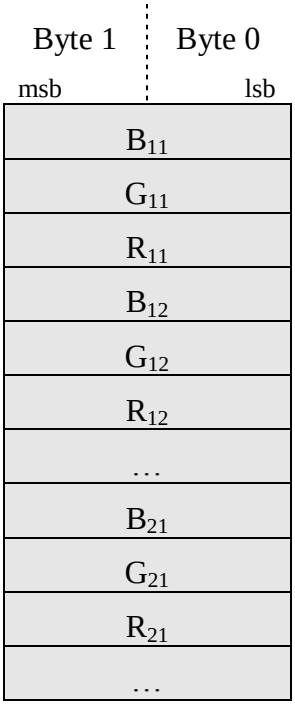


26.3.43 BGR12

This pixel format defines a 12-bit BGR (blue-green-red), unsigned, unpacked format. Refer to the Pixel Format Naming Convention for additional information.

[CR-389s] If supported, "BGR12" pixel format MUST respect the Pixel Format Naming Convention and MUST follow the layout below. [CR25-31s]

	Description	Value
Color/Mono	GVSP_PIX_COLOR	0x02000000
Bits per pixel	48 bits	0x00300000
Pixel ID	Id of pixel	0x0000001B
Pixel Format	BGR12	0x0230001B
Legacy Name	BGR12Packed in GigE Vision 1.x	deprecated

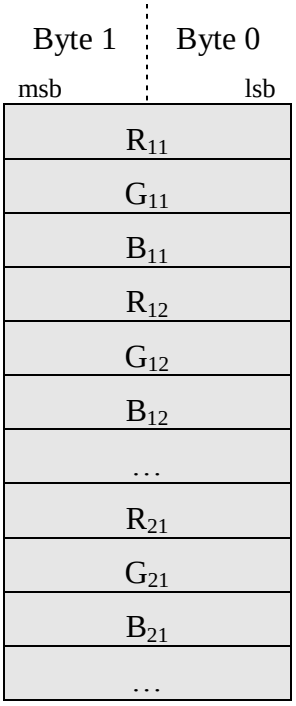


26.3.44 RGB16

This pixel format defines a 16-bit RGB (red-green-blue), unsigned, unpacked format. Refer to the Pixel Format Naming Convention for additional information.

[CR-390s] If supported, "RGB16" pixel format MUST respect the Pixel Format Naming Convention and MUST follow the layout below. [CR25-54s]

	Description	Value
Color/Mono	GVSP_PIX_COLOR	0x02000000
Bits per pixel	48 bits	0x00300000
Pixel ID	Id of pixel	0x00000033
Pixel Format	RGB16	0x02300033
Legacy Name	RGB16Packed in GigE Vision 1.x	deprecated

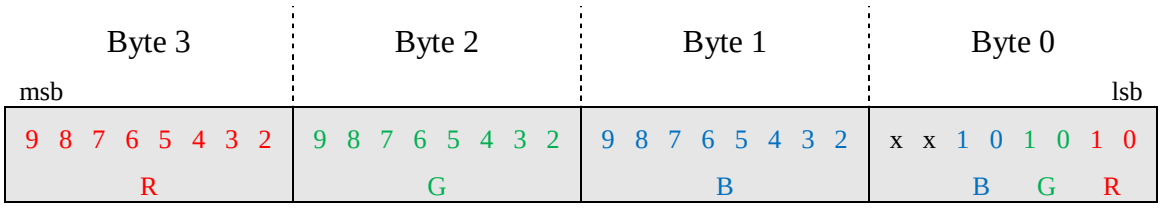


26.3.45 RGB10V1Packed

This pixel format defines a 10-bit RGB (red-green-blue), unsigned, GigE Vision-specific packed format.

Note: This pixel format **does not** respect the Pixel Format Naming Convention as this format was introduced with GigE Vision 1.0 (prior to creation of the Pixel Format Naming Convention).

With this GigE Vision packing style, three 10-bit color components are combined into 32 bits as illustrated below (little-endian layout). Main difference with the grouped style of Pixel Format Naming Convention is the ordering of the bytes.



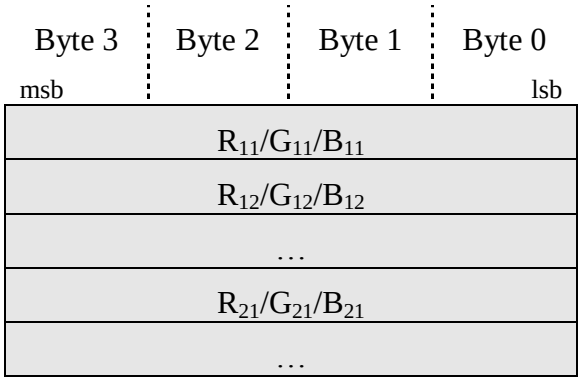
Errata: GigE Vision 1.2 introduced an error in this pixel format by swapping the red and blue component order without changing the pixel value (0x0220001C). This error is seen in [CR25-32s] of GigE Vision 1.2 text under the pixel alignment entry. It should have read “R is position A” and “B is position C”.

GigE Vision 2.0 uses the original definition supported by revision 1.0 and 1.1.

GigE Vision 1.2 definition must be avoided whenever possible as it created an incompatibility.

[CR-391s] If supported, "RGB10V1Packed" pixel format MUST follow the layout below. [CR25-32s]

	Description	Value
Color/Mono	GVSP_PIX_COLOR	0x02000000
Bits per pixel	32 bits	0x00200000
Pixel ID	Id of pixel	0x0000001C
Pixel Format	RGB10V1Packed (see Errata above)	0x0220001C



26.3.46 RGB10p32

This pixel format defines a 10-bit RGB (red-green-blue), unsigned, packed format. Refer to the Pixel Format Naming Convention for additional information.

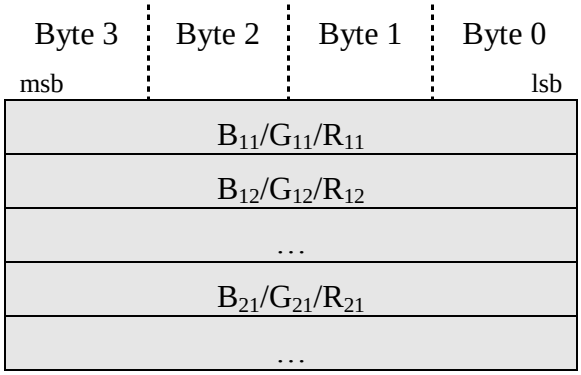
Errata: GigE Vision 1.2 introduced an error in this pixel format by swapping the red and blue component order without changing the pixel value (0x0220001D). This error is seen in [CR25-33s] of GigE Vision 1.2 text under the pixel alignment entry. It should have read "R is position A" and "B is position C".

GigE Vision 2.0 uses the original definition supported by revision 1.0 and 1.1. It also uses the pixel format name from the Pixel Format Naming Convention instead of "RGB10V2Packed".

GigE Vision 1.2 definition must be avoided whenever possible as it created an incompatibility.

[CR-392s] If supported, "RGB10p32" pixel format MUST respect the Pixel Format Naming Convention and MUST follow the layout below. [CR25-33s]

	Description	Value
Color/Mono	GVSP_PIX_COLOR	0x02000000
Bits per pixel	32 bits	0x00200000
Pixel ID	Id of pixel	0x0000001D
Pixel Format	RGB10p32	0x0220001D
Legacy Name	RGB10V2Packed in GigE Vision 1.x (see Errata above)	deprecated

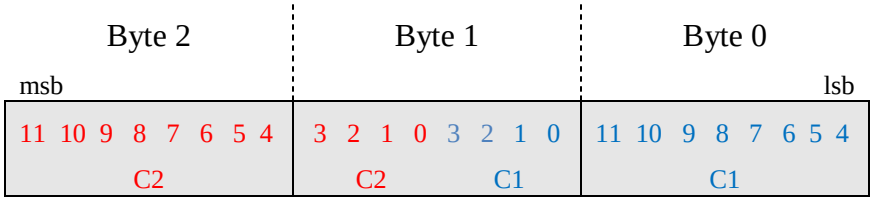


26.3.47 RGB12V1Packed

This pixel format defines a 12-bit RGB (red-green-blue), unsigned, GigE Vision-specific packed format.

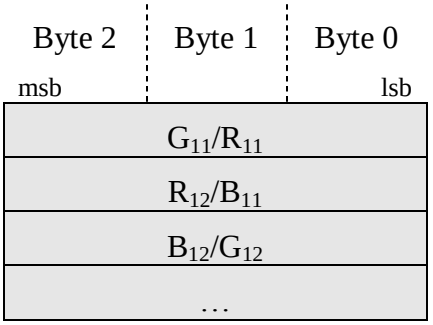
Note: This pixel format **does not** respect the Pixel Format Naming Convention as this format was introduced with GigE Vision 1.0 (prior to creation of the Pixel Format Naming Convention).

With this GigE Vision packing style, two 12-bit color components (C1 and C2) are packed into 24 bits as illustrated below (little-endian layout). Main difference with the grouped style of Pixel Format Naming Convention is the ordering of the bytes (the grouped byte is the last one for the Pixel Format Naming Convention).



[CR-393s] If supported, “RGB12V1Packed” pixel format MUST follow the layout below. [CR25-55s]

	Description	Value
Color/Mono	GVSP_PIX_COLOR	0x02000000
Bits per pixel	36 bits	0x00240000
Pixel ID	Id of pixel	0x00000034
Pixel Format	RGB12V1Packed	0x02240034



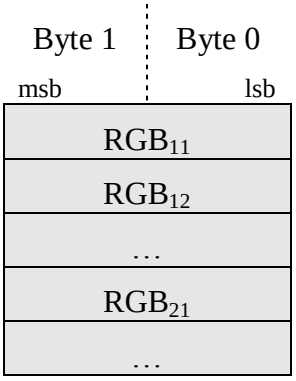
Note: In the above figure, component positioning starts with R_{11} that occupies Byte 0. Other components follow using the R, G, B sequence.

26.3.48 RGB565p

This pixel format defines a RGB (red 5 bits - green 6 bits - blue 5 bits), unsigned, packed format. Refer to the Pixel Format Naming Convention for additional information.

[CR-394s] If supported, "RGB565p" pixel format MUST respect the Pixel Format Naming Convention and MUST follow the layout below. [CR25-57s]

	Description	Value
Color/Mono	GVSP_PIX_COLOR	0x02000000
Bits per pixel	16 bits	0x00100000
Pixel ID	Id of pixel	0x00000035
Pixel Format	RGB565p	0x02100035
Legacy Name	RGB565Packed in GigE Vision 1.x	deprecated

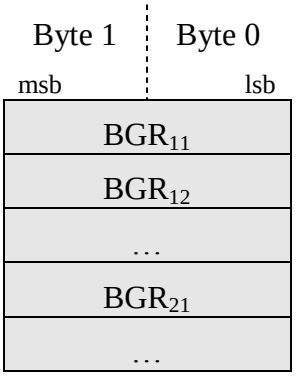


26.3.49 BGR565p

This pixel format defines a BGR (blue 5 bits – green 6 bits – red 5 bits), unsigned, packed format. Refer to the Pixel Format Naming Convention for additional information.

[CR-395s] If supported, "BGR565p" pixel format MUST respect the Pixel Format Naming Convention and MUST follow the layout below. [CR25-58s]

	Description	Value
Color/Mono	GVSP_PIX_COLOR	0x02000000
Bits per pixel	16 bits	0x00100000
Pixel ID	Id of pixel	0x00000036
Pixel Format	BGR565p	0x02100036
Legacy Name	BGR565Packed in GigE Vision 1.x	deprecated



26.3.50 YUV411_8_UYYVYY

This pixel format defines an 8-bit YUV 4:1:1, unsigned, unpacked format. Refer to the Pixel Format Naming Convention for additional information.

[CR-396s] If supported, "YUV411_8_UYYVYY" pixel format MUST respect the Pixel Format Naming Convention and MUST follow the layout below. [CR25-34s]

	Description	Value
Color/Mono	GVSP_PIX_COLOR	0x02000000
Bits per pixel	12 bits	0x000C0000
Pixel ID	Id of pixel	0x0000001E
Pixel Format	YUV411_8_UYYVYY	0x020C001E
Legacy Name	YUV411Packed in GigE Vision 1.x	deprecated

Byte 0

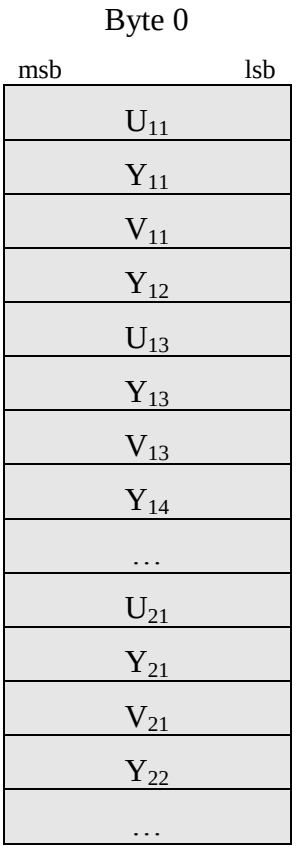
msb lsb
U_{11}
Y_{11}
Y_{12}
V_{11}
Y_{13}
Y_{14}
U_{15}
Y_{15}
Y_{16}
V_{15}
…
U_{21}
Y_{21}
Y_{22}
V_{21}
…

26.3.51 YUV422_8_UYVY

This pixel format defines an 8-bit YUV 4:2:2, unsigned, unpacked format. Refer to the Pixel Format Naming Convention for additional information.

[CR-397s] If supported, "YUV422_8_UYVY" pixel format MUST respect the Pixel Format Naming Convention and MUST follow the layout below. [CR25-35s]

	Description	Value
Color/Mono	GVSP_PIX_COLOR	0x02000000
Bits per pixel	16 bits	0x00100000
Pixel ID	Id of pixel	0x0000001F
Pixel Format	YUV422_8_UYVY	0x0210001F
Legacy Name	YUV422Packed in GigE Vision 1.x	deprecated

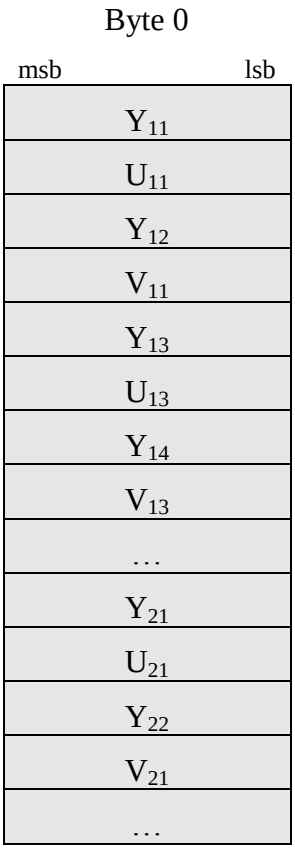


26.3.52 YUV422_8

This pixel format defines an 8-bit YUV 4:2:2, unsigned, unpacked format. Refer to the Pixel Format Naming Convention for additional information.

[CR-398s] If supported, "YUV422_8" pixel format MUST respect the Pixel Format Naming Convention and MUST follow the layout below. [CR25-56s]

	Description	Value
Color/Mono	GVSP_PIX_COLOR	0x02000000
Bits per pixel	16 bits	0x00100000
Pixel ID	Id of pixel	0x00000032
Pixel Format	YUV422_8	0x02100032
Legacy Name	YUYVPacked in GigE Vision 1.x	deprecated



26.3.53 YUV8_UYV

This pixel format defines an 8-bit YUV 4:4:4, unsigned, unpacked format. Refer to the Pixel Format Naming Convention for additional information.

[CR-399s] If supported, "YUV8_UYV" pixel format MUST respect the Pixel Format Naming Convention and MUST follow the layout below. [CR25-36s]

	Description	Value
Color/Mono	GVSP_PIX_COLOR	0x02000000
Bits per pixel	24 bits	0x00180000
Pixel ID	Id of pixel	0x00000020
Pixel Format	YUV8_UYV	0x02180020
Legacy Name	YUV444Packed in GigE Vision 1.x	deprecated

Byte 0

msb lsb
U_{11}
Y_{11}
V_{11}
U_{12}
Y_{12}
V_{12}
…
U_{21}
Y_{21}
V_{21}
…

26.3.54 YCbCr8_CbYCr

This pixel format defines an 8-bit YCbCr 4:4:4, unsigned, unpacked format. This pixel format uses the full range of 256 values for each component. Refer to the Pixel Format Naming Convention for additional information and color space transforms. Note the component order explicitly listed at the end of the pixel format name.

[CR-400s] If supported, "YCbCr8_CbYCr" pixel format MUST respect the Pixel Format Naming Convention and MUST follow the layout below.

	Description	Value
Color/Mono	GVSP_PIX_COLOR	0x02000000
Bits per pixel	24 bits	0x00180000
Pixel ID	Id of pixel	0x0000003A
Pixel Format	YCbCr8_CbYCr	0x0218003A

Byte 0

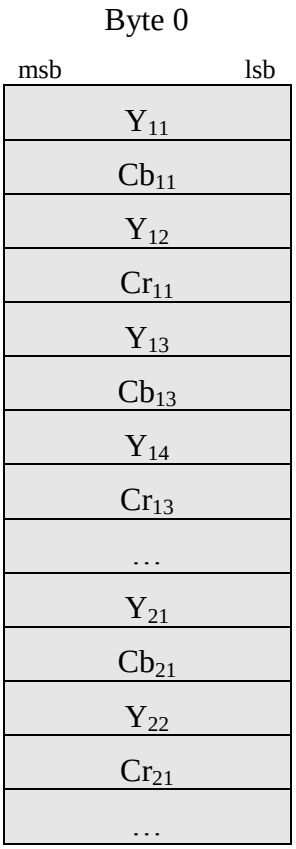
msb	lsb
Cb_{11}	
Y_{11}	
Cr_{11}	
Cb_{12}	
Y_{12}	
Cr_{12}	
…	
Cb_{21}	
Y_{21}	
Cr_{21}	
…	

26.3.55 YCbCr422_8

This pixel format defines an 8-bit YCbCr 4:2:2, unsigned, unpacked format. This pixel format uses the full range of 256 values for each component. Refer to the Pixel Format Naming Convention for additional information and color space transforms.

[CR-401s] If supported, "YCbCr422_8" pixel format MUST respect the Pixel Format Naming Convention and MUST follow the layout below.

	Description	Value
Color/Mono	GVSP_PIX_COLOR	0x02000000
Bits per pixel	16 bits	0x00100000
Pixel ID	Id of pixel	0x0000003B
Pixel Format	YCbCr422_8	0x0210003B

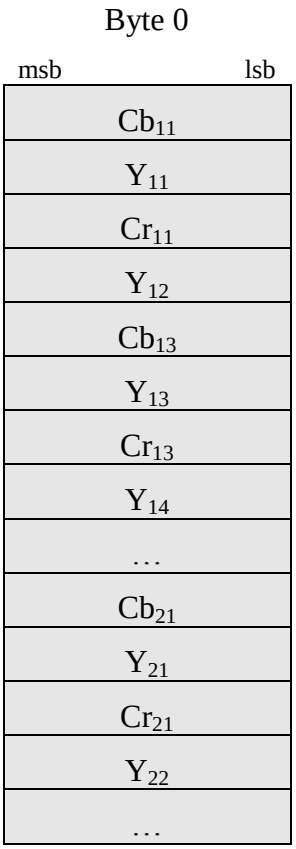


26.3.56 YCbCr422_8_CbYCrY

This pixel format defines an 8-bit YCbCr 4:2:2, unsigned, unpacked format. This pixel format uses the full range of 256 values for each component. Refer to the Pixel Format Naming Convention for additional information and color space transforms. Note the component order explicitly listed at the end of the pixel format name.

[CR-532s] If supported, "YCbCr422_8_CbYCrY" pixel format MUST respect the Pixel Format Naming Convention and MUST follow the layout below.

	Description	Value
Color/Mono	GVSP_PIX_COLOR	0x02000000
Bits per pixel	16 bits	0x00100000
Pixel ID	Id of pixel	0x00000043
Pixel Format	YCbCr422_8_CbYCrY	0x02100043



26.3.57 YCbCr411_8_CbYYCrYY

This pixel format defines an 8-bit YCbCr 4:1:1, unsigned, unpacked format. This pixel format uses the full range of 256 values for each component. Refer to the Pixel Format Naming Convention for additional information and color space transforms. Note the component order explicitly listed at the end of the pixel format name.

[CR-402s] If supported, "YCbCr411_8_CbYYCrYY" pixel format MUST respect the Pixel Format Naming Convention and MUST follow the layout below.

	Description	Value
Color/Mono	GVSP_PIX_COLOR	0x02000000
Bits per pixel	12 bits	0x000C0000
Pixel ID	Id of pixel	0x0000003C
Pixel Format	YCbCr411_8_CbYYCrYY	0x020C003C

Byte 0

msb lsb
Cb_{11}
Y_{11}
Y_{12}
Cr_{11}
Y_{13}
Y_{14}
Cb_{15}
Y_{15}
Y_{16}
Cr_{15}
Y_{17}
Y_{18}
…
Cb_{21}
Y_{21}
Y_{22}
Cr_{21}
Y_{23}
…

26.3.58 YCbCr601_8_CbYCr

This pixel format defines an 8-bit YCbCr 4:4:4, unsigned, unpacked format. This pixel format uses the color space specified in ITU-R BT.601. Refer to the Pixel Format Naming Convention for additional information and color space transforms. Note the component order explicitly listed at the end of the pixel format name.

[CR-403s] If supported, "YCbCr601_8_CbYCr" pixel format MUST respect the Pixel Format Naming Convention and MUST follow the layout below.

	Description	Value
Color/Mono	GVSP_PIX_COLOR	0x02000000
Bits per pixel	24 bits	0x00180000
Pixel ID	Id of pixel	0x0000003D
Pixel Format	YCbCr601_8_CbYCr	0x0218003D

Byte 0

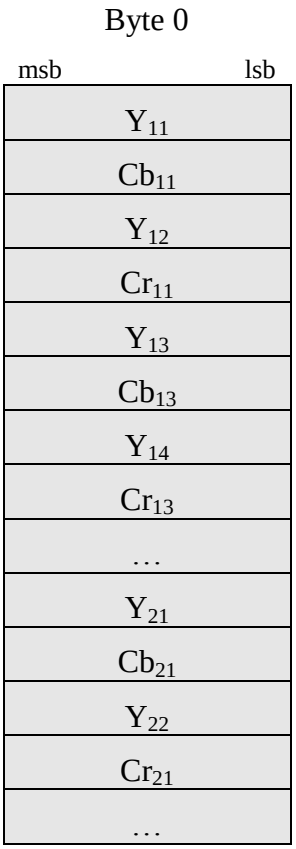
msb lsb
Cb_{11}
Y_{11}
Cr_{11}
Cb_{12}
Y_{12}
Cr_{12}
…
Cb_{21}
Y_{21}
Cr_{21}
…

26.3.59 YCbCr601_422_8

This pixel format defines an 8-bit YCbCr 4:2:2, unsigned, unpacked format. This pixel format uses the color space specified in ITU-R BT.601. Refer to the Pixel Format Naming Convention for additional information and color space transforms.

[CR-404s] If supported, "YCbCr601_422_8" pixel format MUST respect the Pixel Format Naming Convention and MUST follow the layout below.

	Description	Value
Color/Mono	GVSP_PIX_COLOR	0x02000000
Bits per pixel	16 bits	0x00100000
Pixel ID	Id of pixel	0x0000003E
Pixel Format	YCbCr601_422_8	0x0210003E

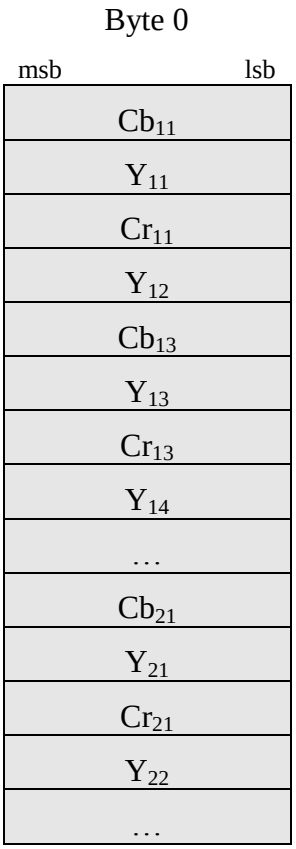


26.3.60 YCbCr601_422_8_CbYCrY

This pixel format defines an 8-bit YCbCr 4:2:2, unsigned, unpacked format. This pixel format uses the color space specified in ITU-R BT.601. Refer to the Pixel Format Naming Convention for additional information and color space transforms. Note the component order explicitly listed at the end of the pixel format name.

[CR-533s] If supported, "YCbCr601_422_8_CbYCrY" pixel format MUST respect the Pixel Format Naming Convention and MUST follow the layout below.

	Description	Value
Color/Mono	GVSP_PIX_COLOR	0x02000000
Bits per pixel	16 bits	0x00100000
Pixel ID	Id of pixel	0x00000044
Pixel Format	YCbCr601_422_8_CbYCrY	0x02100044



26.3.61 YCbCr601_411_8_CbYYCrYY

This pixel format defines an 8-bit YCbCr 4:1:1, unsigned, unpacked format. This pixel format uses the color space specified in ITU-R BT.601. Refer to the Pixel Format Naming Convention for additional information and color space transforms. Note the component order explicitly listed at the end of the pixel format name.

[CR-405s] If supported, "YCbCr601_411_8_CbYYCrYY" pixel format MUST respect the Pixel Format Naming Convention and MUST follow the layout below.

	Description	Value
Color/Mono	GVSP_PIX_COLOR	0x02000000
Bits per pixel	12 bits	0x000C0000
Pixel ID	Id of pixel	0x0000003F
Pixel Format	YCbCr601_411_8_CbYYCrYY	0x020C003F

Byte 0

msb lsb
Cb_{11}
Y_{11}
Y_{12}
Cr_{11}
Y_{13}
Y_{14}
Cb_{15}
Y_{15}
Y_{16}
Cr_{15}
Y_{17}
Y_{18}
…
Cb_{21}
Y_{21}
Y_{22}
Cr_{21}
Y_{23}
…

26.3.62 YCbCr709_8_CbYCr

This pixel format defines an 8-bit YCbCr 4:4:4, unsigned, unpacked format. This pixel format uses the color space specified in ITU-R BT.709. Refer to the Pixel Format Naming Convention for additional information and color space transforms. Note the component order explicitly listed at the end of the pixel format name.

[CR-406s] If supported, "YCbCr709_8_CbYCr" pixel format MUST respect the Pixel Format Naming Convention and MUST follow the layout below.

	Description	Value
Color/Mono	GVSP_PIX_COLOR	0x02000000
Bits per pixel	24 bits	0x00180000
Pixel ID	Id of pixel	0x00000040
Pixel Format	YCbCr709_8_CbYCr	0x02180040

Byte 0

msb lsb
Cb_{11}
Y_{11}
Cr_{11}
Cb_{12}
Y_{12}
Cr_{12}
…
Cb_{21}
Y_{21}
Cr_{21}
…

26.3.63 YCbCr709_422_8

This pixel format defines an 8-bit YCbCr 4:2:2, unsigned, unpacked format. This pixel format uses the color space specified in ITU-R BT.709. Refer to the Pixel Format Naming Convention for additional information and color space transforms.

[CR-407s] If supported, "YCbCr709_422_8" pixel format MUST respect the Pixel Format Naming Convention and MUST follow the layout below.

	Description	Value
Color/Mono	GVSP_PIX_COLOR	0x02000000
Bits per pixel	16 bits	0x00100000
Pixel ID	Id of pixel	0x00000041
Pixel Format	YCbCr709_422_8	0x02100041

Byte 0

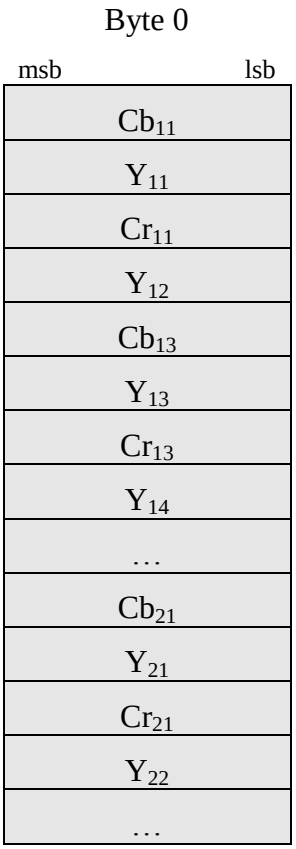
msb	lsb
Y_{11}	
Cb_{11}	
Y_{12}	
Cr_{11}	
Y_{13}	
Cb_{13}	
Y_{14}	
Cr_{13}	
…	
Y_{21}	
Cb_{21}	
Y_{22}	
Cr_{21}	
…	

26.3.64 YCbCr709_422_8_CbYCrY

This pixel format defines an 8-bit YCbCr 4:2:2, unsigned, unpacked format. This pixel format uses the color space specified in ITU-R BT.709. Refer to the Pixel Format Naming Convention for additional information and color space transforms. Note the component order explicitly listed at the end of the pixel format name.

[CR-534s] If supported, "YCbCr709_422_8_CbYCrY" pixel format MUST respect the Pixel Format Naming Convention and MUST follow the layout below.

	Description	Value
Color/Mono	GVSP_PIX_COLOR	0x02000000
Bits per pixel	16 bits	0x00100000
Pixel ID	Id of pixel	0x00000045
Pixel Format	YCbCr709_422_8_CbYCrY	0x02100045



26.3.65 YCbCr709_411_8_CbYYCrYY

This pixel format defines an 8-bit YCbCr 4:1:1, unsigned, unpacked format. This pixel format uses the color space specified in ITU-R BT.709. Refer to the Pixel Format Naming Convention for additional information and color space transforms. Note the component order explicitly listed at the end of the pixel format name.

[CR-408s] If supported, "YCbCr709_411_8_CbYYCrYY" pixel format MUST respect the Pixel Format Naming Convention and MUST follow the layout below.

	Description	Value
Color/Mono	GVSP_PIX_COLOR	0x02000000
Bits per pixel	12 bits	0x000C0000
Pixel ID	Id of pixel	0x00000042
Pixel Format	YCbCr709_411_8_CbYYCrYY	0x020C0042

Byte 0

msb lsb
Cb_{11}
Y_{11}
Y_{12}
Cr_{11}
Y_{13}
Y_{14}
Cb_{15}
Y_{15}
Y_{16}
Cr_{15}
Y_{17}
Y_{18}
…
Cb_{21}
Y_{21}
Y_{22}
Cr_{21}
Y_{23}
…

26.3.66 RGB8_Planar

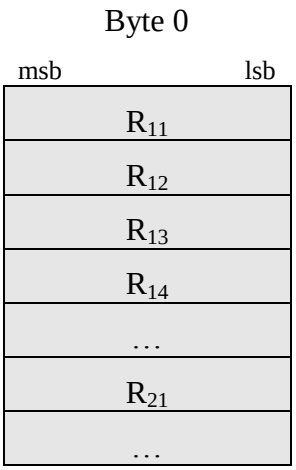
This pixel format defines an 8-bit RGB (red-green-blue), unsigned, unpacked, planar format where each color plane is transmitted on a different stream channel.

[CR-409s] If supported, “RGB8_Planar” pixel format MUST respect the Pixel Format Naming Convention and MUST follow the layout below. [CR25-37s]

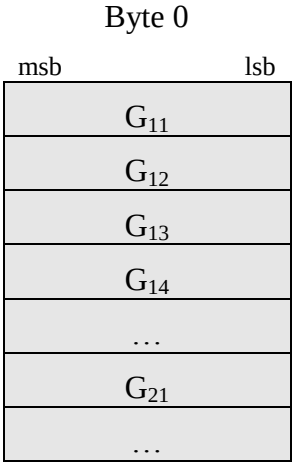
	Description	Value
Color/Mono	GVSP_PIX_COLOR	0x02000000
Bits per pixel	24 bits	0x00180000
Pixel ID	Id of pixel	0x00000021
Pixel Format	RGB8_Planar	0x02180021
Legacy Name	RGB8Planar in GigE Vision 1.x	deprecated

This pixel format requires 3 consecutive stream channels.

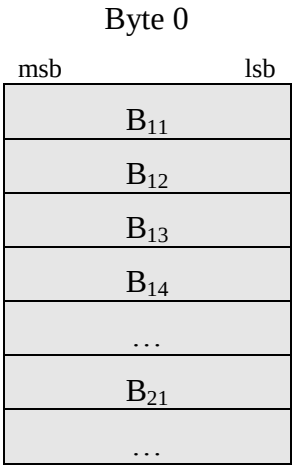
Red plane (stream channel N):



Green plane (stream channel N+1):



Blue plane (stream channel N+2):



26.3.67 RGB10_Planar

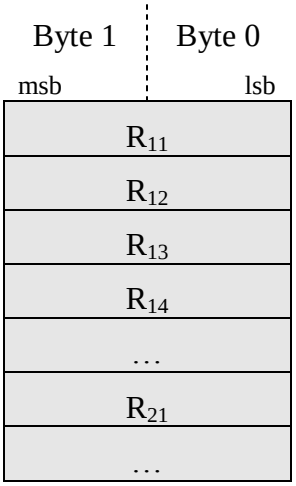
This pixel format defines a 10-bit RGB (red-green-blue), unsigned, unpacked, planar format where each color plane is transmitted on a different stream channel.

[CR-410s] If supported, "RGB10_Planar" pixel format MUST respect the Pixel Format Naming Convention and MUST follow the layout below. [CR25-38s]

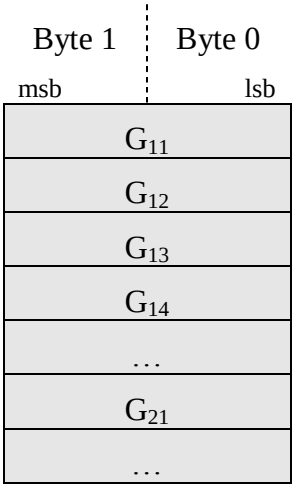
	Description	Value
Color/Mono	GVSP_PIX_COLOR	0x02000000
Bits per pixel	48 bits	0x00300000
Pixel ID	Id of pixel	0x00000022
Pixel Format	RGB10_Planar	0x02300022
Legacy Name	RGB10Planar in GigE Vision 1.x	deprecated

This pixel format requires 3 consecutive stream channels.

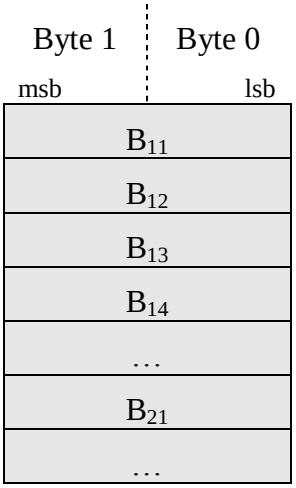
Red plane (stream channel N):



Green plane (stream channel N+1):



Blue plane (stream channel N+2):



26.3.68 RGB12_Planar

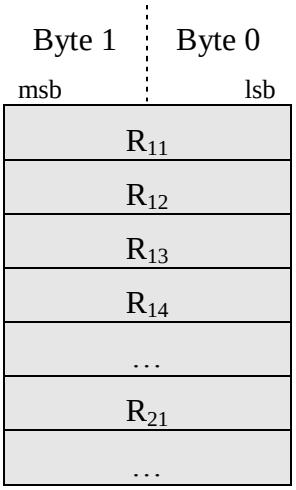
This pixel format defines a 12-bit RGB (red-green-blue), unsigned, unpacked, planar format where each color plane is transmitted on a different stream channel.

[CR-411s] If supported, "RGB12_Planar" pixel format MUST respect the Pixel Format Naming Convention and MUST follow the layout below. [CR25-39s]

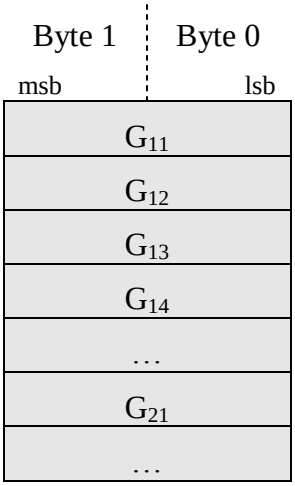
	Description	Value
Color/Mono	GVSP_PIX_COLOR	0x02000000
Bits per pixel	48 bits	0x00300000
Pixel ID	Id of pixel	0x00000023
Pixel Format	RGB12_Planar	0x02300023
Legacy Name	RGB12Planar in GigE Vision 1.x	deprecated

This pixel format requires 3 consecutive stream channels.

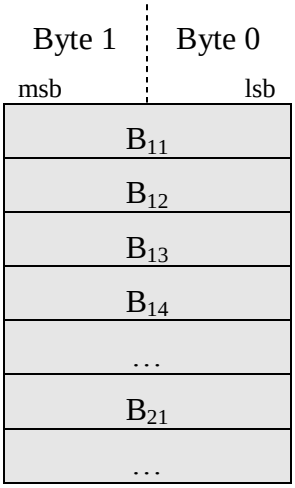
Red plane (stream channel N):



Green plane (stream channel N+1):



Blue plane (stream channel N+2):



26.3.69 RGB16_Planar

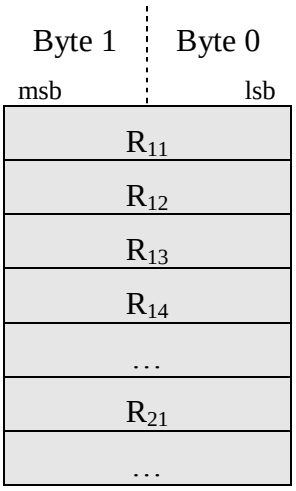
This pixel format defines a 16-bit RGB (red-green-blue), unsigned, unpacked, planar format where each color plane is transmitted on a different stream channel.

[CR-412s] If supported, "RGB16_Planar" pixel format MUST respect the Pixel Format Naming Convention and MUST follow the layout below. [CR25-40s]

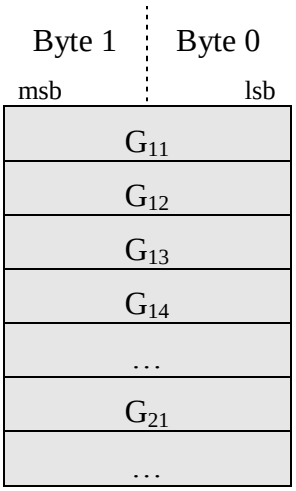
	Description	Value
Color/Mono	GVSP_PIX_COLOR	0x02000000
Bits per pixel	48 bits	0x00300000
Pixel ID	Id of pixel	0x00000024
Pixel Format	RGB16_Planar	0x02300024
Legacy Name	RGB16Planar in GigE Vision 1.x	deprecated

This pixel format requires 3 consecutive stream channels.

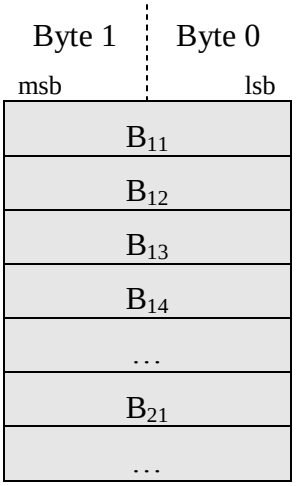
Red plane (stream channel N):



Green plane (stream channel N+1):



Blue plane (stream channel N+2):



27 Pixel Format Defines

The following provides #define for the various pixel formats supported by GVSP. Each pixel format is represented by a 32-bit value. The upper 8-bit indicates the color. The second upper 8-bit indicates the number of bits occupied by a pixel (including any padding). This can be used to quickly compute the amount of memory required to store an image using this pixel format.

> **Note**: With the introduction of the Pixel Format Naming Convention, some `#defines` listed in this section have changed from version 1.2 of the specification.

```
//===================================================
// PIXEL FORMATS
//===================================================
// Indicate if pixel is monochrome or RGB
#define GVSP_PIX_MONO                           0x01000000
#define GVSP_PIX_RGB                            0x02000000 // deprecated in version 1.1
#define GVSP_PIX_COLOR                          0x02000000
#define GVSP_PIX_CUSTOM                         0x80000000
#define GVSP_PIX_COLOR_MASK                     0xFF000000

// Indicate effective number of bits occupied by the pixel (including padding).
// This can be used to compute amount of memory required to store an image.
#define GVSP_PIX_OCCUPY1BIT                     0x00010000
#define GVSP_PIX_OCCUPY2BIT                     0x00020000
#define GVSP_PIX_OCCUPY4BIT                     0x00040000
#define GVSP_PIX_OCCUPY8BIT                     0x00080000
#define GVSP_PIX_OCCUPY12BIT                    0x000C0000
#define GVSP_PIX_OCCUPY16BIT                    0x00100000
#define GVSP_PIX_OCCUPY24BIT                    0x00180000
#define GVSP_PIX_OCCUPY32BIT                    0x00200000
#define GVSP_PIX_OCCUPY36BIT                    0x00240000
#define GVSP_PIX_OCCUPY48BIT                    0x00300000
#define GVSP_PIX_EFFECTIVE_PIXEL_SIZE_MASK      0x00FF0000
#define GVSP_PIX_EFFECTIVE_PIXEL_SIZE_SHIFT     16

// Pixel ID: lower 16-bit of the pixel formats
#define GVSP_PIX_ID_MASK                        0x0000FFFF
#define GVSP_PIX_COUNT                          0x46          // next Pixel ID available
```

27.1 Mono buffer format defines

```
#define GVSP_PIX_MONO1P                 (GVSP_PIX_MONO | GVSP_PIX_OCCUPY1BIT   | 0x0037)
#define GVSP_PIX_MONO2P                 (GVSP_PIX_MONO | GVSP_PIX_OCCUPY2BIT   | 0x0038)
#define GVSP_PIX_MONO4P                 (GVSP_PIX_MONO | GVSP_PIX_OCCUPY4BIT   | 0x0039)
#define GVSP_PIX_MONO8                  (GVSP_PIX_MONO | GVSP_PIX_OCCUPY8BIT   | 0x0001)
#define GVSP_PIX_MONO8S                 (GVSP_PIX_MONO | GVSP_PIX_OCCUPY8BIT   | 0x0002)
#define GVSP_PIX_MONO10                 (GVSP_PIX_MONO | GVSP_PIX_OCCUPY16BIT  | 0x0003)
#define GVSP_PIX_MONO10_PACKED          (GVSP_PIX_MONO | GVSP_PIX_OCCUPY12BIT  | 0x0004)
#define GVSP_PIX_MONO12                 (GVSP_PIX_MONO | GVSP_PIX_OCCUPY16BIT  | 0x0005)
#define GVSP_PIX_MONO12_PACKED          (GVSP_PIX_MONO | GVSP_PIX_OCCUPY12BIT  | 0x0006)
#define GVSP_PIX_MONO14                 (GVSP_PIX_MONO | GVSP_PIX_OCCUPY16BIT  | 0x0025)
#define GVSP_PIX_MONO16                 (GVSP_PIX_MONO | GVSP_PIX_OCCUPY16BIT  | 0x0007)
```

27.2 Bayer buffer format defines

```
#define GVSP_PIX_BAYGR8                (GVSP_PIX_MONO | GVSP_PIX_OCCUPY8BIT   | 0x0008)
#define GVSP_PIX_BAYRG8                (GVSP_PIX_MONO | GVSP_PIX_OCCUPY8BIT   | 0x0009)
#define GVSP_PIX_BAYGB8                (GVSP_PIX_MONO | GVSP_PIX_OCCUPY8BIT   | 0x000A)
#define GVSP_PIX_BAYBG8                (GVSP_PIX_MONO | GVSP_PIX_OCCUPY8BIT   | 0x000B)
#define GVSP_PIX_BAYGR10               (GVSP_PIX_MONO | GVSP_PIX_OCCUPY16BIT  | 0x000C)
#define GVSP_PIX_BAYRG10               (GVSP_PIX_MONO | GVSP_PIX_OCCUPY16BIT  | 0x000D)
#define GVSP_PIX_BAYGB10               (GVSP_PIX_MONO | GVSP_PIX_OCCUPY16BIT  | 0x000E)
#define GVSP_PIX_BAYBG10               (GVSP_PIX_MONO | GVSP_PIX_OCCUPY16BIT  | 0x000F)
#define GVSP_PIX_BAYGR12               (GVSP_PIX_MONO | GVSP_PIX_OCCUPY16BIT  | 0x0010)
#define GVSP_PIX_BAYRG12               (GVSP_PIX_MONO | GVSP_PIX_OCCUPY16BIT  | 0x0011)
#define GVSP_PIX_BAYGB12               (GVSP_PIX_MONO | GVSP_PIX_OCCUPY16BIT  | 0x0012)
#define GVSP_PIX_BAYBG12               (GVSP_PIX_MONO | GVSP_PIX_OCCUPY16BIT  | 0x0013)
#define GVSP_PIX_BAYGR10_PACKED        (GVSP_PIX_MONO | GVSP_PIX_OCCUPY12BIT  | 0x0026)
#define GVSP_PIX_BAYRG10_PACKED        (GVSP_PIX_MONO | GVSP_PIX_OCCUPY12BIT  | 0x0027)
#define GVSP_PIX_BAYGB10_PACKED        (GVSP_PIX_MONO | GVSP_PIX_OCCUPY12BIT  | 0x0028)
#define GVSP_PIX_BAYBG10_PACKED        (GVSP_PIX_MONO | GVSP_PIX_OCCUPY12BIT  | 0x0029)
#define GVSP_PIX_BAYGR12_PACKED        (GVSP_PIX_MONO | GVSP_PIX_OCCUPY12BIT  | 0x002A)
#define GVSP_PIX_BAYRG12_PACKED        (GVSP_PIX_MONO | GVSP_PIX_OCCUPY12BIT  | 0x002B)
#define GVSP_PIX_BAYGB12_PACKED        (GVSP_PIX_MONO | GVSP_PIX_OCCUPY12BIT  | 0x002C)
#define GVSP_PIX_BAYBG12_PACKED        (GVSP_PIX_MONO | GVSP_PIX_OCCUPY12BIT  | 0x002D)
#define GVSP_PIX_BAYGR16               (GVSP_PIX_MONO | GVSP_PIX_OCCUPY16BIT  | 0x002E)
#define GVSP_PIX_BAYRG16               (GVSP_PIX_MONO | GVSP_PIX_OCCUPY16BIT  | 0x002F)
#define GVSP_PIX_BAYGB16               (GVSP_PIX_MONO | GVSP_PIX_OCCUPY16BIT  | 0x0030)
#define GVSP_PIX_BAYBG16               (GVSP_PIX_MONO | GVSP_PIX_OCCUPY16BIT  | 0x0031)
```

27.3 RGB Packed buffer format defines

```
#define GVSP_PIX_RGB8                  (GVSP_PIX_COLOR | GVSP_PIX_OCCUPY24BIT | 0x0014)
#define GVSP_PIX_BGR8                  (GVSP_PIX_COLOR | GVSP_PIX_OCCUPY24BIT | 0x0015)
#define GVSP_PIX_RGBA8                 (GVSP_PIX_COLOR | GVSP_PIX_OCCUPY32BIT | 0x0016)
#define GVSP_PIX_BGRA8                 (GVSP_PIX_COLOR | GVSP_PIX_OCCUPY32BIT | 0x0017)
#define GVSP_PIX_RGB10                 (GVSP_PIX_COLOR | GVSP_PIX_OCCUPY48BIT | 0x0018)
#define GVSP_PIX_BGR10                 (GVSP_PIX_COLOR | GVSP_PIX_OCCUPY48BIT | 0x0019)
#define GVSP_PIX_RGB12                 (GVSP_PIX_COLOR | GVSP_PIX_OCCUPY48BIT | 0x001A)
#define GVSP_PIX_BGR12                 (GVSP_PIX_COLOR | GVSP_PIX_OCCUPY48BIT | 0x001B)
#define GVSP_PIX_RGB16                 (GVSP_PIX_COLOR | GVSP_PIX_OCCUPY48BIT | 0x0033)
#define GVSP_PIX_RGB10V1_PACKED        (GVSP_PIX_COLOR | GVSP_PIX_OCCUPY32BIT | 0x001C)
#define GVSP_PIX_RGB10P32              (GVSP_PIX_COLOR | GVSP_PIX_OCCUPY32BIT | 0x001D)
#define GVSP_PIX_RGB12V1_PACKED        (GVSP_PIX_COLOR | GVSP_PIX_OCCUPY36BIT | 0X0034)
#define GVSP_PIX_RGB565P               (GVSP_PIX_COLOR | GVSP_PIX_OCCUPY16BIT | 0x0035)
#define GVSP_PIX_BGR565P               (GVSP_PIX_COLOR | GVSP_PIX_OCCUPY16BIT | 0X0036)
```

27.4 YUV and YCbCr Packed buffer format defines

```
#define GVSP_PIX_YUV411_8_UYYVYY       (GVSP_PIX_COLOR | GVSP_PIX_OCCUPY12BIT | 0x001E)
#define GVSP_PIX_YUV422_8_UYVY         (GVSP_PIX_COLOR | GVSP_PIX_OCCUPY16BIT | 0x001F)
#define GVSP_PIX_YUV422_8              (GVSP_PIX_COLOR | GVSP_PIX_OCCUPY16BIT | 0x0032)
#define GVSP_PIX_YUV8_UYV              (GVSP_PIX_COLOR | GVSP_PIX_OCCUPY24BIT | 0x0020)
#define GVSP_PIX_YCBCR8_CBYCR          (GVSP_PIX_COLOR | GVSP_PIX_OCCUPY24BIT | 0x003A)
#define GVSP_PIX_YCBCR422_8            (GVSP_PIX_COLOR | GVSP_PIX_OCCUPY16BIT | 0x003B)
#define GVSP_PIX_YCBCR422_8_CBYCRY     (GVSP_PIX_COLOR | GVSP_PIX_OCCUPY16BIT | 0x0043)
#define GVSP_PIX_YCBCR411_8_CBYYCRYY   (GVSP_PIX_COLOR | GVSP_PIX_OCCUPY12BIT | 0x003C)
#define GVSP_PIX_YCBCR601_8_CBYCR      (GVSP_PIX_COLOR | GVSP_PIX_OCCUPY24BIT | 0x003D)
#define GVSP_PIX_YCBCR601_422_8        (GVSP_PIX_COLOR | GVSP_PIX_OCCUPY16BIT | 0x003E)
#define GVSP_PIX_YCBCR601_422_8_CBYCRY (GVSP_PIX_COLOR | GVSP_PIX_OCCUPY16BIT | 0x0044)
#define GVSP_PIX_YCBCR601_411_8_CBYYCRYY (GVSP_PIX_COLOR | GVSP_PIX_OCCUPY12BIT | 0x003F)
#define GVSP_PIX_YCBCR709_8_CBYCR      (GVSP_PIX_COLOR | GVSP_PIX_OCCUPY24BIT | 0x0040)
#define GVSP_PIX_YCBCR709_422_8        (GVSP_PIX_COLOR | GVSP_PIX_OCCUPY16BIT | 0x0041)
```

```
#define GVSP_PIX_YCBCR709_422_8_CBYCRY  (GVSP_PIX_COLOR | GVSP_PIX_OCCUPY16BIT | 0x0045)
#define GVSP_PIX_YCBCR709_411_8_CBYYCRYY (GVSP_PIX_COLOR | GVSP_PIX_OCCUPY12BIT | 0x0042)
```

27.5 RGB Planar buffer format defines

```
#define GVSP_PIX_RGB8_PLANAR            (GVSP_PIX_COLOR | GVSP_PIX_OCCUPY24BIT | 0x0021)
#define GVSP_PIX_RGB10_PLANAR           (GVSP_PIX_COLOR | GVSP_PIX_OCCUPY48BIT | 0x0022)
#define GVSP_PIX_RGB12_PLANAR           (GVSP_PIX_COLOR | GVSP_PIX_OCCUPY48BIT | 0x0023)
#define GVSP_PIX_RGB16_PLANAR           (GVSP_PIX_COLOR | GVSP_PIX_OCCUPY48BIT | 0x0024)
```

PART 4 – Bootstrap Registers

28 Bootstrap Registers

A number of bootstrap registers are defined by this specification to allow configuration of a device. These registers are common to GigE Vision devices and are located at fixed addresses specified by this text. But a device can also allocate non-bootstrap registers in the device-specific memory space starting at address 0xA000. These manufacturer-specific registers are not defined by this specification and are typically advertised through the XML device description file.

[R-413cd] All mandatory bootstrap registers MUST be present on all GigE Vision compliant devices (though some specific bootstrap registers are optional when indicated). These bootstrap registers are listed in Table 28-1. [R27-1cd]

[O-414cd] Bootstrap registers SHOULD be provided in the XML device description file and respect the rules defined in the GenICam™ Standard Features Naming Convention for GigE Vision. [O27-37cd]

Having the bootstrap registers in the XML file provides wider possibility for value validation since XML features may have information about their minimum, maximum and increment.

Device registers are accessed using GVCP. Manufacturer-specific registers start after bootstrap registers memory-space.

[R-415cd] All device registers (bootstrap and manufacturer-specific) MUST be 32-bit aligned. [R27-2cd]

[R-416cd] All device bootstrap registers MUST be visible as big-endian.

Note: Endianess of the manufacturer-specific registers is not defined by this specification.

Bootstrap registers should be accessed by the application using READREG and WRITEREG messages, with the exception of character strings which should be accessed using READMEM and WRITEMEM messages.

[R-417ca] When information spans multiple 32-bit registers, then the register with the lowest address MUST be accessed first, followed by the other registers sequentially until the register with the highest address is accessed. [R27-4ca]

The above requirement would facilitate migration to wider memory architecture than the current 32 bits (such as 64 bits).

[R-418cd] All strings stored in the bootstrap registers space MUST match the character set specified by Device Mode register at address 0x0004 and be NULL-terminated unless the string uses the whole memory section of the bootstrap register. The NULL-termination character is implicit if the string uses the whole register space. [R27-5cd]

Note: In GigE Vision 1.x, the NULL character at the end of a string had to be explicit. With GigE Vision 2.x, the specification relaxes this constraint and asks the application software to append a NULL character if the string consumes the full memory area allocated to the string.

Some registers might not be available on a given device. They are highlighted by the "optional" designation. In this case, the application must refer to the device capabilities indicating the number of message channels, stream channels and network interfaces. Also, optional GVCP commands are indicated by a bitfield register where each bit represents a different optional command.

[R-419cd] All reserved fields of bootstrap registers MUST be set to 0. [R27-6cd]

[R-420ca] An application MUST ignore the reserved fields of the bootstrap registers.

A number of restrictions apply to the address and access type (read or write) of the device registers.

[R-421cd] Trying to access an unsupported bootstrap register MUST return GEV_STATUS_INVALID_ADDRESS. [R27-7cd]

[R-422cd] A device MUST reserve the address space ranging from 0x0000 to 0x9FFF inclusively for bootstrap registers described by this specification. Manufacturer-specific register MUST be allocated after that address range.

[R-423cd] Trying to read from a write-only register MUST return GEV_STATUS_ACCESS_DENIED. Trying to write to a read only register MUST return GEV_STATUS_WRITE_PROTECT. [R27-8cd]

[O-424ca] An application should use the READMEM message to retrieve strings from the bootstrap registers. [O27-9ca]

[CR-425cd] When a device does not support a given optional string bootstrap register, READMEM to this string register MUST return GEV_STATUS_INVALID_ADDRESS with *length* field of the header set to 0. [CR27-16cd]

[CR-426cd] When a device does not support a given optional string bootstrap register, WRITEMEM to this string register MUST return GEV_STATUS_INVALID_ADDRESS with the *index* field of the header set to 0. [CR27-17cd]

[O-427cd] When an application tries to write an unsupported value in a bootstrap register, then the device SHOULD return a GEV_STATUS_INVALID_PARAMETER in the acknowledge. In this situation, the register must maintain its previous value.

[O-428cd] When an application tries to access a register, but the device is momentarily not in a position to execute the request, the device SHOULD return GEV_STATUS_BUSY.

The previous objective can be used as an alternative to PENDING_ACK, as mentioned in section 10.7.

The following table lists the bootstrap registers. It indicates if a register is (M)andatory or (O)ptional. It also indicates the type of access (R)ead and (W)rite that can be performed.

Table 28-1: Bootstrap Registers

Address	Name	Support	Type	Length (bytes)	Description
0x0000	Version	M	R	4	Version of the GigE Standard with which the device is compliant The two most-significant bytes are allocated to the major number of the version, the two least-significant bytes for the minor number.
0x0004	Device Mode	M	R	4	Information about device mode of operation.
0x0008	Device MAC Address – High (Network interface #0)	M	R	4	The two most-significant bytes of this area are reserved and will return 0. Upper 2 bytes of the MAC address are stored in the lower 2 significant bytes of this register.
0x000C	Device MAC Address – Low (Network interface #0)	M	R	4	Lower 4 bytes of the MAC address
0x0010	Network Interface Capability (Network interface #0)	M	R	4	Supported IP Configuration and PAUSE schemes. Bits can be OR-ed. All other bits are reserved and set to 0. DHCP and LLA bits must be on.
0x0014	Network Interface Configuration (Network interface #0)	M	R/W	4	Activated IP Configuration and PAUSE schemes. Bits can be OR-ed. LLA is always activated and is read-only. PAUSE settings might be hard-coded.
0x0018	Reserved	-	-	-	
0x0024	Current IP Address (Network interface #0)	M	R	4	Current IP address of this device on its first interface.
0x0028	Reserved	-	-	-	
0x0034	Current Subnet Mask (Network interface #0)	M	R	4	Current subnet mask used by this device on its first interface.
0x0038	Reserved	-	-	-	
0x0044	Current Default Gateway (Network interface #0)	M	R	4	Current default gateway used by this device on its first interface.
0x0048	Manufacturer Name	M	R	32	Provides the name of the manufacturer of the device. This string is provided in the *manufacturer_name* field of the DISCOVERY_ACK message. String using the format in *character_set_index* of Device Mode register.
0x0068	Model Name	M	R	32	Provides the model name of the device. This string is provided in the *model_name* field of the DISCOVERY_ACK message. String using the format in *character_set_index* of Device Mode register.

Address	Name	Support	Type	Length (bytes)	Description
0x0088	Device Version	M	R	32	Provides the version of the device. This string is provided in the *device_version* field of the DISCOVERY_ACK message. String using the format in *character_set_index* of Device Mode register.
0x00A8	Manufacturer Info	M	R	48	Provides extended manufacturer information about the device. This string is provided in the *manufacturer_specific_info* field of the DISCOVERY_ACK message. String using the format in *character_set_index* of Device Mode register.
0x00D8	Serial Number	O	R	16	When supported, this string contains the serial number of the device. It can be used to identify the device. This optional register is supported when bit 1 of the GVCP Capability register is set. This string is provided in the *serial_number* field of the DISCOVERY_ACK message (set to all NULL when not supported). String using the format in *character_set_index* of Device Mode register.
0x00E8	User-defined Name	O	R/W	16	When supported, this string contains a user-programmable name to assign to the device. It can be used to identify the device. This optional register is supported when bit 0 of the GVCP Capability register is set. This string is provided in the *user_defined_name* field of the DISCOVERY_ACK message (set to all NULL when not supported). String using the format in *character_set_index* of Device Mode register.
0x00F8	Reserved	-	-	-	
0x0200	First URL	M	R	512	String using the format in *character_set_index* of Device Mode register.
0x0400	Second URL	M	R	512	String using the format in *character_set_index* of Device Mode register.
0x0600	Number of Network Interfaces	M	R	4	Indicates the number of physical network interfaces on this device. A device must have at least one network interface.
0x0604	Reserved	-	-	-	
0x064C	Persistent IP Address (Network interface #0)	O	R/W	4	Only available if Persistent IP is supported by the device.
0x0650	Reserved	-	-	-	
0x065C	Persistent Subnet Mask (Network interface #0)	O	R/W	4	Only available if Persistent IP is supported by the device.
0x0660	Reserved	-	-	-	

Address	Name	Support	Type	Length (bytes)	Description
0x066C	Persistent Default Gateway (Network interface #0)	O	R/W	4	Only available if Persistent IP is supported by the device.
0x0670	Link Speed (Network interface #0)	O	R	4	32-bit value indicating current Ethernet link speed in Mbits per second on the first interface.
0x0674	Reserved	-	-	-	
0x0680	Device MAC Address – High (Network interface #1)	O	R	4	The two most-significant bytes of this area are reserved and will return 0. Upper 2 bytes of the MAC address are stored in the lower 2 significant bytes of this register.
0x0684	Device MAC Address – Low (Network interface #1)	O	R	4	Lower 4 bytes of the MAC address
0x0688	Network Interface Capability (Network interface #1)	O	R	4	Supported IP Configuration and PAUSE schemes. Bits can be OR-ed. All other bits are reserved and set to 0. DHCP and LLA bits must be on.
0x068C	Network Interface Configuration (Network interface #1)	O	R/W	4	Activated IP Configuration and PAUSE schemes. Bits can be OR-ed. LLA is always activated and is read-only. PAUSE settings might be hard-coded.
0x0690	Reserved	-	-	-	
0x069C	Current IP Address (Network interface #1)	O	R	4	Current IP address of this device on its second interface.
0x06A0	Reserved	-	-	-	
0x06AC	Current Subnet Mask (Network interface #1)	O	R	4	Current subnet mask used by this device on its second interface.
0x06B0	Reserved	-	-	-	
0x06BC	Current Default Gateway (Network interface #1)	O	R	4	Current default gateway used by this device on its second interface.
0x06C0	Reserved	-	-	-	
0x06CC	Persistent IP Address (Network interface #1)	O	R/W	4	Only available if Persistent IP is supported by the device.
0x06D0	Reserved	-	-	-	
0x06DC	Persistent Subnet Mask (Network interface #1)	O	R/W	4	Only available if Persistent IP is supported by the device.
0x06E0	Reserved	-	-	-	
0x06EC	Persistent Default Gateway (Network interface #1)	O	R/W	4	Only available if Persistent IP is supported by the device.
0x06F0	Link Speed (Network interface #1)	O	R	4	32-bit value indicating current Ethernet link speed in Mbits per second on the second interface.
0x06F4	Reserved	-	-	-	

Address	Name	Support	Type	Length (bytes)	Description
0x0700	Device MAC Address – High (Network interface #2)	O	R	4	The two most-significant bytes of this area are reserved and will return 0. Upper 2 bytes of the MAC address are stored in the lower 2 significant bytes of this register.
0x0704	Device MAC Address – Low (Network interface #2)	O	R	4	Lower 4 bytes of the MAC address
0x0708	Network Interface Capability (Network interface #2)	O	R	4	Supported IP Configuration and PAUSE schemes. Bits can be OR-ed. All other bits are reserved and set to 0. DHCP and LLA bits must be on.
0x070C	Network Interface Configuration (Network interface #2)	O	R/W	4	Activated IP Configuration and PAUSE schemes. Bits can be OR-ed. LLA is always activated and is read-only. PAUSE settings might be hard-coded.
0x0710	Reserved	-	-	-	
0x071C	Current IP Address (Network interface #2)	O	R	4	Current IP address of this device on its third interface.
0x0720	Reserved	-	-	-	
0x072C	Current Subnet Mask (Network interface #2)	O	R	4	Current subnet mask used by this device on its third interface.
0x0730	Reserved	-	-	-	
0x073C	Current Default Gateway (Network interface #2)	O	R	4	Current default gateway used by this device on its third interface.
0x0740	Reserved	-	-	-	
0x074C	Persistent IP Address (Network interface #2)	O	R/W	4	Only available if Persistent IP is supported by the device.
0x0750	Reserved	-	-	-	
0x075C	Persistent Subnet Mask (Network interface #2)	O	R/W	4	Only available if Persistent IP is supported by the device.
0x0760	Reserved	-	-	-	
0x076C	Persistent Default Gateway (Network interface #2)	O	R/W	4	Only available if Persistent IP is supported by the device.
0x0770	Link Speed (Network interface #2)	O	R	4	32-bit value indicating current Ethernet link speed in Mbits per second on the third interface.
0x0774	Reserved	-	-	-	
0x0780	Device MAC Address – High (Network interface #3)	O	R	4	The two most-significant bytes of this area are reserved and will return 0. Upper 2 bytes of the MAC address are stored in the lower 2 significant bytes of this register.
0x0784	Device MAC Address – Low (Network interface #3)	O	R	4	Lower 4 bytes of the MAC address
0x0788	Network Interface Capability (Network interface #3)	O	R	4	Supported IP Configuration and PAUSE schemes. Bits can be OR-ed. All other bits are reserved and set to 0. DHCP and LLA bits must be on.

Address	Name	Support	Type	Length (bytes)	Description
0x078C	Network Interface Configuration (Network interface #3)	O	R/W	4	Activated IP Configuration and PAUSE schemes. Bits can be OR-ed. LLA is always activated and is read-only. PAUSE settings might be hard-coded.
0x0790	Reserved	-	-	-	
0x079C	Current IP Address (Network interface #3)	O	R	4	Current IP address of this device on its fourth interface.
0x07A0	Reserved	-	-	-	
0x07AC	Current Subnet Mask (Network interface #3)	O	R	4	Current subnet mask used by this device on its fourth interface.
0x07B0	Reserved	-	-	-	
0x07BC	Current Default Gateway (Network interface #3)	O	R	4	Current default gateway used by this device on its fourth interface.
0x07C0	Reserved	-	-	-	
0x07CC	Persistent IP Address (Network interface #3)	O	R/W	4	Only available if Persistent IP is supported by the device.
0x07D0	Reserved	-	-	-	
0x07DC	Persistent Subnet Mask (Network interface #3)	O	R/W	4	Only available if Persistent IP is supported by the device.
0x07E0	Reserved	-	-	-	
0x07EC	Persistent Default Gateway (Network interface #3)	O	R/W	4	Only available if Persistent IP is supported by the device.
0x07F0	Link Speed (Network interface #3)	O	R	4	32-bit value indicating current Ethernet link speed in Mbits per second on the fourth interface.
0x07F4	Reserved	-	-	-	
0x0900	Number of Message Channels	M	R	4	Indicates the number of message channels supported by this device. It can take two values: 0 or 1.
0x0904	Number of Stream Channels	M	R	4	Indicates the number of stream channels supported by this device. It can take any value from 0 to 512.
0x0908	Number of Action Signals	O	R	4	Indicates the number of separate action signals supported by this device. It can take any value ranging from 0 to 128.
0x090C	Action Device Key	O	W	4	Device key to check the validity of action commands.
0x0910	Number of Active Links	M	R	4	Indicates the number of physical links that are currently active.
0x0914	Reserved	-	-	-	
0x092C	GVSP Capability	M[1]	R	4	Indicates the capabilities of the stream channels. It lists which of the non-mandatory stream channel features are supported.

[1] Mandatory only for transmitter, receiver and transceiver devices. Not applicable for peripherals.

Address	Name	Support	Type	Length (bytes)	Description
0x0930	Message Channel Capability	M	R	4	Indicates the capabilities of the message channel. It lists which of the non-mandatory message channel features are supported.
0x0934	GVCP Capability	M	R	4	This is a capability register indicating which one of the non-mandatory GVCP features are supported by this device.
0x0938	Heartbeat Timeout	M	R/W	4	In msec, default is 3000 msec. Internally, the heartbeat is rounded according to the clock used for heartbeat. The heartbeat timeout shall have a minimum precision of 100 ms. The minimal value is 500 ms.
0x093C	Timestamp Tick Frequency – High	O	R	4	64-bit value indicating the number of timestamp clock tick in 1 second. This register holds the most significant bytes. Timestamp tick frequency is 0 if timestamp is not supported.
0x0940	Timestamp Tick Frequency – Low	O	R	4	64-bit value indicating the number of timestamp clock tick in 1 second. This register holds the least significant bytes. Timestamp tick frequency is 0 if timestamp is not supported.
0x0944	Timestamp Control	O	W	4	Used to latch the current timestamp value. No need to clear to 0.
0x0948	Timestamp Value – High	O	R	4	Latched value of the timestamp (most significant bytes)
0x094C	Timestamp Value – Low	O	R	4	Latched value of the timestamp (least significant bytes)
0x0950	Discovery ACK Delay	O	R/(W)	4	Maximum randomized delay in ms to acknowledge a DISCOVERY_CMD.
0x0954	GVCP Configuration	O	R/W	4	Configuration of GVCP optional features
0x0958	Pending Timeout	M	R	4	Pending Timeout to report the longest GVCP command execution time before issuing a PENDING_ACK. If PENDING_ACK is not supported, then this is the worst-case execution time before command completion.
0x095C	Control Switchover Key	O	W	4	Key to authenticate primary application switchover requests.
0x0960	GVSP Configuration	O	R/W	4	Configuration of the optional GVSP features.
0x0964	Physical Link Configuration Capability	M	R	4	Indicates the physical link configuration supported by this device.
0x0968	Physical Link Configuration	M	R/(W)	4	Indicates the currently active physical link configuration
0x096C	IEEE 1588 Status	O	R	4	Reports the state of the IEEE 1588 clock
0x0970	Scheduled Action Command Queue Size	O	R	4	Indicates the number of Scheduled Action Commands that can be queued (size of the queue).
0x0974	Reserved	-	-	-	
0x0A00	Control Channel Privilege (CCP)	M	R/W	4	Control Channel Privilege register.
0x0A04	Primary Application Port	O	R	4	UDP source port of the control channel of the primary application.

Address	Name	Support	Type	Length (bytes)	Description
0x0A08	Reserved	-	-	-	
0x0A14	Primary Application IP Address	O	R	4	Source IP address of the control channel of the primary application.
0x0A18	Reserved	-	-	-	
0x0B00	Message Channel Port (MCP)	O	R/W	4	Message Channel Port register.
0x0B04	Reserved	-	-	-	
0x0B10	Message Channel Destination Address (MCDA)	O	R/W	4	Message Channel Destination Address register.
0x0B14	Message Channel Transmission Timeout (MCTT)	O	R/W	4	Message Channel Transmission Timeout in ms.
0x0B18	Message Channel Retry Count (MCRC)	O	R/W	4	Message Channel Retry Count.
0x0B1C	Message Channel Source Port (MCSP)	O	R	4	Message Channel Source Port.
0x0B20	Reserved	-	-	-	
0x0D00	Stream Channel Port 0 (SCP0)	M[1]	R/W	4	First Stream Channel Port register.
0x0D04	Stream Channel Packet Size 0 (SCPS0)	M[1]	R/W	4	First Stream Channel Packet Size register.
0x0D08	Stream Channel Packet Delay 0 (SCPD0)	M[2]	R/W	4	First Stream Channel Packet Delay register.
0x0D0C	Reserved	-	-	-	
0x0D18	Stream Channel Destination Address 0 (SCDA0)	M[1]	R/W	4	First Stream Channel Destination Address register.
0x0D1C	Stream Channel Source Port 0 (SCSP0)	O	R	4	First Stream Channel Source Port register.
0x0D20	Stream Channel Capability 0 (SCC0)	O	R	4	First Stream Channel Capability register.
0x0D24	Stream Channel Configuration 0 (SCCFG0)	O	R/W	4	First Stream Channel Configuration register.
0x0D28	Stream Channel Zone (SCZ0)	O	R	4	First Stream Channel Zone register (multi-zone image payload type only).
0x0D2C	Stream Channel Zone Direction (SCZD0)	O	R	4	First Stream Channel Zone Direction register (multi-zone image payload type only).
0x0D30	Reserved	-	-	-	
0x0D40	Stream Channel Port 1 (SCP1)	O	R/W	4	Second stream channel, if supported.
0x0D44	Stream Channel Packet Size 1 (SCPS1)	O	R/W	4	Second stream channel, if supported.
0x0D48	Stream Channel Packet Delay 1 (SCPD1)	O	R/W	4	Second stream channel, if supported.

[1] Mandatory only for transmitter, receiver and transceiver devices. Not applicable for peripherals.
[2] Mandatory only for transmitter devices. Not applicable for others.

Address	Name	Support	Type	Length (bytes)	Description
0x0D4C	Reserved	-	-	-	
0x0D58	Stream Channel Destination Address 1 (SCDA1)	O	R/W	4	Second stream channel, if supported.
0x0D5C	Stream Channel Source Port 1 (SCSP1)	O	R	4	Second stream channel, if supported.
0x0D60	Stream Channel Capability 1 (SCC1)	O	R	4	Second stream channel, if supported.
0x0D64	Stream Channel Configuration 1 (SCCFG1)	O	R/W	4	Second stream channel, if supported.
0x0D68	Stream Channel Zone (SCZ1)	O	R	4	Second stream channel, if supported.
0x0D6C	Stream Channel Zone Direction (SCZD1)	O	R	4	Second stream channel, if supported.
0x0D70	Reserved	-	-	-	
0x0D80	*Other stream channels registers*	O	R/W	-	Each stream channel is allocated a section of 64 bytes (0x40). Only supported channels are available.
0x8CC0	Stream Channel Port 511 (SCP511)	O	R/W	4	512th stream channel, if supported.
0x8CC4	Stream Channel Packet Size 511 (SCPS511)	O	R/W	4	512th stream channel, if supported.
0x8CC8	Stream Channel Packet Delay 511 (SCPD511)	O	R/W	4	512th stream channel, if supported.
0x8CCC	Reserved	-	-	-	
0x8CD8	Stream Channel Destination Address 511 (SCDA511)	O	R/W	4	512th stream channel, if supported.
0x8CDC	Stream Channel Source Port 511 (SCSP511)	O	R	4	512th stream channel, if supported.
0x8CE0	Stream Channel Capability 511 (SCC511)	O	R	4	512th stream channel, if supported.
0x8CE4	Stream Channel Configuration 511 (SCCFG511)	O	R/W	4	512th stream channel, if supported.
0x8CE8	Stream Channel Zone (SCZ511)	O	R	4	512th stream channel, if supported.
0x8CEC	Stream Channel Zone Direction (SCZD511)	O	R	4	512th stream channel, if supported.
0x8CF0	Reserved	-	-	-	
0x9000	Manifest Table	O	R	512	Manifest Table providing information to retrieve XML documents stored in the device.
0x9200	Reserved	-	-	-	
0x9800	Action Group Key 0	O	R/W	4	First action signal group key
0x9804	Action Group Mask 0	O	R/W	4	First action signal group mask
0x9808	Reserved	-	-	-	
0x9810	Action Group Key 1	O	R/W	4	Second action signal group key

Address	Name	Support	Type	Length (bytes)	Description
0x9814	Action Group Mask 1	O	R/W	4	Second action signal group mask
0x9818	Reserved	-	-	-	
0x9820	*Other action signals registers*	O	R/W	-	Each action signal is allocated a section of 16 bytes (0x10). Only supported action signals are available.
0x9FF0	Action Group Key 127	O	R/W	4	128th action signal group key
0x9FF4	Action Group Mask 127	O	R/W	4	128th action signal group mask
0x9FF8	Reserved	-	-	-	
0xA000	Start of manufacturer-specific register space	-	-	-	Start of manufacturer-specific registers. These are not covered by the specification and should be reported through the XML device description file.

GEV V2.0 020000

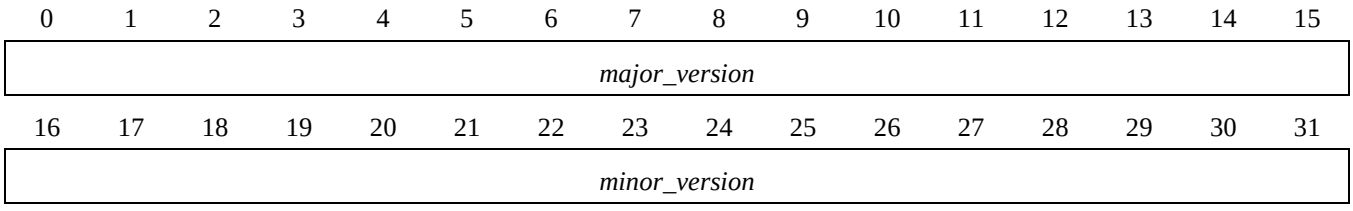
28.1 Version Register

This register indicates the version of the GigE Vision specification implemented by this device. Version 2.0 of this specification shall return 0x00020000.

The Version register can be used by the application to validate the device is compliant with the specified version of the GigE Vision specification.

[R-429cd] A device MUST implement the Version register.

Address	0x0000
Length	4 bytes
Access type	Read
Factory Default	0x00010002



Bits	Name	Description
0 – 15	*major_version*	This field represents the major version of the specification. For instance, GigE Vision version **2**.0 would have the major version set to 2.
16 – 31	*minor_version*	This field represents the minor version of the specification. For instance, GigE Vision version 2.**0** would have the minor version set to 0.

28.2 Device Mode Register

This register indicates the character set used by the various strings present in the bootstrap registers and other device-specific information, such as the link configuration and the device class.

[R-430cd] A device MUST implement the Device Mode register.

Address	0x0004
Length	4 bytes
Access type	Read
Factory Default	Device-specific

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
E	*DC*			*CLC*				-	-	-	-	-	-	-	-
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
-	-	-	-	-	-	-	-	*character_set_index*							

Bits	Name	Description
0	*endianess (E)*	Endianess might be used to interpret multi-byte data for READMEM and WRITEMEM commands. This represents the endianess of the device bootstrap registers. 0: reserved, do not use 1: big-endian device GigE Vision 2.0 devices MUST always report big-endian and set this bit to 1.
1 – 3	*device_class (DC)*	This field represents the class of the device. It must take one of the following values 0: Transmitter 1: Receiver 2: Transceiver 3: Peripheral other: reserved
4 – 7	*current_link_configuration (CLC)*	Indicate the current Physical Link configuration of the device 0: Single Link Configuration 1: Multiple Links Configuration 2: Static LAG Configuration 3: Dynamic LAG Configuration Note: when Multiple Links Configuration and LAG are used concurrently, the device shall report the configuration with the highest value.
8 – 23	reserved	Always 0. Ignore when reading.

24 – 31	*character_set_index*	This register represents the character set. It must take one of the following values 0: reserved 1: UTF-8 2: ASCII (preferred for compatibility with the GenICam standard) other: reserved

28.3 Device MAC Address Registers

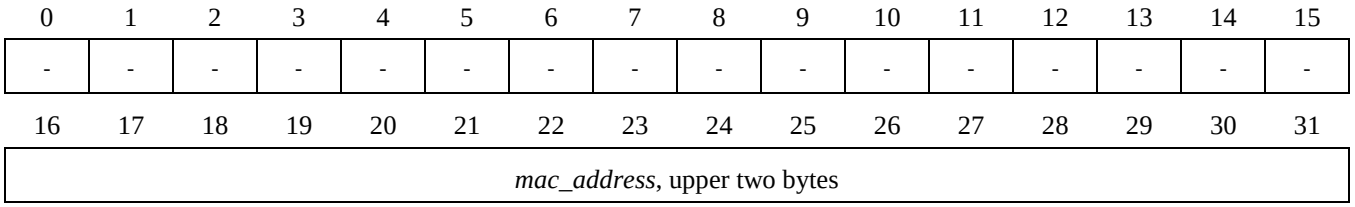
These registers store the MAC address of the given network interface. If link aggregation is used, this is the MAC address used by the link aggregation group.

[R-431cd] A device MUST implement the Device MAC Address registers on all of its supported network interfaces. Both registers (High Part and Low Part) MUST be available. Devices must return GEV_STATUS_INVALID_ADDRESS for unsupported network interfaces. [R27-10cd]

An application needs to access the high part before accessing the low part.

28.3.1 High Part

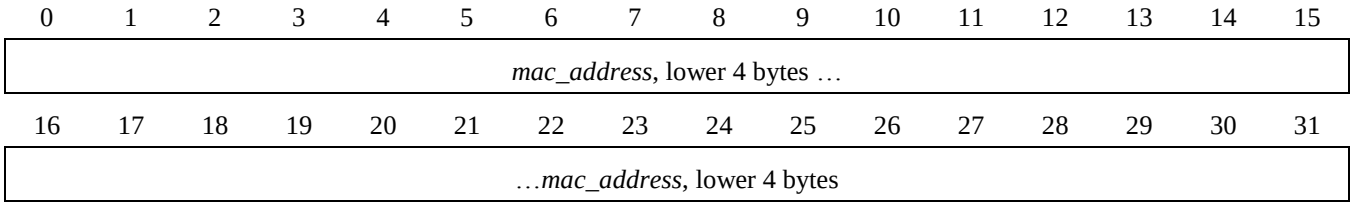
Address	0x0008 (interface #0) 0x0680 (interface #1) *[optional]* 0x0700 (interface #2) *[optional]* 0x0780 (interface #3) *[optional]*
Length	4 bytes
Access type	Read
Factory Default	Device-specific



Bits	Name	Description
0 – 15	reserved	Always 0. Ignore when reading.
16 – 31	*mac_address*	Hold the upper two bytes of the Device MAC address

28.3.2 Low Part

Address	0x000C (interface #0) 0x0684 (interface #1) *[optional]* 0x0704 (interface #2) *[optional]* 0x0784 (interface #3) *[optional]*
Length	4 bytes
Access type	Read
Factory Default	Device-specific



Bits	Name	Description
0 – 31	*mac_address*	Hold the lower four bytes of the Device MAC address

28.4 Network Interface Capability Registers

These registers indicate the network and IP configuration schemes supported on the given network interface. Multiple schemes can be supported simultaneously. If link aggregation is used, this is the IP configuration scheme supported by the link aggregation group.

[R-432cd] The device MUST implement the Network Interface Capability register on all of its supported network interfaces. Devices must return GEV_STATUS_INVALID_ADDRESS for unsupported network interfaces. [R27-11cd]

Address	0x0010 (interface #0) 0x0688 (interface #1) *[optional]* 0x0708 (interface #2) *[optional]* 0x0788 (interface #3) *[optional]*
Length	4 bytes
Access type	Read
Factory Default	Device-specific

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
PR	PG	-	-	-	-	-	-	-	-	-	-	-	-	-	-

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
-	-	-	-	-	-	-	-	-	-	-	-	-	*L*	*D*	*P*

Bits	Name	Description
0	*PAUSE_reception (PR)*	Set to 1 if the network interface can handle PAUSE frames.
1	*PAUSE_generation (PG)*	Set to 1 if the network interface can generate PAUSE frames.
2 – 28	reserved	Always 0. Ignore when reading.
29	*LLA (L)*	Link-local address is supported. Always 1
30	*DHCP (D)*	DHCP is supported. Always 1
31	*Persistent_IP (P)*	1 if Persistent IP is supported, 0 otherwise.

28.5 Network Interface Configuration Registers

These registers indicate which network and IP configurations schemes are currently activated on the given network interface. If link aggregation is used, this is the current IP configuration scheme used by the link aggregation group.

[R-433cd] The device MUST implement the Network Interface Configuration register on all of its supported network interfaces. Devices must return GEV_STATUS_INVALID_ADDRESS for unsupported network interfaces. [R27-12cd]

Address	0x0014 (interface #0) 0x068C (interface #1) *[optional]* 0x070C (interface #2) *[optional]* 0x078C (interface #3) *[optional]*
Length	4 bytes
Access type	Read/Write
Factory Default	see description

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
PR	*PG*	-	-	-	-	-	-	-	-	-	-	-	-	-	-

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
-	-	-	-	-	-	-	-	-	-	-	-	-	*L*	*D*	*P*

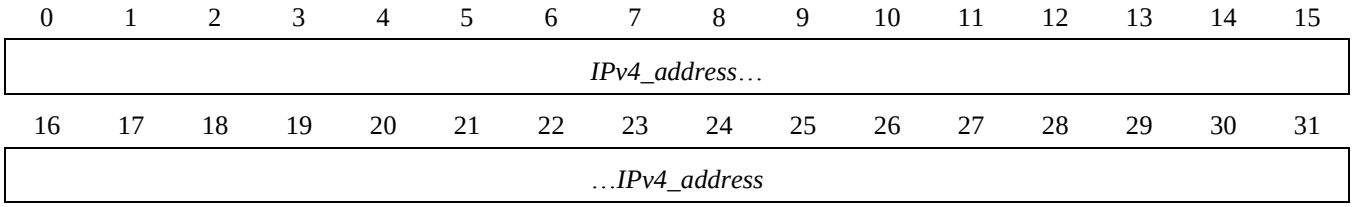
Bits	Name	Description
0	*PAUSE_reception (PR)*	Set to 1 to enable the reception of PAUSE frames on this network interface. Factory default is device-specific and might be hard-coded if not configurable. New settings to take effect on next link negotiation. This setting must persist across device reset.
1	*PAUSE_generation (PG)*	Set to 1 to enable the generation of PAUSE frames from this network interface. Factory default is device-specific and might be hard-coded if not configurable. New settings to take effect on next link negotiation. This setting must persist across device reset.
2 – 28	reserved	Always 0. Ignore when reading.
29	*LLA (L)*	Link-local address is activated. Always 1.
30	*DHCP (D)*	DHCP is activated on this interface. Factory default is 1.
31	*Persistent_IP (P)*	Persistent IP is activated on this interface. Factory default is 0.

28.6 Current IP Address Registers

These registers report the IP address for the given network interface once it has been configured. If link aggregation is used, this is the IP address used by the link aggregation group.

[R-434cd] The device MUST support the Current IP Address register on all of its supported network interfaces. Devices must return GEV_STATUS_INVALID_ADDRESS for unsupported interfaces. [R27-13cd]

Address	0x0024 (interface #0) 0x069C (interface #1) *[optional]* 0x071C (interface #2) *[optional]* 0x079C (interface #3) *[optional]*
Length	4 bytes
Access type	Read
Factory Default	0x00000000



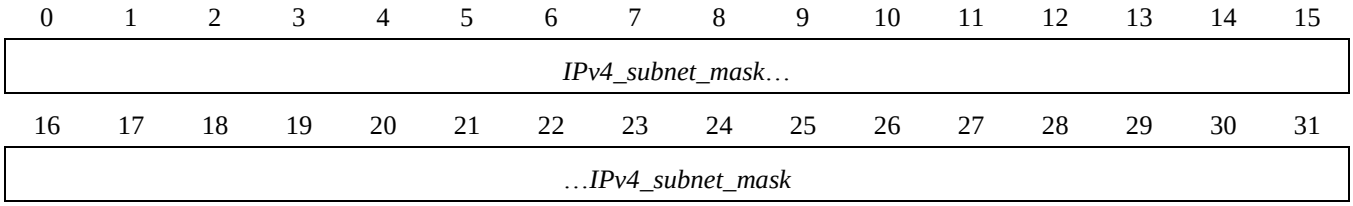
Bits	Name	Description
0 – 31	*IPv4_address*	IP address for this network interface

28.7 Current Subnet Mask Registers

These registers provide the subnet mask of the given interface. If link aggregation is used, this is the subnet mask used by the link aggregation group.

[R-435cd] The device MUST support the Current Subnet Mask register on all of its supported network interfaces. Devices must return GEV_STATUS_INVALID_ADDRESS for unsupported network interfaces. [R27-14cd]

Address	0x0034 (interface #0) 0x06AC (interface #1) *[optional]* 0x072C (interface #2) *[optional]* 0x07AC (interface #3) *[optional]*
Length	4 bytes
Access type	Read
Factory Default	0x00000000



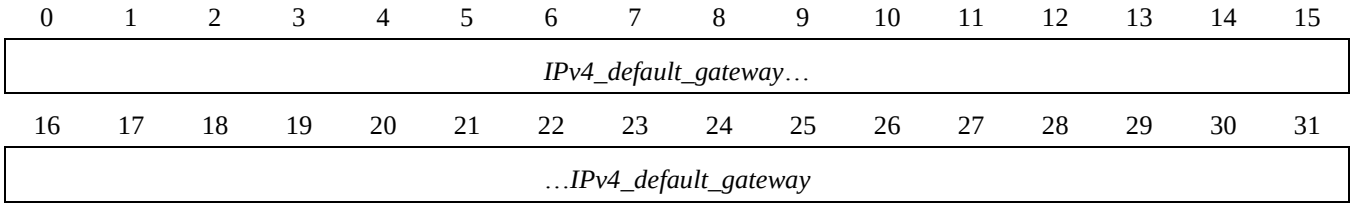
Bits	Name	Description
0 – 31	*IPv4_subnet_mask*	Subnet mask for this network interface

28.8 Current Default Gateway Registers

These registers indicate the default gateway IP address to be used on the given network interface. If link aggregation is used, this is the default gateway used by the link aggregation group.

[R-436cd] The device MUST support the Current Default Gateway register on all of its supported network interfaces. Devices must return GEV_STATUS_INVALID_ADDRESS for unsupported network interfaces. [R27-15cd]

Address	0x0044 (interface #0) 0x06BC (interface #1) *[optional]* 0x073C (interface #2) *[optional]* 0x07BC (interface #3) *[optional]*
Length	4 bytes
Access type	Read
Factory Default	Device-specific



Bits	Name	Description
0 – 31	*IPv4_default_gateway*	Default gateway for this network interface

28.9 Manufacturer Name Register

This register stores a string containing the manufacturer name. This string uses the character set indicated in the Device Mode register.

[R-437cd] A device MUST implement the Manufacturer Name register.

Address	0x0048
Length	32 bytes
Access type	Read
Factory Default	Device-specific

Bits	Name	Description
-	*manufacturer_name*	String indicating the manufacturer name. Returned in DISCOVERY_ACK and in the TXT record of DNS-SD.

28.10 Model Name Register

This register stores a string containing the device model name. This string uses the character set indicated in the Device Mode register.

[R-438cd] A device MUST implement the Model Name register.

Address	0x0068
Length	32 bytes
Access type	Read
Factory Default	Device-specific

Bits	Name	Description
-	*model_name*	String indicating the device model. Returned in DISCOVERY_ACK and in the TXT record of DNS-SD.

28.11 Device Version Register

This register stores a string containing the version of the device. This string uses the character set indicated in the Device Mode register. The XML device description file should also provide this information to ensure the device matches the description file.

[R-439cd] A device MUST implement the Device Version register.

Address	0x0088
Length	32 bytes
Access type	Read
Factory Default	Device-specific

Bits	Name	Description
-	*device_version*	String indicating the version of the device. Returned in DISCOVERY_ACK and in the TXT record of DNS-SD.

28.12 Manufacturer Info Register

This register stores a string containing additional manufacturer-specific information about the device. This string uses the character set indicated in the Device Mode register.

[R-440cd] A device MUST implement the Manufacturer Info register.

Address	0x00A8
Length	48 bytes
Access type	Read
Factory Default	Device-specific

Bits	Name	Description
-	*manufacturer_specific_information*	String providing additional information about this device. Returned in DISCOVERY_ACK and in the TXT record of DNS-SD.

28.13 Serial Number Register

This register is optional. It can store a string containing the serial number of the device. This string uses the character set indicated in the Device Mode register.

An application can check bit 1 of the GVCP Capability register at address 0x0934 to determine if the serial number register is supported by the device.

[O-441cd] A device SHOULD implement the Serial Number register.

Address	0x00D8 *[optional]*
Length	16 bytes
Access type	Read
Factory Default	Device-specific

Bits	Name	Description
-	*serial_number*	String providing the serial number of this device. If supported, returned in DISCOVERY_ACK and in the TXT record of DNS-SD.

28.14 User-defined Name Register

This register is optional. It can store a user-programmable string providing the name of the device. This string uses the character set indicated in the Device Mode register.

An application can check bit 0 of the GVCP Capability register at address 0x0934 to determine if the user-defined name register is supported by the device.

[O-442cd] A device SHOULD implement the User-defined Name register.

[CR-443cd] When the User-defined Name register is supported, the device MUST provide persistent storage to memorize the user-defined name. This name shall remain readable across power-up cycles. The state of this register is stored in non-volatile memory as soon as the User-defined Name register is set by an application. [CR27-18cd]

Address	0x00E8 *[optional]*
Length	16 bytes
Access type	Read/Write
Factory Default	Device-specific

Bits	Name	Description
-	*user_defined_name*	String providing the device name. If supported, returned in DISCOVERY_ACK and in the TXT record of DNS-SD.

28.15 First URL Register

This register stores the first URL to the XML device description file. This string uses the character set indicated in the Device Mode register. The first URL is used as the first choice by the application to retrieve the XML device description file, except if a Manifest Table is available.

[R-444cd] A device MUST implement the First URL register, even if a Manifest Table is present.

Address	0x0200
Length	512 bytes
Access type	Read
Factory Default	Device-specific

Bits	Name	Description
-	*first_URL*	String providing the first URL to the XML device description file.

28.16 Second URL Register

This register stores the second URL to the XML device description file. This string uses the character set indicated in the Device Mode register. This URL is an alternative if the application was unsuccessful to retrieve the device description file using the first URL.

[R-445cd] A device MUST implement the Second URL register, even if a Manifest Table is present.

Address	0x0400
Length	512 bytes
Access type	Read
Factory Default	Device-specific

Bits	Name	Description
-	*second_URL*	String providing the second URL to the XML device description file.

28.17 Number of Network Interfaces Register

This register indicates the number of network interfaces supported by this device. This is generally equivalent to the number of Ethernet connectors on the device, except when Link Aggregation is used. In this case, the physical interfaces regrouped by the Aggregator are counted as one virtual interface.

[R-446cd] A device MUST implement the Number of Network Interfaces register.

A device need to support at least one network interfaces (the primary interface = interface #0). A device can support at most four network interfaces.

Note that interface #0 is the only one supporting GVCP. Additional network interfaces only supports stream channels in order to increase available bandwidth out of the device.

Address	0x0600
Length	4 bytes
Access type	Read
Factory Default	Device-specific

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
-	-	-	-	-	-	-	-	-	-	-	-	-	*number_of_interfaces*		

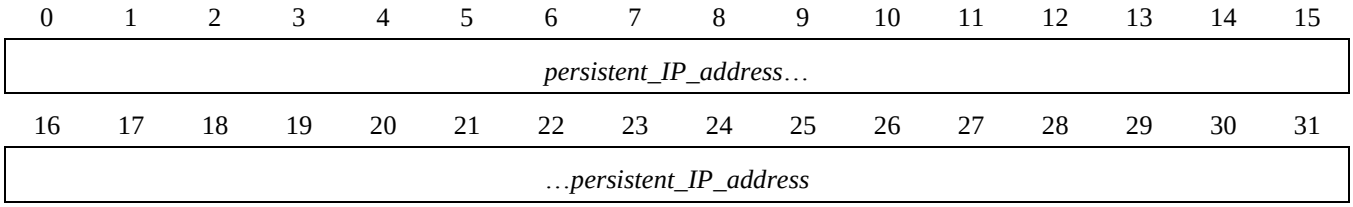
Bits	Name	Description
0 – 28	reserved	Always 0. Ignore when reading.
29 – 31	*number_of_interfaces*	Indicates the number of network interfaces. This field takes a value from 1 to 4. All other values are invalid.

28.18 Persistent IP Address Registers

These optional registers indicate the persistent IP address for the given network interface. They are only used when the device boots with the Persistent IP configuration scheme. If link aggregation is used, this is the persistent IP address used by the link aggregation group.

[CR-447cd] When Persistent IP is supported, the device MUST implement Persistent IP Address register on all of its supported network interfaces. Devices must return GEV_STATUS_INVALID_ADDRESS for unsupported network interfaces. [CR27-21cd]

Address	0x064C (interface #0) *[optional]* 0x06CC (interface #1) *[optional]* 0x074C (interface #2) *[optional]* 0x07CC (interface #3) *[optional]*
Length	4 bytes
Access type	Read/Write
Factory Default	Device-specific



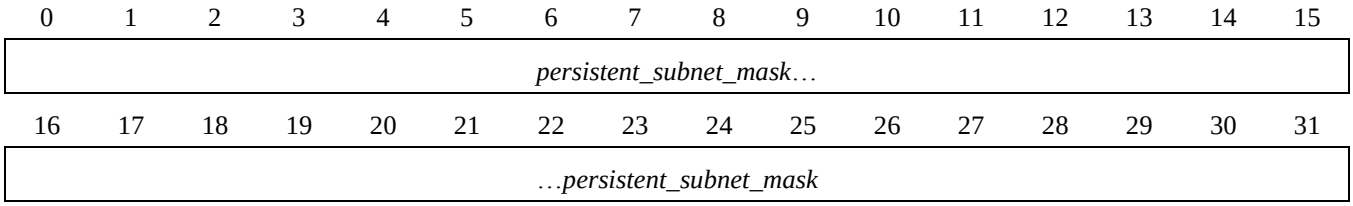
Bits	Name	Description
0 – 31	*persistent_IP_address*	IPv4 persistent IP address for this network interface

28.19 Persistent Subnet Mask Registers

These optional registers indicate the persistent subnet mask associated with the persistent IP address on the given network interface. They are only used when the device boots with the Persistent IP configuration scheme. If link aggregation is used, this is the persistent subnet mask used by the link aggregation group.

[CR-448cd] When Persistent IP is supported, the device MUST implement Persistent Subnet Mask register on all of its supported network interfaces. Devices must return GEV_STATUS_INVALID_ADDRESS for unsupported network interfaces. [CR27-22cd]

Address	0x065C (interface #0) *[optional]* 0x06DC (interface #1) *[optional]* 0x075C (interface #2) *[optional]* 0x07DC (interface #3) *[optional]*
Length	4 bytes
Access type	Read/Write
Factory Default	Device-specific



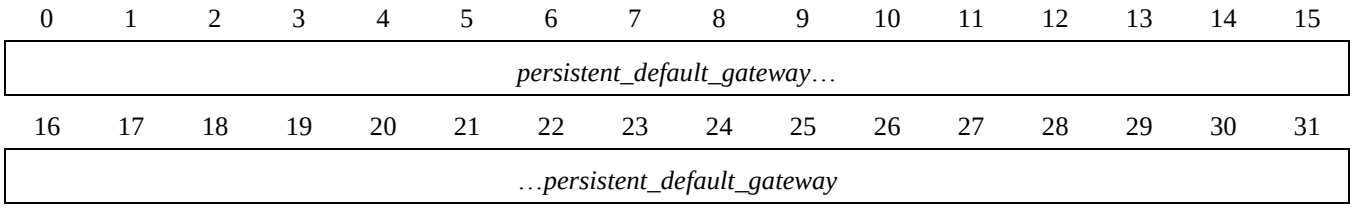
Bits	Name	Description
0 – 31	*persistent_subnet_mask*	IPv4 persistent subnet mask for this network interface

28.20 Persistent Default Gateway Registers

These optional registers indicate the persistent default gateway for the given network interface. They are only used when the device boots with the Persistent IP configuration scheme. If link aggregation is used, this is the persistent default gateway used by the link aggregation group.

[CR-449cd] When Persistent IP is supported, the device MUST implement Persistent Default Gateway register on all of its supported network interfaces. Devices must return GEV_STATUS_INVALID_ADDRESS for unsupported network interfaces. [CR27-23cd]

Address	0x066C (interface #0) *[optional]* 0x06EC (interface #1) *[optional]* 0x076C (interface #2) *[optional]* 0x07EC (interface #3) *[optional]*
Length	4 bytes
Access type	Read/Write
Factory Default	Device-specific



Bits	Name	Description
0 – 31	*persistent_default_gateway*	IPv4 persistent default gateway for this network interface

28.21 Link Speed Registers

These optional registers provide the Ethernet link speed (in Mbits per second) for the given network interface. This can be used to compute the transmission speed. If link aggregation is used, this is aggregated link speed of all links participating in the aggregator.

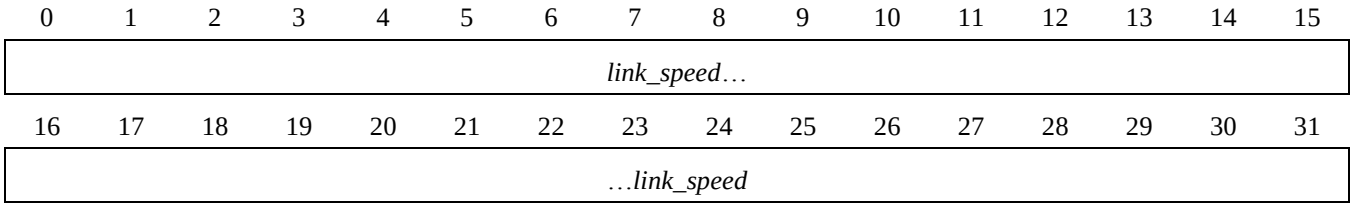
An application can check bit 3 of the GVCP Capability register at address 0x0934 to determine if the link speed registers are supported by the device.

[CR-450cd] When Link Speed registers are supported, the device MUST implement a Link Speed register on all of its supported network interfaces. Devices must return GEV_STATUS_INVALID_ADDRESS for unsupported network interfaces. [CR27-38cd]

For instance, if a device only implements a single network interface, then it shall only support one link speed registers assigned to network interface #0. A link speed of 1 gigabit per second is equal to 1000 Mbits per second while a link speed of 10 GigE is equal to 10 000 Mbits per second.

A capability bit in the GVCP Capability register (bit 3 at address 0x0934) indicates if these registers are supported.

Address	0x0670 (interface #0) *[optional]* 0x06F0 (interface #1) *[optional]* 0x0770 (interface #2) *[optional]* 0x07F0 (interface #3) *[optional]*
Length	4 bytes
Access type	Read
Factory Default	Device-specific



Bits	Name	Description
0 – 31	*link_speed*	Ethernet link speed value in Mbps. 0 if Link is down.

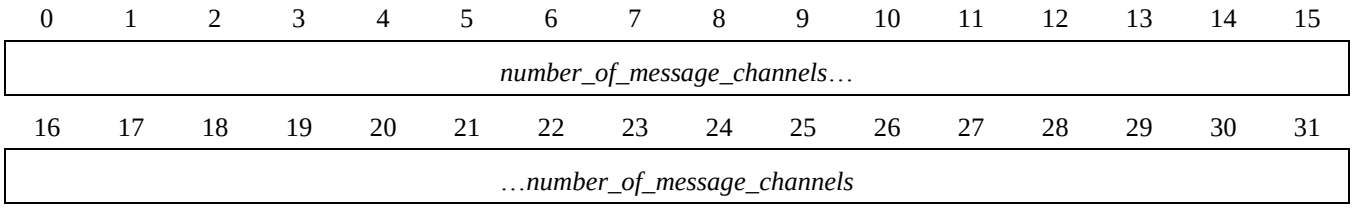
28.22 Number of Message Channels Register

This register reports the number of message channels supported by this device.

[R-451cd] A device MUST implement the Number of Message Channels register.

A device may support at most 1 message channel, but it is allowed to support no message channel.

Address	0x0900
Length	4 bytes
Access type	Read
Factory Default	Device-specific



Bits	Name	Description
0 – 31	*number_of_message_channels*	Number of message channels supported by this device. Can be 0 or 1.

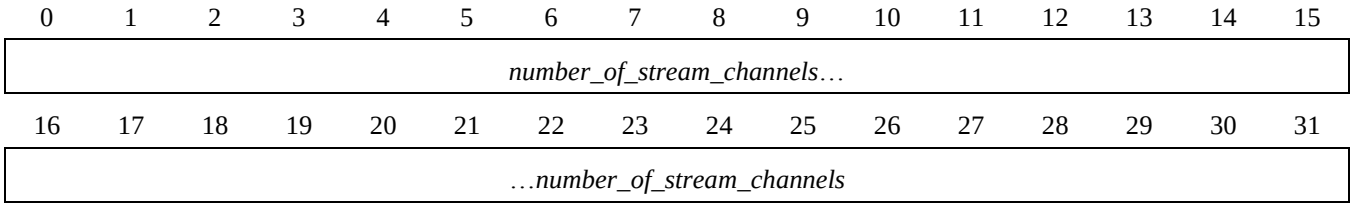
28.23 Number of Stream Channels Register

This register reports the number of stream channels supported by this device.

[R-452cd] A device MUST implement the Number of Stream Channels register.

A product can support from 0 up to 512 stream channels.

Address	0x0904
Length	4 bytes
Access type	Read
Factory Default	Device-specific



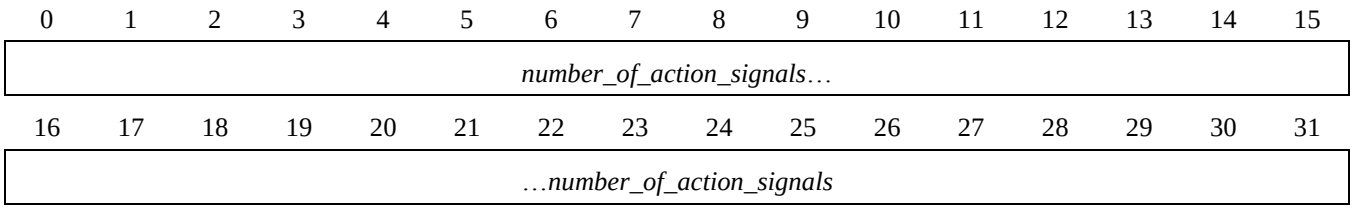
Bits	Name	Description
0 – 31	*number_of_stream_channels*	Number of stream channels supported by this device. A value from 0 to 512.

28.24 Number of Action Signals Register

This optional register reports the number of action signal supported by this device.

[CR-453cd] If a device supports Action commands, then it MUST implement the Number of Action Signals register.

Address	0x0908 *[optional]*
Length	4 bytes
Access type	Read
Factory Default	Device-specific



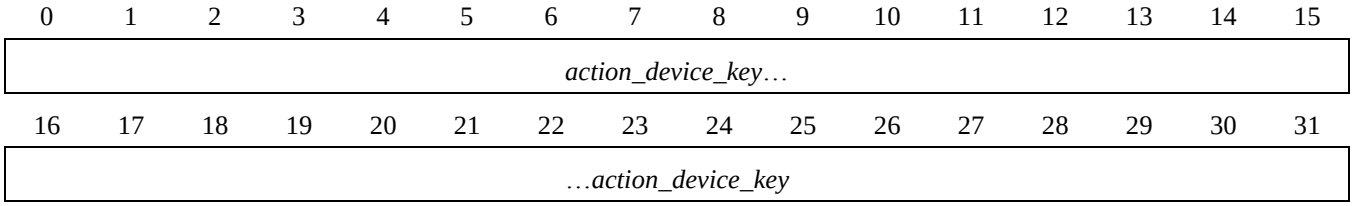
Bits	Name	Description
0 – 31	*number_of_action_signals*	Number of action signals supported by this device. A value from 0 to 128.

28.25 Action Device Key Register

This optional register provides the device key to check the validity of action commands. The action device key is **write-only** to hide the key if CCP is not in exclusive access mode. The intention is for the Primary Application to have absolute control. The Primary Application can send the key to a Secondary Application. This mechanism prevents a rogue application to have access to the ACTION_CMD mechanism.

[CR-454cd] If a device supports Action commands, then it MUST implement the Action Device Key register.

Address	0x090C *[optional]*
Length	4 bytes
Access type	Write-only
Factory Default	0



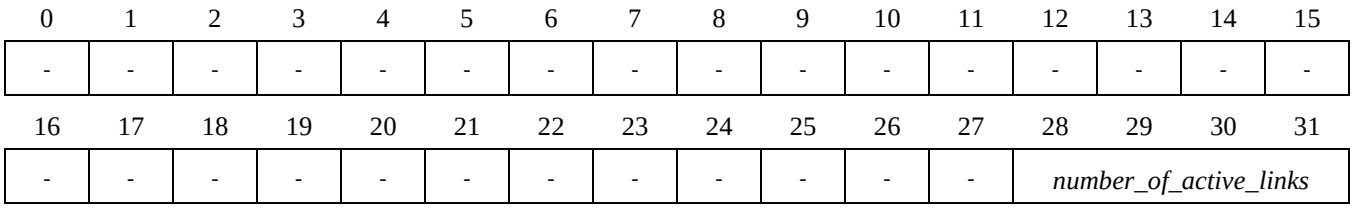
Bits	Name	Description
0 – 31	*action_device_key*	Device key to check the validity of action commands

28.26 Number of Active Links

This register indicates the current number of active links. A link is considered active as soon as it is connected to another Ethernet device. This happens after Ethernet link negotiation is completed.

[R-455cd] A device MUST implement the Number of Active Links register.

Address	0x0910
Length	4 bytes
Access type	Read
Factory Default	Device-specific



Bits	Name	Description
0 – 27	reserved	Always 0. Ignore when reading.
28 – 31	*number_of_active_links*	A value indicating the number of active links. This number must be lower or equal to the Number of Network Interfaces reported at address 0x0600. Single network interface device shall report a value of 1.

28.27 GVSP Capability Register

This register reports the optional GVSP stream channel features supported by this device.

[R-456cd] A transmitter, receiver or transceiver device MUST implement the GVSP Capability register.

Address	0x092C
Length	4 bytes
Access type	Read
Factory Default	Device-specific

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
SP	*LB*	-	-	-	-	-	-	-	-	-	-	-	-	-	-
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Bits	Name	Description
0	*SCSPx_supported (SP)*	Indicates the SCSPx registers (stream channel source port) are available for all supported stream channels.
1	*legacy_16bit_block_id_supported (LB)*	Indicates this GVSP transmitter or receiver can support 16-bit *block_id*. Note that GigE Vision 2.0 transmitters and receivers MUST support 64-bit *block_id64*. **Note**: When 16-bit *block_id* are used, then 24-bit *packet_id* must be used. When 64-bit *block_id64* are used, then 32-bit *packet_id32* must be used.
2 – 31	Reserved	Always 0. Ignore when reading.

28.28 Message Channel Capability Register

This register reports the optional message channel features supported by this device.

[R-457cd] A device MUST implement the Message Channel Capability register.

Address	0x0930
Length	4 bytes
Access type	Read
Factory Default	Device-specific

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
SP	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Bits	Name	Description
0	*MCSP_supported (SP)*	Indicates the MCSP register (message channel source port) is available for the message channel.
1 – 31	reserved	Always 0. Ignore when reading.

28.29 GVCP Capability Register

This register reports the optional GVCP commands and bootstrap registers supported by this device on its Control Channel. When supported, some of these features are enabled through the GVCP Configuration register (at address 0x0954).

[R-458cd] A device MUST implement the GVCP Capability register.

Address	0x0934
Length	4 bytes
Access type	Read
Factory Default	Device-specific

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
UN	*SN*	*HD*	*LS*	*CAP*	*MT*	*TD*	*DD*	*WD*	*ES*	*PAS*	*UA*	*PTP*	*ES2*	*SAC*	-
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
-	-	-	-	-	-	-	-	-	*A*	*PA*	*ED*	*E*	*PR*	*W*	*C*

Bits	Name	Description
0	*user_defined_name (UN)*	User-defined name register is supported.
1	*serial_number (SN)*	Serial number register is supported.
2	*heartbeat_disable (HD)*	Heartbeat can be disabled.
3	*link_speed_register (LS)*	Link Speed registers are supported.
4	*CCP_application_port_IP_register (CAP)*	CCP Application Port and IP address registers are supported.
5	*manifest_table (MT)*	Manifest Table is supported. When supported, the application must use the Manifest Table to retrieve the XML device description file.
6	*test_data (TD)*	Test packet is filled with data from the LFSR generator.
7	*discovery_ACK_delay (DD)*	Discovery ACK Delay register is supported.
8	*writable_discovery_ACK_delay (WD)*	When Discovery ACK Delay register is supported, this bit indicates that the application can write it. If this bit is 0, the register is read-only.
9	*extended_status_code_for_1_1 (ES)*	Support generation of extended status codes introduced in specification 1.1: GEV_STATUS_PACKET_RESEND, GEV_STATUS_PACKET_NOT_YET_AVAILABLE, GEV_STATUS_PACKET_AND_PREV_REMOVED_FROM_MEMORY and GEV_STATUS_PACKET_REMOVED_FROM_MEMORY.
10	*primary_application_switchover (PAS)*	Primary application switchover capability is supported.
11	*unconditional_ACTION (UA)*	Unconditional ACTION_CMD is supported (i.e. ACTION_CMD are processed when the primary control channel is closed).
12	*IEEE1588_support (PTP)*	Support for IEEE 1588 PTP
13	*extended_status_code_for_2_0 (ES2)*	Support generation of extended status codes introduces in specification 2.0: GEV_STATUS_PACKET_TEMPORARILY_UNAVAILABLE, GEV_STATUS_OVERFLOW, GEV_STATUS_NO_REF_TIME.
14	*scheduled_action_command (SAC)*	Scheduled Action Commands are supported. When this bit is set, this also indicates that the Scheduled Action Command Queue Size register is present.
15 – 24	reserved	Always 0. Ignore when reading.
25	*ACTION (A)*	ACTION_CMD and ACTION_ACK are supported.
26	*PENDING_ACK (PA)*	PENDING_ACK is supported.
27	*EVENTDATA (ED)*	EVENTDATA_CMD and EVENTDATA_ACK are supported.
28	*EVENT (E)*	EVENT_CMD and EVENT_ACK are supported.
29	*PACKETRESEND (PR)*	PACKETRESEND_CMD is supported.
30	*WRITEMEM (W)*	WRITEMEM_CMD and WRITEMEM_ACK are supported.
31	*concatenation (C)*	Multiple operations in a single message are supported.

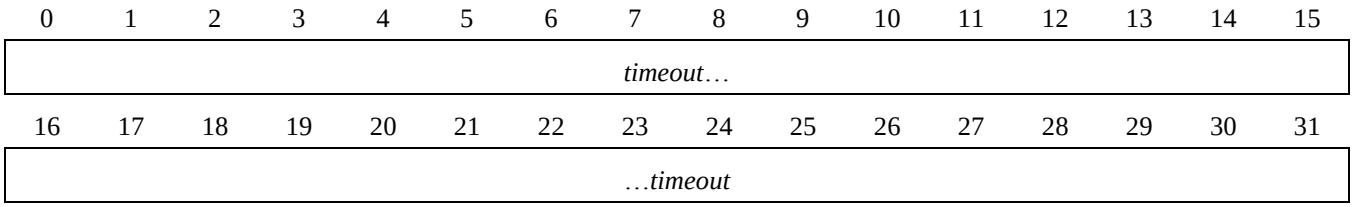
28.30 Heartbeat Timeout Register

This register indicates the current heartbeat timeout in milliseconds.

[R-459cd] A device MUST implement the Heartbeat Timeout register.

[O-460cd] A value smaller than 500 ms SHOULD be defaulted to 500 ms by the device. In this case, the Heartbeat Timeout register content is changed to reflect the actual value used by the device. [O27-24cd]

Address	0x0938
Length	4 bytes
Access type	Read/Write
Factory Default	3000 = 0x0BB8



Bits	Name	Description
0 – 31	*timeout*	Heartbeat timeout in milliseconds (minimum is 500 ms)

Note: Upon changing the heartbeat timeout register, it might take up to the amount of time previously specified in this register for the new value to take effect.

28.31 Timestamp Tick Frequency Registers

These optional registers indicate the number of timestamp tick during 1 second. This corresponds to the timestamp frequency in Hertz. For example, a 100 MHz clock would have a value of 100 000 000 = 0x05F5E100. They are combined to form a 64-bit value.

If IEEE 1588 is used, then the Timestamp Tick Frequency register must be 1 000 000 000 Hz (equivalent to 1 GHz). This indicates that the timestamp value is reported in units of 1 ns. But the precision of the timestamp register might be worst than 1 ns (i.e. the timestamp register does not necessarily takes all possible values when it increments, it can skip some value depending on the precision of the IEEE 1588 clock).

Note the timestamp counter is also used to compute stream channel inter-packet delay.

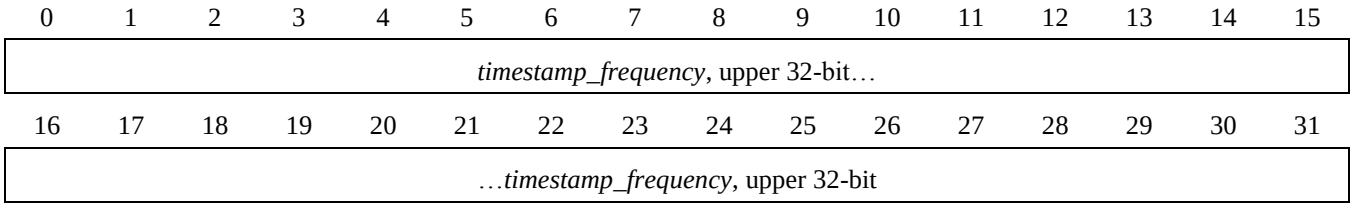
No timestamp is supported if both of these registers are equal to 0.

[CR-461cd] If a device supports a device timestamp counter, then it MUST implement the Timestamp Tick Frequency registers. Both registers MUST be available (High Part and Low Part). If its value is set to 0 or if this register is not available, then no timestamp counter is present. This means no mechanism is offered to control inter-packet delay. [O27-25cd]

An application needs to access the high part before accessing the low part.

28.31.1 High Part

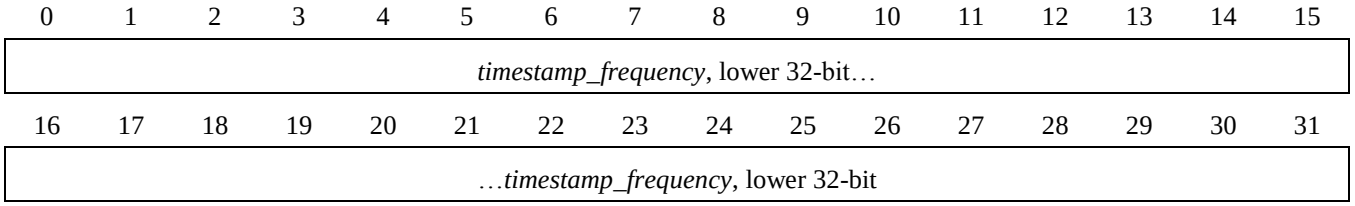
Address	0x093C *[optional]*
Length	4 bytes
Access type	Read
Factory Default	Device-specific



Bits	Name	Description
0 – 31	*timestamp_frequency*	Timestamp frequency, upper 32-bit

28.31.2 Low Part

Address	0x0940 *[optional]*
Length	4 bytes
Access type	Read
Factory Default	Device-specific



Bits	Name	Description
0 – 31	*timestamp_frequency*	Timestamp frequency, lower 32-bit

28.32 Timestamp Control Register

This optional register is used to control the timestamp counter.

[CR-462cd] If a device supports a device timestamp counter, then it MUST implement the Timestamp Control register.

[CR-463ca] If a timestamp counter exists, an application MUST not attempt to read this register. This register is write-only. [CR27-26ca]

Address	0x0944 *[optional]*
Length	4 bytes
Access type	Write-only
Factory Default	0

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
-	-	-	-	-	-	-	-	-	-	-	-	-	-	*L*	*R*

Bits	Name	Description
0 – 29	reserved	Always 0.
30	*latch (L)*	Latch current timestamp counter into Timestamp Value register at address 0x0948.
31	*reset (R)*	Reset timestamp 64-bit counter to 0. It is not possible to reset the timestamp when operating with an IEEE 1588 disciplined clock.

[CR-464cd] If a timestamp counter exists, when the application set both the Latch and Reset bit in the same access, then the device MUST first latch the timestamp before resetting it. [CR27-27cd]

Note: Writing a 1 into a bit causes the requested operation to execute. There is no need to write a 0 into the register afterwards.

28.33 Timestamp Value Registers

These optional registers report the latched value of the timestamp counter. It is necessary to latch the 64-bit timestamp value to guarantee its integrity when performing the two 32-bit read accesses to retrieve the higher and lower 32-bit portions.

[CR-465cd] If a device supports a device timestamp counter, then it MUST implement the Timestamp Value registers. Both registers MUST be available (High Part and Low Part)

[CR-466ca] If an application wants to retrieve the 64-bit value of the timestamp counter and the timestamp counter is supported by the device, it MUST write 1 into bit 30 (*latch*) of Timestamp Control register for the free-running timestamp counter to be copied into these registers. [CR27-28ca]

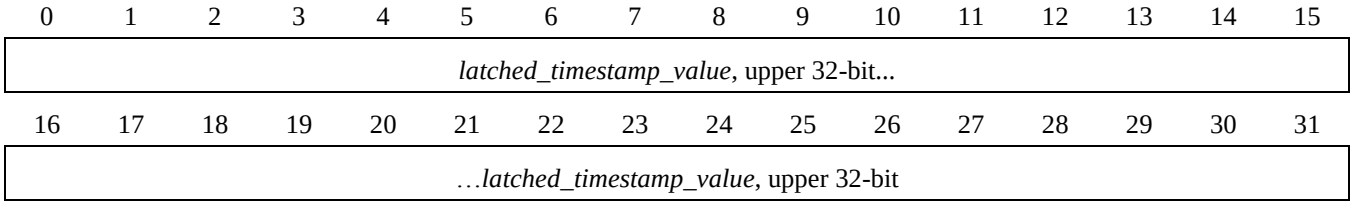
[CR-467cd] The Timestamp Value register MUST always be 0 if timestamp is not supported. [CR27-29cd]

These registers do not necessarily take all possible values when they increments: they can skip some value depending on the precision of the clock source relative to the Timestamp Tick Frequency register. For instance, if the device supports IEEE 1588, it must report a tick frequency of 1 GHz. But the quality of the IEEE 1588 master clock might not provide that level of accuracy.

An application needs to access the high part before accessing the low part.

28.33.1 High Part

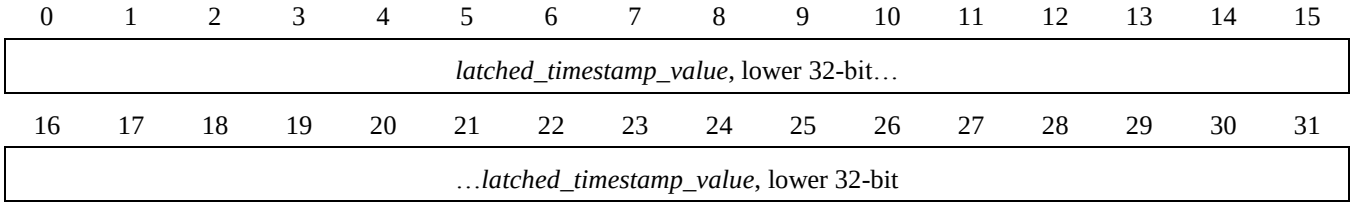
Address	0x0948 *[optional]*
Length	4 bytes
Access type	Read
Factory Default	0



Bits	Name	Description
0 – 31	*latched_timestamp_value*	Latched timestamp value, upper 32-bit

28.33.2 Low Part

Address	0x094C *[optional]*
Length	4 bytes
Access type	Read
Factory Default	0



Bits	Name	Description
0 – 31	*latched_timestamp_value*	Latched timestamp value, lower 32-bit

28.34 Discovery ACK Delay Register

This optional register indicates the maximum randomized delay in milliseconds the device will wait upon reception of a DISCOVERY_CMD before it will send back the DISCOVERY_ACK. This randomized delay can take any value from 0 second up to the value indicated by this bootstrap register.

An application can check bit 7 of the GVCP Capability register at address 0x0934 to determine if the Discovery ACK Delay register is supported by the device.

[O-468cd] A device SHOULD implement the Discovery ACK Delay register.

[CR-469cd] If Discovery ACK Delay register is writable, then its value MUST be persistent and stored in non-volatile memory to be re-used on next device reset or power cycle. [CR27-41cd]

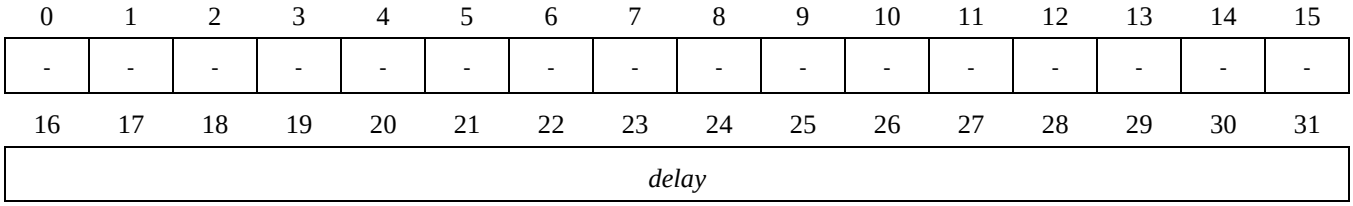
[CR-470cd] If Discovery ACK Delay is supported, the maximum factory default value for this register MUST be lower or equal to 1000 ms (1 second). [CR27-42cd]

The previous requirement is necessary to ensure backward compatibility with version 1.0 of the specification.

This register can be read-only if the device only supports a fixed value. Furthermore a device is allowed to set this register has read-only with a value of 0 if it does not support such a randomized delay. Bit 8 (*writable_discovery_ACK_delay*) of the GVCP Capability register (at address 0x0934) indicates if this register is writable.

An application should retrieve information from the XML device description file to see if this register is writable and to determine the maximum value it can take.

Address	0x0950
Length	4 bytes
Access type	Read/(Write)
Factory Default	Device-specific (≤1 second)



Bits	Name	Description
0 – 15	reserved	Always 0. Ignore when reading.
16 – 31	*delay*	Maximum random delay in ms to wait before sending DISCOVERY_ACK upon reception of DISCOVERY_CMD.

28.35 GVCP Configuration Register

This optional register provides additional control over GVCP. These additional functions must be indicated by the GVCP Capability register (at address 0x0934).

For instance, it can be used to disable Heartbeat when this capability is supported. This can be useful for debugging purposes.

[O-471cd] A device SHOULD implement the GVCP Configuration register.

Address	0x0954 *[optional]*
Length	4 bytes
Access type	Read/Write
Factory Default	see description

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
-	-	-	-	-	-	-	-	-	-	-	-	*PTP*	*ES2*	-	-
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
-	-	-	-	-	-	-	-	-	-	-	-	*UA*	*ES*	*PE*	*HD*

Bits	Name	Description
0 – 11	reserved	Always 0. Ignore when reading.
12	*IEEE1588_PTP_enable (PTP)*	Enable usage of the IEEE 1588 Precision Time Protocol to source the timestamp register. Only available when the *IEEE1588_support* bit of the GVCP Capability register is set. When PTP is enabled, the Timestamp Control register cannot be used to reset the timestamp. Factory default is device-specific. When PTP is enabled or disabled, the value of Timestamp Tick Frequency and Timestamp Value registers might change to reflect the new time domain.
13	*extended_status_code_ for_2_0 (ES2)*	Enable the generation of extended status codes introduced in specification 2.0: GEV_STATUS_PACKET_TEMPORARILY_UNAVAILABLE, GEV_STATUS_OVERFLOW, GEV_STATUS_NO_REF_TIME. Only available when the *extended_status_code_for_2_0* bit of the GVCP Capability register is set. Factory default must be 0.
14 – 27	reserved	Always 0. Ignore when reading.
28	*unconditional_ACTION_ enable (UA)*	Enable unconditional action command mode where ACTION_CMD are processed even when the primary control channel is closed. Only available when the *unconditional_ACTION* bit of the GVCP Capability register is set. Factory default must be 0. **Note**: *unconditional_ACTION_enable* is provided to allow some level of protection against inappropriate reception of action commands. This might be useful as a security measure in industrial control for instance.
29	*extended_status_code_ for_1_1 (ES)*	Enable generation of extended status codes introduced in specification 1.1: GEV_STATUS_PACKET_RESEND, GEV_STATUS_PACKET_NOT_YET_AVAILABLE, GEV_STATUS_PACKET_AND_PREV_REMOVED_FROM_MEMORY and GEV_STATUS_PACKET_REMOVED_FROM_MEMORY. Only available if the *extended_status_code_for_1_1* bit of the GVCP Capability register is set Factory default must be 0.
30	*PENDING_ACK_enable (PE)*	Enable generation of PENDING_ACK by this device. Only available when the *PENDING_ACK* capability bit of the GVCP Capability register is set is set. Factory default must be 0.
31	*heartbeat_disable (HD)*	Disable heartbeat. Only available when *heartbeat_disable* capability bit of the GVCP Capability register is set is set. Factory default must be 0.

28.36 Pending Timeout Register

This register indicates the longest GVCP command execution time before an ACK is returned. The packet travel time on the network is not accounted for by this register.

[R-472cd] A device MUST implement the Pending Timeout register.

Two scenarios exist:

1. If PENDING_ACK is disabled, then this register represents the worst-case single GVCP command execution time (not including concatenated read/write access).
2. If PENDING_ACK is enabled, then this register represents the maximum amount of time it takes before a PENDING_ACK is issued.

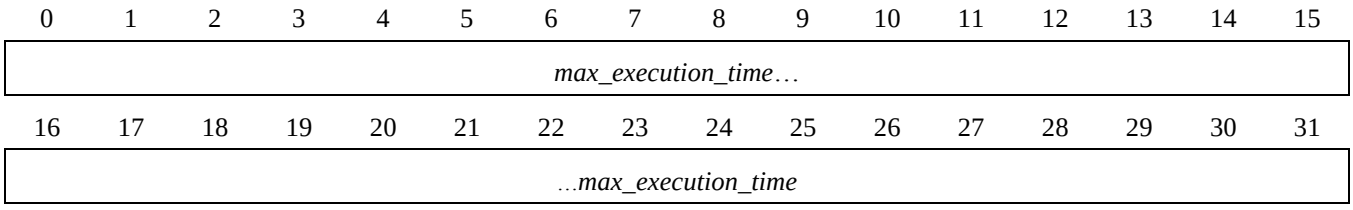
Therefore, a device might change the value of this read-only register when PENDING_ACK is enabled or disabled through the GVCP Configuration register (bit 30, *PENDING_ACK_enable* field, at address 0x0954).

The Application can use this value to deduce a suitable ACK timeout to wait for when it issues the various GVCP commands.

Note: The Pending Timeout register does not account for concatenated commands when they are supported. Hence the application must determine the actual worst case timeout to use by factoring in the number of read or write access when concatenation is used.

A device should strongly consider supporting PENDING_ACK if its worst-case GVCP command execution time is relatively long so the device stays responsive when short duration GVCP commands are lost during transmission and thus require retransmission.

Address	0x0958
Length	4 bytes
Access type	Read
Factory Default	Device-specific



Bits	Name	Description
0 – 31	*max_execution_time*	Provide the worst-case execution time (in ms) before the device will issue a PENDING_ACK, if supported and enabled, to notify the application to extend the ACK timeout for the current GVCP command. This time is for non-concatenated read/write accesses.

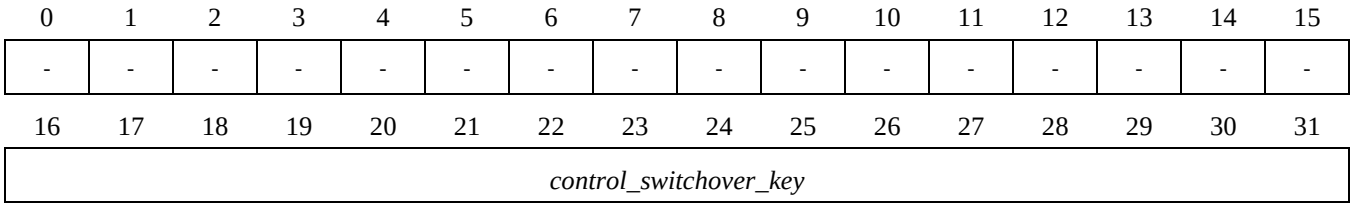
28.37 Control Switchover Key Register

This optional register provides the key to authenticate primary application switchover requests. The register is **write-only** to hide the key from secondary applications. The primary intent is to have a mechanism to control who can take control over a device. The primary application or a higher level system management entity can send the key to an application that would request control over a device.

An application can check bit 10 of the GVCP Capability register at address 0x0934 to determine if primary application switchover is supported by the device.

[CR-473cd] If a device supports primary application switchover, then it MUST implement the Control Switchover Key register.

Address	0x095C *[optional]*
Length	4 bytes
Access type	Write-only
Factory Default	0



Bits	Name	Description
0 – 15	reserved	Always 0.
16 – 31	*control_switchover_key*	Key to authenticate primary application switchover requests

28.38 GVSP Configuration Register

This optional register provides additional global control over GVSP configuration. These additional functions are indicated by the GVSP Capability register (at address 0x092C).

[O-474cd] A device SHOULD implement the GVSP Configuration register.

Address	0x0960 *[optional]*
Length	4 bytes
Access type	Read/Write
Factory Default	Device-specific

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
-	*BL*	-	-	-	-	-	-	-	-	-	-	-	-	-	-

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Bits	Name	Description
0	reserved	Always 0. Ignore when reading.
1	*64bit_block_id_enable (BL)*	Enable the 64-bit *block_id64* for GVSP. This bit cannot be reset if the stream channels do not support the standard ID mode (i.e. EI field is always 0). **Note**: If a transmitter supports 16-bit *block_id*, as reported by bit 1 (*legacy_16bit_block_id_supported*) of the GVSP Capability register (at address 0x092C), then it SHOULD have this bit set to 0 as the factory default to start in 16-bit *block_id* mode. Otherwise, it cannot be advertised as a GigE Vision 1.x device. **Note**: When 16-bit *block_id*s are used, then 24-bit *packet_id*s must be used. When 64-bit *block_id64*s are used, then 32-bit *packet_id32*s must be used.
2 – 31	reserved	Always 0. Ignore when reading.

28.39 Physical Link Configuration Capability Register

This register indicates the physical link configuration supported by this device.

[R-475cd] A device MUST implement the Physical Link Configuration Capability register.

Address	0x0964
Length	4 bytes
Access type	Read
Factory Default	Device-specific

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
-	-	-	-	-	-	-	-	-	-	-	-	*dLAG*	*sLAG*	*ML*	*SL*

Bits	Name	Description
0 – 27	reserved	Always 0. Ignore when reading.
28	*dynamic_link_aggregation_group_configuration (dLAG)*	This device supports dynamic LAG configuration.
29	*static_link_aggregation_group_configuration (sLAG)*	This device supports static LAG configuration.
30	*multiple_links_configuration (ML)*	This device supports multiple link (ML) configuration.
31	*single_link_configuration (SL)*	This device supports single link (SL) configuration.

28.40 Physical Link Configuration Register

This register indicates the principal physical link configuration currently enabled on this device. It should be used in conjunction with the Physical Link Configuration Capability register to determine which values are valid. After a change in configuration, it is necessary to restart the device (either by power cycling the device or by some other mean) in order for the new setting to take effect.

[R-476cd] A device MUST implement the Physical Link Configuration register.

Address	0x0968
Length	4 bytes
Access type	Read/Write
Factory Default	Device-specific

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
-	-	-	-	-	-	-	-	-	-	-	-	-	-	*link_configuration*	

Bits	Name	Description
0 – 29	reserved	Always 0. Ignore when reading.
30 - 31	*link_configuration*	Principal Physical Link Configuration to use on next restart/power-up of the device. 0: Single Link Configuration 1: Multiple Links Configuration 2: Static LAG Configuration 3: Dynamic LAG Configuration Note: IP configuration is not sufficient for the device to use the new link configuration. Hence FORCEIP_CMD cannot be used to use the new link configuration settings.

28.41 IEEE 1588 Status Register

This optional register indicates the state of the IEEE 1588 clock.

[CR-477cd] If a device supports IEEE 1588, then it MUST implement the IEEE 1588 Status register.

Address	0x096C
Length	4 bytes
Access type	Read
Factory Default	0

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
-	-	-	-	-	-	-	-	-	-	-	-	*clock_status*			

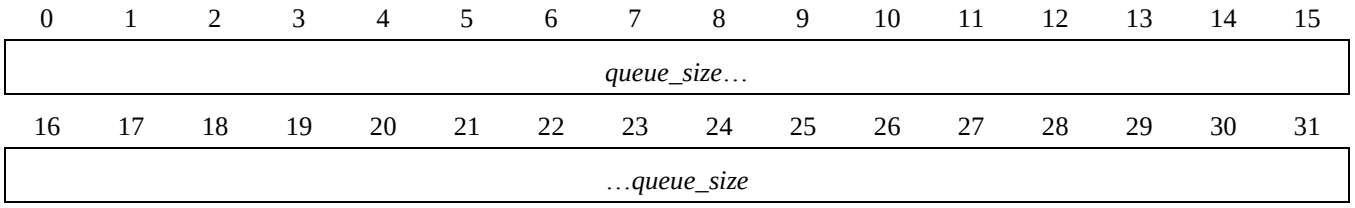
Bits	Name	Description
0 – 27	reserved	Always 0. Ignore when reading.
28 – 31	*clock_status*	Provides the state of the IEEE 1588 clock. Values of this field must match the IEEE 1588 PTP port state enumeration (INITIALIZING, FAULTY, DISABLED, LISTENING, PRE_MASTER, MASTER, PASSIVE, UNCALIBRATED, SLAVE). Please refer to IEEE 1588 for additional information.

28.42 Scheduled Action Command Queue Size Register

This optional register reports the number of Scheduled Action Commands that can be queued at any given time. This is essentially the size of the queue. This register is present when bit 14 (*scheduled_action_command*) of the GVCP Capability register is set

[CR-478cd] If a device supports Scheduled Action Commands, then it MUST implement the Scheduled Action Command Queue Size register.

Address	0x0970
Length	4 bytes
Access type	Read
Factory Default	Device-specific



Bits	Name	Description
0 – 31	*queue_size*	Indicates the size of the queue. This number represents the maximum number of Scheduled Action Commands that can be pending at a given point in time.

28.43 Control Channel Privilege Register (CCP)

This register is used to grant privilege to an application. Only one application is allowed to control the device. This application is able to write into device's registers. Other applications can read device's register only if the controlling application does not have the exclusive privilege.

[R-479cd] A device MUST implement the Control Channel Privilege register.

Address	0x0A00
Length	4 bytes
Access type	Read/Write
Factory Default	0

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
control_switchover_key															
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
-	-	-	-	-	-	-	-	-	-	-	-	-	*CSE*	*CA*	*EA*

Bits	Name	Description
0 – 15	*control_switchover_key*	This field is applicable only when the device supports the primary application switchover capability. It is taken into account only when the *control_switchover_enable* bit (bit 29) was set prior to the current transaction writing to the CCP register and the transaction comes from a different application. In this case, the primary application switchover will occur only if the value written to this field matches the value in the Control Switchover Key bootstrap register. No matter if the primary application switchover capability is supported or not, this field always reads as zero.
16 – 28	reserved	Always 0. Ignore when reading.
29	*control_switchover_enable (CSE)*	This bit is applicable only when the device supports the primary application switchover capability. The application writing a 1 to this bit enables another application to request exclusive or control access to the device (without or with switchover enabled). This bit is used in conjunction with the *control_access* bit (bit 30). It is “don’t care” when the *exclusive_access* bit (bit 31) is set. This bit always reads as zero when the primary application switchover capability is not supported.
30	*control_access (CA)*	The application writing a 1 to this bit requests control access to the device (without or with switchover enabled depending of the value written to the *control_switchover_enable* bit). If a device grants control access, no other application can control the device if the *control_switchover_enable* bit is set to zero. Otherwise, another application can request to take over exclusive or control access of the device (without or with switchover enabled). Other applications are still able to monitor the device.
31	*exclusive_access (EA)*	The application writing a 1 to this bit requests exclusive access to the device. If a device grants exclusive access, no other application can control or monitor the device. *exclusive_access* has priority over *control_access* when both bits are set.

Since exclusive access is more restrictive than control access (without or with switchover enabled), the device must be in exclusive access mode if the *exclusive_access* bit is set independently of the state of the *control_access* and *control_switchover_enable* bits. Control access is defined according to Table 28-2.

Table 28-2: Control Access Definition

control_switchover_enable	*control_access*	*exclusive_access*	Access Mode
x	0	0	Open access.
0	1	0	Control access.
1	1	0	Control access with switchover enabled.
x	x	1	Exclusive access.

28.44 Primary Application Port Register

This optional register provides UDP port information about the primary application holding the control channel privilege.

An application can check bit 4 of the GVCP Capability register at address 0x0934 to determine if the Primary Application Port register is supported by the device.

[O-480cd] A device SHOULD implement the Primary Application Port register.

Address	0x0A04 *[optional]*
Length	4 bytes
Access type	Read
Factory Default	0

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
primary_application_port															

Bits	Name	Description
0 – 15	reserved	Always 0. Ignore when reading.
16 – 31	*primary_application_port*	The UDP source port of the primary application. This value must be 0 when no primary application is bound to the device (CCP register equal to 0).

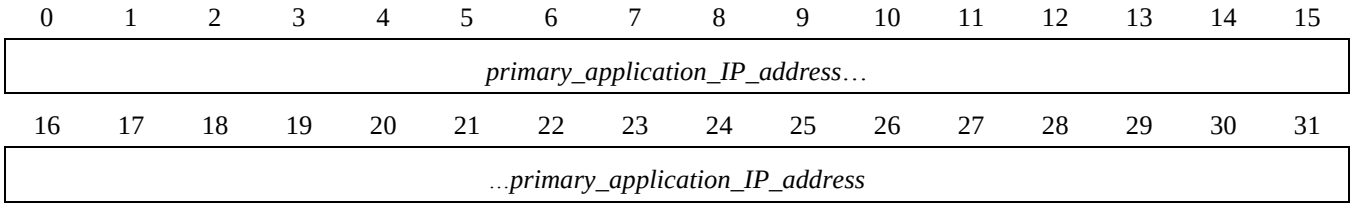
28.45 Primary Application IP Address Register

This optional register provides IP address information about the primary application holding the control channel privilege.

An application can check bit 4 of the GVCP Capability register at address 0x0934 to determine if the Primary Application IP Address register is supported by the device.

[O-481cd] A device SHOULD implement the Primary Application IP Address register.

Address	0x0A14 *[optional]*
Length	4 bytes
Access type	Read
Factory Default	0



Bits	Name	Description
0 – 31	*primary_application_IP_address*	The IPv4 address of the primary application. This must be a unicast address. This value must be 0 when no primary application is bound to the device (CCP register equal to 0).

28.46 Message Channel Port Register (MCP)

This optional register provides port information about the message channel.

[CR-482cd] If a device supports a Message Channel, then it MUST implement the Message Channel Port register.

Address	0x0B00 *[optional]*
Length	4 bytes
Access type	Read/Write
Factory Default	0

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
-	-	-	-	-	-	-	-	-	-	-	-	*network_interface_index*			
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
host_port															

Bits	Name	Description
0 – 11	reserved	Always 0. Ignore when reading.
12 – 15	*network_interface_index*	Always 0 in this version. Only the primary interface (#0) supports GVCP.
16 – 31	*host_port*	The port to which the device must send messages. Setting this value to 0 closes the message channel.

28.47 Message Channel Destination Address Register (MCDA)

This optional register indicates the destination IP address for the message channel.

[CR-483cd] If a device supports a Message Channel, then it MUST implement the Message Channel Destination Address register.

Address	0x0B10 *[optional]*
Length	4 bytes
Access type	Read/Write
Factory Default	0

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
channel_destination_IP…															
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
…channel_destination_IP															

Bits	Name	Description
0 – 31	*channel_destination_IP*	Message channel destination IPv4 address. The destination address can be a multicast or a unicast address.

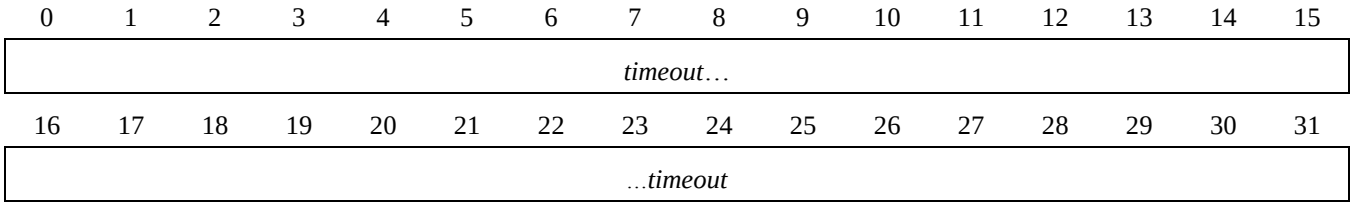
28.48 Message Channel Transmission Timeout Register (MCTT)

This optional register provides the transmission timeout value in milliseconds.

[CR-484cd] If a device supports a Message Channel, then it MUST implement the Message Channel Transmission Timeout register.

[CR-485cd] This register indicates the amount of time the device MUST wait for acknowledge after a message is sent on the message channel before timeout when an acknowledge is requested when a message channel is supported. [CR27-32cd]

Address	0x0B14 *[optional]*
Length	4 bytes
Access type	Read/Write
Factory Default	Device-specific



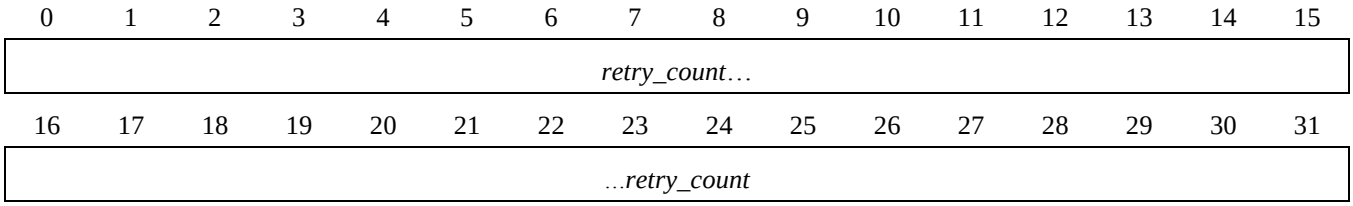
Bits	Name	Description
0 – 31	*timeout*	Transmission timeout value in ms. Set to 0 to disable acknowledge on message channel.

28.49 Message Channel Retry Count Register (MCRC)

This optional register indicates the number of retransmissions allowed when a message channel message times out.

[CR-486cd] If a device supports a Message Channel, then it MUST implement the Message Channel Retry Count register.

Address	0x0B18 *[optional]*
Length	4 bytes
Access type	Read/Write
Factory Default	Device-specific



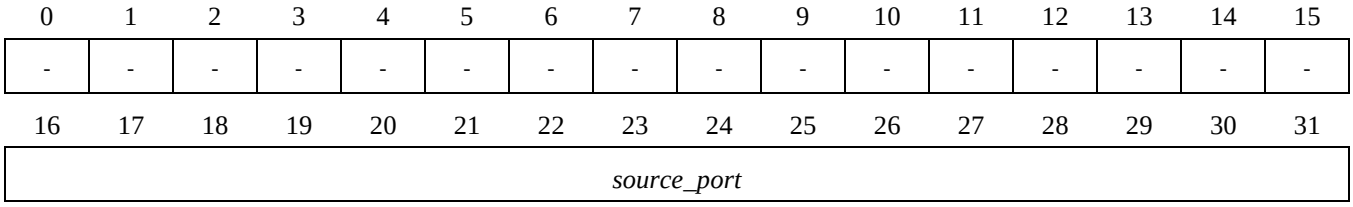
Bits	Name	Description
0 – 31	*retry_count*	Number of retransmissions allowed on the message channel.

28.50 Message Channel Source Port Register (MCSP)

This optional register indicates the source UDP port for the message channel. The purpose of this information is to allow the host application to take measures to ensure asynchronous event traffic from the device is allowed back to the host. The expected usage of this is to send dummy UDP packets from the application's message channel reception port to that device's source port, thus making the message channel traffic appear to be requested by the application instead of unsolicited.

[CO-487cd] If a device supports a Message Channel, then it SHOULD implement the Message Channel Source Port register.

Address	0x0B1C *[optional]*
Length	4 bytes
Access type	Read-Only
Factory Default	0 if unavailable



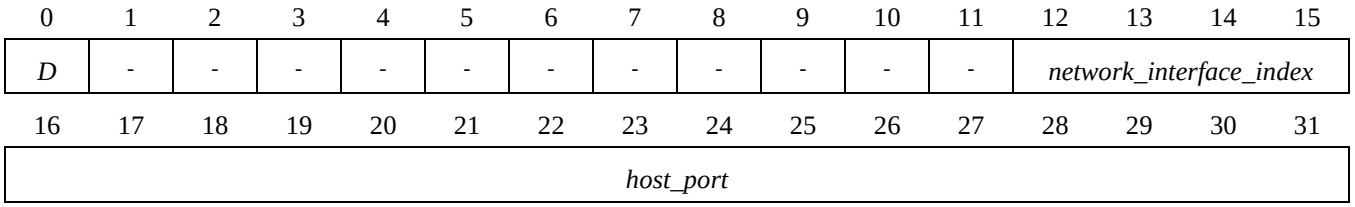
Bits	Name	Description
0 - 15	reserved	Always 0. Ignore when reading.
16 - 31	*source_port*	Indicates the UDP source port message channel traffic will be generated from while the message channel is open.

28.51 Stream Channel Port Registers (SCPx)

These registers provide port information for the given stream channel.

[CR-488cd] If a device supports stream channels, it MUST implement the Stream Channel Port register for all of its supported stream channels.

Address	0x0D00 + 0x40 * x with 0 ≤ x < 512. *[optional for 0 < x < 512 for GVSP transmitters and receivers]* *[Not applicable for peripherals]*
Length	4 bytes
Access type	Read/Write
Factory Default	Device-specific



Bits	Name	Description
0	*direction (D)*	This bit is read-only and reports the direction of the stream channel. It must take one of the following values 0: Transmitter 1: Receiver
1 – 11	reserved	Always 0. Ignore when reading.
12 – 15	*network_interface_index*	Index of network interface to use (from 0 to 3). Specific streams might be hard-coded to a specific network interfaces. Therefore, this field might not be programmable on certain devices. It is read-only for this case. For link aggregation configuration, only a single network interface is made visible by the device.
16 - 31	*host_port*	The port to which a GVSP transmitter must send data stream. The port from which a GVSP receiver may receive data stream. Setting this value to 0 closes the stream channel. For a GVSP transmitter, the stream channel block ID must be reset to 1 when the stream channel is opened.

28.52 Stream Channel Packet Size Registers (SCPSx)

These registers indicate the packet size in bytes for the given stream channel. This is the total packet size, including all headers (IP header = 20 bytes, UDP header = 8 bytes and GVSP header). For GVSP transmitters, they also provide a way to set the IP header "don't fragment" bit and to send stream test packet to the GVSP receiver.

[CR-489cd] If a device supports stream channels, it MUST implement the Stream Channel Packet Size register for all of its supported stream channels.

Write access to this register is not possible while streaming (transmission) is active on this channel.

Address	0x0D04 + 0x40 * x with 0 ≤ x < 512. *[optional for 0 < x < 512 for GVSP transmitters and receivers]* *[Not applicable for peripherals]*
Length	4 bytes
Access type	Read/Write
Factory Default	Device-specific

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
F	*D*	*P*	-	-	-	-	-	-	-	-	-	-	-	-	-

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
packet_size															

Bits	Name	Description
0	*fire_test_packet (F)*	When this bit is set and the stream channel is a transmitter, the GVSP transmitter will fire one test packet of size specified by bits 16 to 31. The "don't fragment" bit of IP header must be set for this test packet. If the GVSP transmitter supports the LFSR generator, then the test packet payload will be filled with data according to this polynomial. Otherwise, the payload data is "don't care". When the stream channel is a receiver, this bit doesn't have any effect and always reads as zero.
1	*do_not_fragment (D)*	For GVSP transmitters, this bit is copied into the "don't fragment" bit of IP header of each stream packet. It can be used by the application to prevent IP fragmentation of packets on the stream channel. When the stream channel is a receiver, this bit doesn't have any effect and always reads as zero.

2	*pixel_endianess (P)*	Endianess of multi-byte pixel data for this stream. 0: little-endian 1: big-endian This is an optional feature. A GVSP transmitter or receiver that does not support this feature must support little-endian and always leave that bit clear.
3 – 15	reserved	Always 0. Ignore when reading.
16 – 31	*packet_size*	For GVSP transmitters, this field represents the stream packet size to send on this channel, except for data leader and data trailer; and the last data packet which might be of smaller size (since packet size is not necessarily a multiple of block size for stream channel). The value is in bytes. For a camera, it can be accessed through the GevSCPSPacketSize mandatory feature. If a GVSP transmitter cannot support the requested packet_size, then it must not fire a test packet when requested to do so. For GVSP receivers, this field represents the maximum GVSP packet size supported by the receiver. It is read-only.

The packet size is used by the application to determine where it has to copy the data in the receiving buffer. As such, it is important that this value remains the same once it has been configured by the application.

[R-490st] A GVSP transmitter is allowed to round the value of the *packet_size* field to the nearest supported value when the application writes to the SCPSx register. But the transmitter MUST not change the packet size value without having the application directly writing into this register.

For instance, it is allowed for the GVSP transmitter to round a packet size value to a multiple of 4 bytes, but it is not allowed for the transmitter to change the packet size if the application changes the data format (such as the image width or pixel format for a camera device).

It is recommended for the application to read back the packet size after writing to SCPSx since the value could have been rounded by the GVSP transmitter. The application should also ensure the packet size contains a valid value before asking for a Test Packet.

Note: An application must be able to cope with such rounding of the packet size during packet size negotiation. This is typically performed using the test packet function.

28.53 Stream Channel Packet Delay Registers (SCPDx)

These registers indicate the minimal delay (in timestamp counter unit) to insert between each packet transmitted on a given physical link for the stream channel. This can be used as a crude flow-control mechanism if the GVSP receiver cannot keep up with the packets coming from the device.

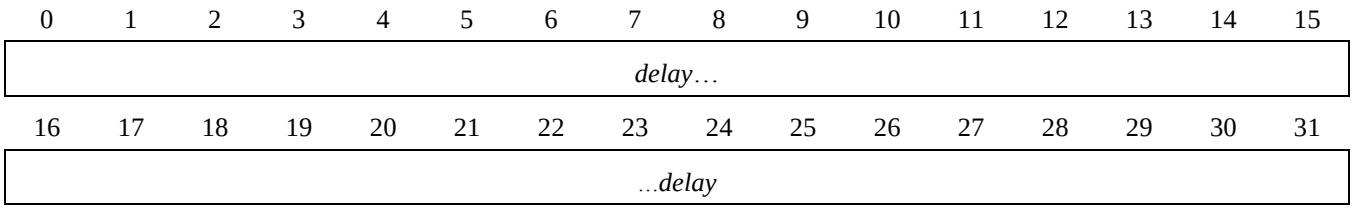
Note: The above essentially means that each network interface must have a separate timer per stream to measure the inter-packet delay.

As an addition to the inter-packet delay, a device might elect to support the PAUSE option of IEEE 802.3.

[CR-491cd] If a device supports stream channels, it MUST implement the Stream Channel Packet Delay register for all of its supported stream channels.

This delay normally uses the same granularity as the timestamp counter to ensure a very high precision in the packet delay. This counter is typically of very high frequency. Inter-packet delay is typically very small. If the timestamp is not supported, then this register has no effect.

Address	0x0D08 + 0x40 * x with 0 ≤ x < 512. *[optional for 0 < x < 512 for GVSP transmitters]* *[Not applicable for GVSP receivers and peripherals]*
Length	4 bytes
Access type	Read/Write
Factory Default	0



Bits	Name	Description
0 – 31	*delay*	Inter-packet delay in timestamp ticks.

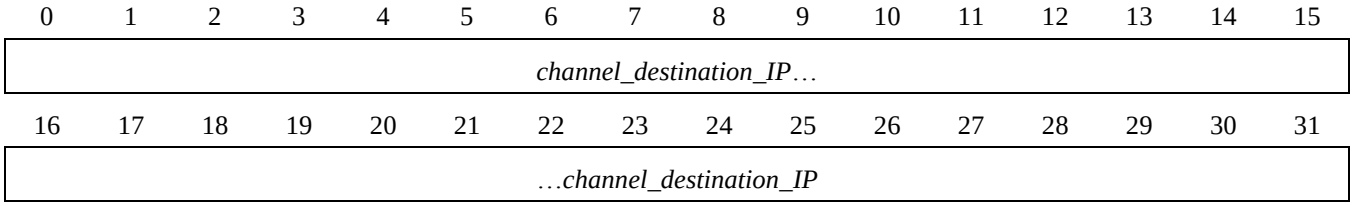
28.54 Stream Channel Destination Address Registers (SCDAx)

For GVSP transmitters, these registers indicate the destination IP address for the given stream channel. For GVSP receivers, these registers indicate the destination IP address from which the given receiver may receive data stream from.

[R-492cd] If a device supports stream channels, it MUST implement the Stream Channel Destination Address register for all of its supported Stream Channels.

For GVSP transmitters, write access to this register is not possible while streaming is active on this channel.

Address	0x0D18 + 0x40 * x with 0 ≤ x < 512. *[optional for 0 < x < 512 for GVSP transmitters and receivers]* *[Not applicable for peripherals]*
Length	4 bytes
Access type	Read/Write
Factory Default	0 if transmitter (*SCPx_direction* field is set to 0) Device-specific if receiver (*SCPx_direction* field is set to 1)



Bits	Name	Description
0 – 31	*channel_destination_IP*	Stream channel destination IPv4 address. The destination address can be a multicast or a unicast address.

28.55 Stream Channel Source Port Registers (SCSPx)

These optional registers indicate the source UDP port for the given GVSP transmitter stream channel. The purpose of this information is to allow the host application to take measures to ensure streaming traffic from the device is allowed back to the GVSP receiver. The expected usage of this is to send dummy UDP packets from the GVSP receiver stream reception port to that GVSP transmitter source port, thus making the streaming traffic appear to be requested by the GVSP receiver instead of unsolicited.

[CO-493cd] If a device supports stream channels, it SHOULD implement the Stream Channel Source Port register for all of its supported stream channels.

Address	0x0D1C + 0x40 * x with 0 ≤ x < 512. *[optional for 0 ≤ x < 512 for GVSP transmitters]* *[Not applicable for GVSP receivers and peripherals]*
Length	4 bytes
Access type	Read-Only
Factory Default	0 if unavailable

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
source_port															

Bits	Name	Description
0 - 15	reserved	Always 0. Ignore when reading.
16 - 31	*source_port*	Indicates the UDP source port GVSP traffic will be generated from while the streaming channel is open.

28.56 Stream Channel Capability Registers (SCCx)

These optional registers provide a list of capabilities specific to the given stream channel.

[CO-494cd] If a device supports stream channels, it SHOULD implement the Stream Channel Capability register for all of its supported stream channels.

Address	0x0D20 + 0x40 * x with 0 ≤ x < 512. *[optional for 0 ≤ x < 512 for GVSP transmitters and receivers]* *[Not applicable for peripherals]*
Length	4 bytes
Access type	Read
Factory Default	Device-specific

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
BE	*R*	-	-	-	-	-	-	-	-	-	-	-	-	-	-

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
-	-	-	-	-	-	-	-	-	-	-	MZ	*PRD*	*AIT*	*US*	*EC*

Bits	Name	Description
0	*big_and_little_endian_supported (BE)*	Indicates this stream channel supports both big and little-endian. Note that all stream channels must support little-endian.
1	*IP_reassembly_supported (R)*	For GVSP receivers, indicates this stream channel supports the reassembly of fragmented IP packets. Otherwise, it reads as zero.
2-26	reserved	Always 0. Ignore when reading.
27	*multi_zone_supported (MZ)*	Set to 1 to indicate this stream channel supports the SCZx and SCZDx registers.
28	*packet_resend_destination_option (PRD)*	Indicates if an alternate destination exists for Packet resend request. This is only valid for GVSP transmitter. For GVSP transmitter: 0: No alternative, always use the destination specified by SCDAx. 1: Use the destination indicated by the *PRD* flag of the SCCFGx register. For GVSP receiver: must be 0.
29	*all_in_transmission_supported (AIT)*	For GVSP transmitters, indicates that the stream channel supports the All-in Transmission mode. Otherwise, it reads as zero.
30	*unconditional_streaming_supported (US)*	For GVSP transmitters, indicates that the stream channel supports unconditional streaming capabilities. Otherwise, it reads as zero.
31	*extended_chunk_data_supported (EC)*	Indicates that the stream channel supports the deprecated extended chunk data payload type. Otherwise, it reads as zero.

28.57 Stream Channel Configuration Registers (SCCFGx)

These optional registers provide additional control over optional features specific to the given stream channel. These optional features must be indicated by the Stream Channel Capability register of the given stream channel.

For instance, they can be used to enable GVSP transmitters to use the All-in Transmission mode.

[CO-495cd] If a device supports stream channels, it SHOULD implement the Stream Channel Configuration register for all of its supported stream channels.

Address	0x0D24 + 0x40 * x with $0 \leq x < 512$. *[optional for $0 \leq x < 512$ for GVSP transmitters and receivers]* *[Not applicable for peripherals]*
Length	4 bytes
Access type	Read/Write
Factory Default	0

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
-	-	-	-	-	-	-	-	-	-	-	-	*PRD*	*AIT*	*US*	*EC*

Bits	Name	Description
0 – 27	reserved	Always 0. Ignore when reading.
28	*packet_resend_destination (PRD)*	Enable the alternate IP destination for stream packet resent due to a Packet Resend request. This is only valid for GVSP transmitter. For GVSP transmitter: 0: Use SCDAx 1: Use the source IP address provided in the PACKETRESEND_CMD packet For GVSP receiver: don't care.
29	*all_in_transmission_enabled (AIT)*	Enable GVSP transmitter (*direction* field of SCPx set to 0) to use the single packet per data block All-in Transmission mode.
30	*unconditional_streaming_enabled (US)*	Enable GVSP transmitters (*direction* field of SCPx set to 0) to continue to stream if the control channel of its associated device is closed or regardless of any ICMP messages, such as the destination unreachable messages, received by the device associated to them. This bit has no effect for GVSP receivers (*direction* field of SCPx set to 1). In the latter case, it always reads back as zero.
31	*extended_chunk_data_enable (EC)*	Enable GVSP transmitters (*direction* field of SCPx set to 0) to use the deprecated extended chunk data payload type. This bit has no effect for GVSP receivers (*direction* field of SCPx set to 1). In the latter case, it always reads back as zero.

28.58 Stream Channel Zone Registers (SCZx)

These optional registers indicate the number of zones in a Multi-zone Image data block transmitted on the corresponding GVSP transmitter stream channels.

This register is available when bit 27 of the corresponding SCCx register is set.

[CR-496cd] If a device supporting stream channels also supports the Multi-zone Image payload type, then it MUST implement the Stream Channel Zone register for each stream channel supporting the Multi-zone Image payload type.

Address	0x0D28 + 0x40 * x with 0 ≤ x < 512. *[optional for 0 ≤ x < 512 for GVSP transmitters]* *[Not applicable for GVSP receivers and peripherals]*
Length	4 bytes
Access type	Read-Only
Factory Default	Device-specific

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
-	-	-	-	-	-	-	-	-	-	-	*additional_zones*				

Bits	Name	Description
0 – 26	reserved	Always 0. Ignore when reading.
27 – 31	*additional_zones*	Reports the number of additional zones in a block transmitted on the corresponding stream channel. The number of zones is equal to the value of this field plus one.

28.59 Stream Channel Zone Direction Registers (SCZDx)

These optional registers indicate the transmission direction of each zone in a Multi-zone Image data block transmitted on the corresponding GVSP transmitter stream channels.

This register is available when bit 27 of the corresponding SCCx register is set.

[CR-497cd] If a device supporting stream channels supports the Multi-zone Image payload type, the it MUST implement the Stream Channel Zone Direction register for each stream channel supporting the Multi-zone Image payload type.

Address	0x0D2C + 0x40 * x with 0 ≤ x < 512. *[optional for 0 ≤ x < 512 for GVSP transmitters]* *[Not applicable for GVSP receivers and peripherals]*
Length	4 bytes

Access type	Read-Only
Factory Default	Device-specific

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Z0	*Z1*	*Z2*	*Z3*	*Z4*	*Z5*	*Z6*	*Z7*	*Z8*	*Z9*	*Z10*	*Z11*	*Z12*	*Z13*	*Z14*	*Z15*
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Z16	*Z17*	*Z18*	*Z19*	*Z20*	*Z21*	*Z22*	*Z23*	*Z24*	*Z25*	*Z25*	*Z27*	*Z28*	*Z29*	*Z30*	*Z31*

Bits	Name	Description
0 – 31	*zone_transmission_ direction (Zx)*	Reports the transmission direction of each zone. bit 0 (msb) - Direction of zone 0. Reports the transmission direction of zone ID 0. When set to zero, the zone is transmitted top-down. Otherwise, it is transmitted bottom up. bit 1 - Direction of zone 1. Reports the transmission direction of zone ID 1. When set to zero, the zone is transmitted top-down. Otherwise, it is transmitted bottom up. … bit 31 (lsb) - Direction of zone 31. Reports the transmission direction of zone ID 31. When set to zero, the zone is transmitted top-down. Otherwise, it is transmitted bottom up.

28.60 Manifest Table

The optional Manifest Table starts with a header indicating the number of entries in the table followed by a list of entries, each describing one document. This table is only present if the *manifest_table* capability bit (bit 5) is set in the GVCP Capability register (at address 0x0934).

The Manifest Table is optional for a GigE Vision device. Even if this Manifest table is present, the device must provide valid entries in the First URL and Second URL registers.

[O-498cd] A device SHOULD implement the Manifest Table.

[R-499ca] An application MUST support Manifest Tables.

For 8-byte entries in the Manifest Table, the application needs to access the high part (bit 0 to 31, lowest address) before accessing the low part (bit 32 to 63, highest address).

Address	0x9000 *[optional]*
Length	512 bytes
Access type	Read
Factory Default	Device-specific

Address	Name	Support	Type	Length (bytes)	Description
0x9000	ManifestHeader	M*	R	8	Header of the manifest table
0x9000+8	ManifestEntry_1	M*	R	8	First entry in the manifest table
0x9000+16	ManifestEntry_2	O	R	8	Second entry in the manifest table
…	…	…	…	…	…
0x9000+8*N	ManifestEntry_N	O	R	8	Entry N in the manifest table
…	…	…	…	…	
0x9000+8*63	ManifestEntry_63	O	R	8	Last entry in the manifest table

*: mandatory only if Manifest Table is supported.

28.60.1 ManifestHeader

The ManifestHeader contains the number of manifest entries.

The number of entries can be 0, in which situation the First URL register (at address 0x0200) and Second URL register (at address 0x0400) shall be used to retrieve the XML device description file.

Bits	Name	Description
0 – 5	*NumEntries*	Number (N) of entries in the Manifest Table where $0 \leq N \leq 63$. When N equals 0, then the First URL register and Second URL register must be used.
6 – 63	reserved	Always 0. Ignore when reading.

28.60.2 ManifestEntry

Each ManifestEntry describes the type and version of one XML document and refers to a pair of URL registers indicating the location of the XML document. This URL pair must provide addresses outside of the bootstrap register section. This means the URL pair must be stored in the manufacturer-specific register space (which starts at address 0xA000).

Bits	Name	Description
0 – 5	*XMLMajorVersion*	The major version number of the referenced GenApi XML file
6 – 11	*XMLMinorVersion*	The minor version number of the referenced GenApi XML file
12 – 17	*XMLSubMinorVersion*	The subminor version number of the referenced GenApi XML file
18 – 23	*SchemaMajorVersion*	The major version number of the schema used by the XML file.
24 – 29	*SchemaMinorVersion*	The minor version number of the schema used by the XML file
30 – 31	reserved	Always 0. Ignore when reading.
32 – 63	*URLRegisterAddress*	This field contains the address to the URLPair Register used to name and locate the document. This address must be located in the manufacturer-specific register space except if this register points to the First URL register at address 0x0200.

The ManifestRegister is defined in a way that the First URL register and the Second URL register defined since version 1.0 of this specification can be reused.

A minimum Manifest Table for a camera supporting an XML file with revision 3.1.0 which is compliant to the GenApi schema version 1.0 would for example look like this:

```
Manifest.ManifestHeader.NumEntries = 1
Manifest.ManifestEntry_1.XMLMajorVersion = 3
Manifest.ManifestEntry_1.XMLMinorVersion = 1
Manifest.ManifestEntry_1.XMLSubMinorVersion = 0
Manifest.ManifestEntry_1.SchemaMajorVersion = 1
Manifest.ManifestEntry_1.SchemaMinorVersion = 0
Manifest.ManifestEntry_1.URLRegisterAddress = 0x0200
```

28.60.3 URL Pair

For each document, a primary and a secondary URL is given describing how to retrieve the document. An application should first try the primary URL and only if that fails fall back to the secondary URL. This is the same scheme used for the First URL bootstrap register (at address 0x0200) and Second URL bootstrap register (at address 0x0400), as explained in section 14.1.

[CR-500cd] If a URL Pair is supported, then the URL strings referenced by URL Pair MUST use the character set indicated in the Device Mode register. [CR27-47cd]

Address	Device-specific
Length	1024 bytes
Access type	Read
Factory Default	Device-specific

Bytes	Name	Description
0 – 511	*primary_URL*	String providing the primary URL to the document.
512 – 1023	*secondary_URL*	String providing the secondary URL to the document.

[CR-501cd] If a URL Pair is supported, then in case a file is zipped, the referenced document MUST be the first one to be found in the ZIP file. Any number of other files may follow but their content is not defined by this specification. [CR27-48cd]

[CR-502cd] If a URL Pair is supported, then the Ntuple {XMLMajorVersion, XMLMinorVersion, XMLSubMinorVersion, SchemaMajorVersion, SchemaMinorVersion} in the manifest entry MUST change if the document's content changes. Thus this Ntuple can be used as key for caching the document. [CR27-49cd]

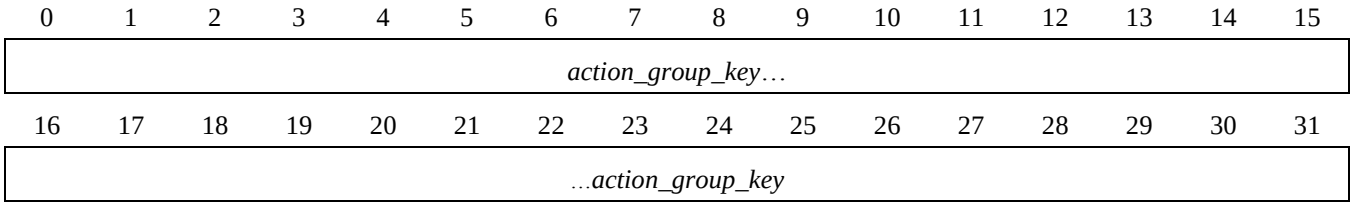
28.61 Action Group Key Registers (ACTION_GROUP_KEYx)

These optional registers provide the group key associated with the given device signal (ACTION_x). Up to 128 actions can be supported by the device (ACTION_0 up to ACTION_127).

[CR-503cd] If a device supports Action commands, then it MUST implement Action Group Key registers for the number of supported actions (up to 128).

When action commands are supported, these registers provide the 32-bit group key the device uses to match against the *group_key* field of incoming ACTION_CMD packets for the given device signal.

Address	0x9800 + 0x10 * x with $0 \leq x < 128$. *[optional]*
Length	4 bytes
Access type	Read/Write
Factory Default	0



Bits	Name	Description
0 – 31	*action_group_key*	Action Group Key entry.

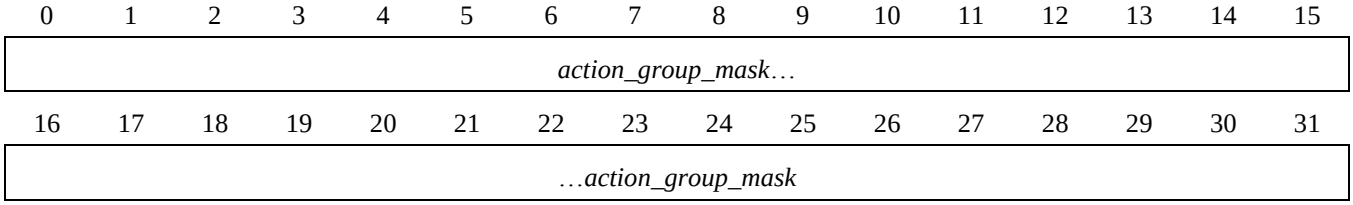
28.62 Action Group Mask Registers (ACTION_GROUP_MASKx)

These optional registers provide the group mask associated with the given device signal (ACTION_x). Up to 128 actions can be supported by the device.

[CR-504cd] If a device supports Action commands, then it MUST implement Action Group Mask registers for the number of supported actions (up to 128).

When action commands are supported, these registers provide the 32-bit group mask the device uses to "bitwise AND" against the *group_mask* field of incoming ACTION_CMD packets for the given device signal.

Address	0x9804 + 0x10 * x with 0 ≤ x < 128. *[optional]*
Length	4 bytes
Access type	Read/Write
Factory Default	0 (disabled)



Bits	Name	Description
0 – 31	*action_group_mask*	Action Group Mask entry.

29 Standard Features List for Cameras

29.1 Introduction

The GigE Vision specification relies on the GenICam™ specification (www.genicam.org) to describe the features supported by the camera. This description takes the form of an XML device description file respecting the syntax defined by the GenApi module of the GenICam specification.

[R-505cd] To allow interoperability between GigE Vision cameras and GigE Vision application software, any GigE Vision device MUST provide an XML device description file compliant to the GenApi module of the GenICam specification. [R28-1cd]

This XML file is retrieved and interpreted by the application software to enumerate the features supported by the device. The XML device description file provides the mapping between a device feature and the device register supporting it.

Since a large portion of GigE Vision devices are cameras, this section provides a list of mandatory features that must be present in the GigE Vision camera XML description file. Note that non-camera devices are not covered by this section. Any camera providing an XML device description file compliant to the syntax of the GenApi module of the GenICam specification and including the mandatory features presented in this section is considered suitable for GigE Vision.

Note the requirements in this section only apply to camera devices. Therefore, all requirements and objectives below are conditional as they only apply to this type of GigE Vision devices.

29.2 GenICam™ Standard

The GenICam is standard and trademark of the European Machine Vision Association (EMVA, www.emva.org).

The GenICam standard provides a high level of dynamism since the feature mapping can be tailored to a specific camera. This is very different from the GigE Vision bootstrap registers which forces a unique mapping for all cameras. This dynamism offers the advantage that features of the camera can be determined and described by the camera manufacturer. The naming of these features can thus follow the manufacturer naming convention.

The drawback of this flexibility is that application software cannot recognize the meaning of a particular feature name. A way around this limitation is to provide a set of standard feature names to be used across various camera models. This way, application software is aware of the meaning associated to a given feature name. The extent to which these standard feature names are defined may limit the freedom of camera manufacturer to implement a given feature. Therefore, care should be taken not to over-specify all features.

29.3 Level of Interoperability

An important consideration is the level of interoperability achieved between GigE Vision cameras and application software.

The simplest level of interoperability is realized when a graphical user interface (GUI) simply displays the list of features. This is often realized by a camera configuration program. In this case, it is the user who

looks and interprets the meaning of each feature. On-line help (such as tooltips) can be used to explain the meaning of the feature.

A more complex level of interoperability is achieved when the software is able to correctly interpret the functionality associated to a particular feature name. In this case, the software can take appropriate decision without the need for the user to get involved.

One issue with the level of interoperability is the number of characteristics associated to a feature:

1. Name
2. Representation (integer, float, boolean, enumeration, …)
3. Unit of measurement
4. Behavior

For instance, the feature "Gain" can be expressed as a float using the decibels unit where the value represents the level of amplification to apply to the analog video signal coming out of the sensor. But one could also define the Gain as an enumeration that can take a pre-determined set of value, such as {GAIN_x1, GAIN_x2, GAIN_x4}. Thus only standardizing the feature name is insufficient to guarantee interoperability.

Considering the large number of camera manufacturers and image processing software provider, it is difficult to reach a consensus on feature name, representation, unit and behavior. In many situations, the representation of a feature is directly linked to the camera architecture.

29.4 Use Cases

One of the goals of GigE Vision is to allow a user to buy a GigE Vision camera and easily interface it to any GigE Vision compliant application software. In order to guarantee a minimal level of interoperability, the GigE Vision specification addresses two use cases:

1. Continuous acquisition and display of images
2. Simplest GigE Vision camera

The features required to support these two use cases are defined mandatory by this specifications. These mandatory features are described in this section.

Some other features are listed in a separate document: "GenICam Standard Features Naming Convention". The GigE Vision specification recommends to camera manufacturers they use the name provided in that document whenever possible to facilitate interoperability with third-party software.

> **Note**: The mandatory feature list only provides for a single stream channel. If multiple stream channels are available, follow the recommendation provided in the "GenICam Standard Features Naming Convention". This involves a separate feature called the "Selector" used to indicate the index of the stream channel to configure.

29.4.1 Use Case #1: Continuous Acquisition and Display

For this test case, the application must be able to initialize the camera to start a free-running acquisition and to display the acquired images to the screen. The camera's factory default settings must ensure the camera shows a suitable live image when acquisition control is turned on without any further configuration.

To achieve this, the following actions are required:

1. Control the camera using GVCP
2. Create a stream channel using GVSP registers
3. Retrieve image characteristics through the XML camera description file
4. Allocate image buffers on the PC
5. Start the continuous acquisition through the stream channel

Step 1 and 2 require the use of the GigE Vision bootstrap registers.

Step 3 and 5 require the use of the standard features provided in the XML description file of the camera.

Step 4 does not require any interaction with the camera.

29.4.2 Use Case #2: Simplest GigE Vision Camera

For this test case, we consider the simplest camera possible. This is basically the equivalent of an RS-170 analog camera. This type of camera does not offer any of the following: trigger control, exposure control, analog gain control, etc. It is only a basic camera that acquire continuously at its nominal frame rate.

The idea is that mandatory features must be present on all cameras. Features not required on all cameras cannot be mandatory.

29.5 XML Description File Mandatory Features

[CR-506st] On top of the bootstrap registers that are required to control the camera and instantiate stream channels, all GigE Vision cameras MUST support the feature provided in Table 29-1 in their XML description files. [CR28-2st]

Table 29-1: GigE Vision Mandatory Features for XML Description File

Feature Name	Interface	Access	Units	Description
"Width"	IInteger	R/(W)	pixels	Width of the image output by the camera on the stream channel.
"Height"	IInteger	R/(W)	pixels	Height of the image output by the camera on the stream channel. For linescan sources, this represents the height of the frame created by combining consecutive lines together.
"PixelFormat"	IEnumeration	R/(W)	N/A	Format of the pixel output by the camera on the stream channel. Refer to the Pixel format section (p. 240)
"PayloadSize"	IInteger	R	Bytes	Maximum number of bytes transferred for each image on the stream channel, including any end-of-line, end-of-frame statistics or other stamp data. This is the maximum total size of data payload for a block. UDP and GVSP headers are not considered. Data leader and data trailer are not included. This is mainly used by the application software to determine size of image buffers to allocate (largest buffer possible for current mode of operation). For example, an image with no statistics or stamp

				data as PayloadSize equals to (width x height x pixel size) in bytes. It is strongly recommended to retrieve PayloadSize from the camera instead of relying on the above formula.
"GevSCPSPacketSize"	IInteger	R/W	bytes	Size of the GVSP packets to transmit during acquisition. This reflects the *packet_size* field of the SCPS register associated to this stream channel. This feature MUST explicitly state the minimum, maximum and increment value supported by the camera for GevSCPSPacketSize.
"AcquisitionMode"	IEnumeration	R/(W)	N/A	Used by application software to set the mode of acquisition. This feature indicates how image are sequenced out of the camera (continuously, single shot, multi-shot, …) Only a single value is mandatory for GigE Vision. {"Continuous"} *Continuous*: Configures the camera to stream an infinite sequence of images that must be stopped explicitly by the application. Note that the AcquistionMode can have more than this single entry. However, only this one is mandatory. The "AcquisitionStart" and "AcquisitionStop" features are used to control the acquisition state.
"AcquisitionStart"	ICommand	(R)/W	N/A	Start image acquisition using the specified acquisition mode.
"AcquisitionStop"	ICommand	(R)/W	N/A	Stop image acquisition using the specified acquisition mode.

Note: Access modes given between parentheses are optional.

This list is only intended for camera devices. Non-camera devices do not have to support those features.

It is not required that all GigE Vision bootstrap registers are exposed via the XML description file although it is recommended as per [O-414cd]. The XML description file is intended to expose features which are relevant to the user.

29.6 Width and Height Features

Width and Height represent the dimensions of the image coming out of the camera. This is basically the dimensions of the area of interest (AOI) that gets extracted from the sensor. Figure 29-1 shows the various features used to describe the AOI. Other features (HeightMax, WidthMax, OffsetY, OffsetX) are recommended names to use, as defined in the "GenICam Standard Features Naming Convention". But only Width and Height are mandatory for GigE Vision.

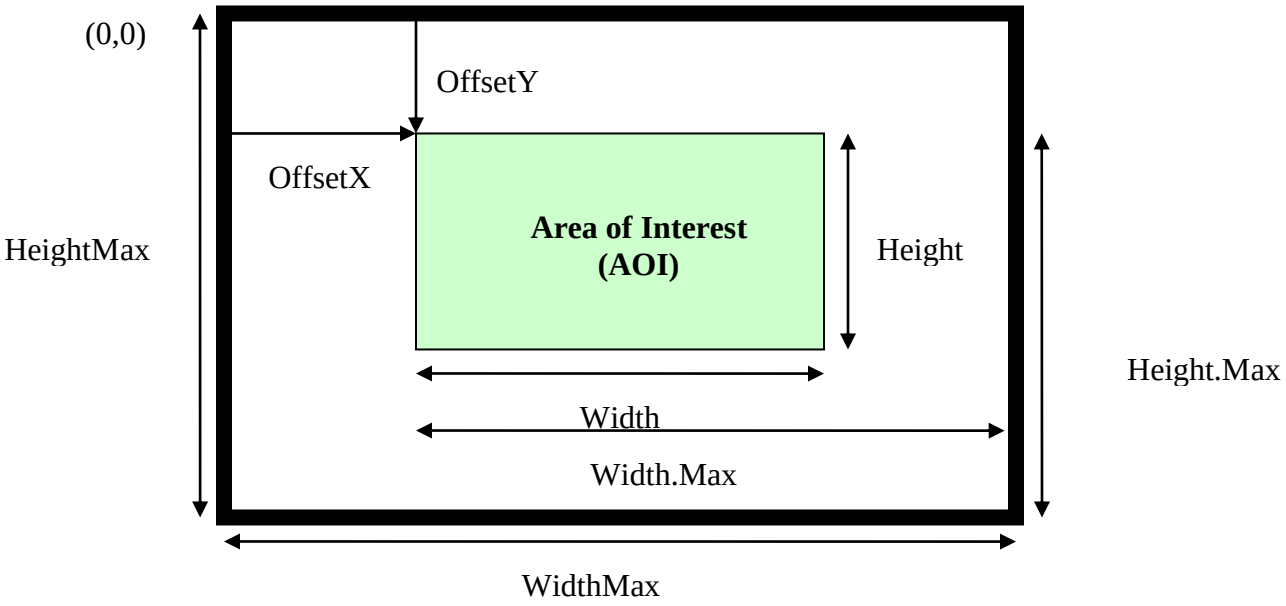


Figure 29-1: Width and Height Features

[CR-507st] Width and Height features MUST be represented using the IInteger interface of the GenICam specification (64-bit integer). [CR28-3st]

[CR-508st] Width and Height features MUST be expressed in pixels. [CR28-4st]

Width can take any value ranging from 1 up to MaxWidth.

[CR-509st] Width feature MUST take only positive values. [CR28-5st]

[CR-510st] Width feature MUST be readable. [CR28-6st]

Width might be writable if it is possible to configure the size of the AOI. Default value for Width is typically the sensor width. Some cameras might elect to define a coarser Width feature increment to avoid packing or grouping together pixels from different lines. For instance, when using Mono1p, the Width feature increment might be restricted to 8 to ensure that 8 pixels from the same line are packed together in a byte.

Height can take any value ranging from 1 up to the HeightMax.

[CR-511st] Height feature MUST take only positive values. [CR28-7st]

[CR-512st] Height feature MUST be readable. [CR28-8st]

Height might be writable if it is possible to configure the size of the AOI. Default value for Height is typically the sensor height.

If no other features than Width and Height are provided to describe the AOI, then the image starts at position (0, 0). This is equivalent to

- OffsetY = 0
- OffsetX = 0
- Width = WidthMax
- Height = HeightMax

When Width and Height are writable, they can be used to crop the image coming out of the sensor.

For linescan cameras, Height represents the number of lines combined together to form a frame. Each of these frames is encapsulated between a data leader and a data trailer packet, as defined by GVSP. Note it is perfectly legal to have frames of only one line (this is the traditional definition of linescan).

Examples

1) For a 640x480 8-bit monochrome areascan camera:
   - Width = 640 pixels
   - Height = 480 pixels

2) For a 640x480 RGB 8-bit packed monochrome areascan camera:
   - Width = 640 pixels
   - Height = 480 pixels

3) For a 4Kpixels linescan camera using a frame size of 64 lines:
   - Width = 4096 pixels
   - Height = 64 pixels

4) For a 4Kpixels linescan camera with frame of one line:
   - Width = 4096 pixels
   - Height = 1 pixels

29.7 PixelFormat Feature

PixelFormat provides the type of pixel output on the stream channel, as defined in GVSP (see p. 240).

[CR-513st] PixelFormat feature MUST be represented using the IEnumeration interface of the GenICam specification. [CR28-9st]

[CR-514st] A camera MUST use the name (and corresponding value) provided in Table 29-2 for PixelFormat enumeration. [CR28-11st]

Table 29-2: PixelFormat Strings

Pixel Format	Name	Value (hexadecimal)
GVSP_PIX_MONO1P	“Mono1p”	0x01010037
GVSP_PIX_MONO2P	“Mono2p”	0x01020038
GVSP_PIX_MONO4P	“Mono4p”	0x01040039
GVSP_PIX_MONO8	“Mono8”	0x01080001
GVSP_PIX_MONO8S	“Mono8s”	0x01080002
GVSP_PIX_MONO10	“Mono10”	0x01100003
GVSP_PIX_MONO10_PACKED	“Mono10Packed”	0x010C0004
GVSP_PIX_MONO12	“Mono12”	0x01100005
GVSP_PIX_MONO12_PACKED	“Mono12Packed”	0x010C0006
GVSP_PIX_MONO14	“Mono14”	0x01100025
GVSP_PIX_MONO16	“Mono16”	0x01100007
GVSP_PIX_BAYGR8	“BayerGR8”	0x01080008
GVSP_PIX_BAYRG8	“BayerRG8”	0x01080009
GVSP_PIX_BAYGB8	“BayerGB8”	0x0108000A
GVSP_PIX_BAYBG8	“BayerBG8”	0x0108000B
GVSP_PIX_BAYGR10	“BayerGR10”	0x0110000C
GVSP_PIX_BAYRG10	“BayerRG10”	0x0110000D
GVSP_PIX_BAYGB10	“BayerGB10”	0x0110000E
GVSP_PIX_BAYBG10	“BayerBG10”	0x0110000F
GVSP_PIX_BAYGR12	“BayerGR12”	0x01100010
GVSP_PIX_BAYRG12	“BayerRG12”	0x01100011
GVSP_PIX_BAYGB12	“BayerGB12”	0x01100012
GVSP_PIX_BAYBG12	“BayerBG12”	0x01100013
GVSP_PIX_BAYGR10_PACKED	“BayerGR10Packed”	0x010C0026
GVSP_PIX_BAYRG10_PACKED	“BayerRG10Packed”	0x010C0027
GVSP_PIX_BAYGB10_PACKED	“BayerGB10Packed”	0x010C0028
GVSP_PIX_BAYBG10_PACKED	“BayerBG10Packed”	0x010C0029
GVSP_PIX_BAYGR12_PACKED	“BayerGR12Packed”	0x010C002A
GVSP_PIX_BAYRG12_PACKED	“BayerRG12Packed”	0x010C002B
GVSP_PIX_BAYGB12_PACKED	“BayerGB12Packed”	0x010C002C
GVSP_PIX_BAYBG12_PACKED	“BayerBG12Packed”	0x010C002D
GVSP_PIX_BAYGR16	“BayerGR16”	0x0110002E
GVSP_PIX_BAYRG16	“BayerRG16”	0x0110002F

Pixel Format	Name	Value (hexadecimal)
GVSP_PIX_BAYGB16	“BayerGB16”	0x01100030
GVSP_PIX_BAYBG16	“BayerBG16”	0x01100031
GVSP_PIX_RGB8	“RGB8”	0x02180014
GVSP_PIX_BGR8	“BGR8”	0x02180015
GVSP_PIX_RGBA8	“RGBa8”	0x02200016
GVSP_PIX_BGRA8	“BGRa8”	0x02200017
GVSP_PIX_RGB10	“RGB10”	0x02300018
GVSP_PIX_BGR10	“BGR10”	0x02300019
GVSP_PIX_RGB12	“RGB12”	0x0230001A
GVSP_PIX_BGR12	“BGR12”	0x0230001B
GVSP_PIX_RGB16	“RGB16”	0x02300033
GVSP_PIX_RGB10V1_PACKED	“RGB10V1Packed”	0x0220001C
GVSP_PIX_RGB10P32	“RGB10p32”	0x0220001D
GVSP_PIX_RGB12V1_PACKED	“RGB12V1Packed”	0x02240034
GVSP_PIX_RGB565P	“RGB565p”	0x02100035
GVSP_PIX_BGR565P	“BGR565p”	0x02100036
GVSP_PIX_YUV411_8_UYYVYY	“YUV411_8_UYYVYY”	0x020C001E
GVSP_PIX_YUV422_8_UYVY	“YUV422_8_UYVY”	0x0210001F
GVSP_PIX_YUV422_8	“YUV422_8”	0x02100032
GVSP_PIX_YUV8_UYV	“YUV8_UYV”	0x02180020
GVSP_PIX_YCBCR8_CBYCR	“YCbCr8_CbYCr”	0x0218003A
GVSP_PIX_YCBCR422_8	“YCbCr422_8”	0x0210003B
GVSP_PIX_YCBCR422_8_CBYCRY	“YCbCr422_8_CbYCrY”	0x02100043
GVSP_PIX_YCBCR411_8_CBYYCRYY	“YCbCr411_8_CbYYCrYY”	0x020C003C
GVSP_PIX_YCBCR601_8_CBYCR	“YCbCr601_8_CbYCr”	0x0218003D
GVSP_PIX_YCBCR601_422_8	“YCbCr601_422_8”	0x0210003E
GVSP_PIX_YCBCR601_422_8_CBYCRY	“YCbCr601_422_8_CbYCrY”	0x02100044
GVSP_PIX_YCBCR601_411_8_CBYYCRYY	“YCbCr601_411_8_CbYYCrYY”	0x020C003F
GVSP_PIX_YCBCR709_8_CBYCR	“YCbCr709_8_CbYCr”	0x02180040
GVSP_PIX_YCBCR709_422_8	“YCbCr709_422_8”	0x02100041
GVSP_PIX_YCBCR709_422_8_CBYCRY	“YCbCr709_422_8_CbYCrY”	0x02100045
GVSP_PIX_YCBCR709_411_8_CBYYCRYY	“YCbCr709_411_8_CbYYCrYY”	0x020C0042
GVSP_PIX_RGB8_PLANAR	“RGB8_Planar”	0x02180021

Pixel Format	Name	Value (hexadecimal)
GVSP_PIX_RGB10_PLANAR	“RGB10_Planar”	0x02300022
GVSP_PIX_RGB12_PLANAR	“RGB12_Planar”	0x02300023
GVSP_PIX_RGB16_PLANAR	“RGB16_Planar”	0x02300024

The Pixel Format Naming Convention is used as a reference in naming those pixels.

Note that Bayer pixel format names represent the first 2 pixels of the Bayer mosaic of the full image (no ROI).

Examples

1) For a 640x480 8-bit monochrome areascan camera:
- PixelFormat = “Mono8”

2) For a 4Kpixels RGB-alpha 8-bit per component, unpacked in 32 bits, color linescan camera:
- PixelFormat = “RGBa8”

29.8 PayloadSize Feature

PayloadSize is a read-only feature representing the maximum number of data bytes sent for one block ID in the data payload packets on the stream channel. This does not include IP, UDP, GVSP headers or the data leader and data trailer packets. But it does include any data that can be appended to the image itself. The primary usage of PayloadSize is to provide an easy to retrieve buffer size to allocate for data transfer on the image stream channel.

In the case where data is of variable size, then the maximum possible value (worst-case value) must be provided by PayloadSize.

[CR-515st] PayloadSize feature MUST be represented using the IInteger interface of the GenICam specification (64-bit integer). [CR28-12st]

[CR-516st] PayloadSize feature MUST be expressed in bytes (8-bit value). [CR28-13st]

[CR-517st] PayloadSize feature MUST only take positive values. [CR28-14st]

The typical value for PayloadSize for a camera is (Width x Height x Pixel Size) when no extra information is appended to the image. If the stream channel concatenates other information to the image (such as end-of-frame statistics or padding), then the PC buffer required to store the data might by larger than the size of the image.

Examples

1) For a 640x480 8-bit monochrome areascan camera:
- PayloadSize = 640 x 480 = 307200 bytes

2) For a 4Kpixels RGB-alpha 8-bit per component, unpacked into 32 bits, color linescan camera with frames of 64 lines:

- PayloadSize = 4K pixels x 4 bytes/pixels x 64 lines = 1048576 bytes

29.9 GevSCPSPacketSize

The GevSCPSPacketSize feature helps the application determines the value of packet size supported by the camera when streaming data. It essentially reflects the *packet_size* field of the SCPS register, but provides additional information about the range of value (minimal, maximal and increment) to use when searching for optimal packet size. Note that this range represents the limitation imposed by the camera and does not account for any limitation introduced by other network equipment, such as Ethernet switches. As such, it is recommended to use a Test Packet to confirm the overall system supports the requested packet size.

[CR-518st] GevSCPSPacketSize feature MUST be represented using the IInteger interface of the GenICam specification (64-bit integer).

[CR-519st] GevSCPSPacketSize feature MUST be expressed in bytes (8-bit value).

[CR-520st] GevSCPSPacketSize feature MUST only take integer values in the range from 46 to 65535. A camera might impose a smaller range in that range.

Note: The minimum value for GevSCPSPacketSize comes from the minimum Ethernet frame size of 64 bytes from which we subtract 6 bytes for the destination MAC, 6 bytes for the source MAC, 2 bytes for the Ethertype field and 4 bytes for the frame check sequence (32-bit CRC). This leaves 46 bytes of data.

[CR-521st] The value read or set through the GevSCPSPacketSize feature MUST be reflected in the *packet_size* field of the corresponding SCPS bootstrap register.

[CR-522st] A camera is allowed to round the value written to the GevSCPSPacketSize feature to the nearest supported value. But the camera MUST NOT change GevSCPSPacketSize without having an application directly writing into this feature.

For instance, it is allowed for the camera to round a packet size value to a multiple of 4 bytes, but it is not allowed for the camera to change the packet size feature if the application changes the data format (such as the image width or pixel format features). Generally, the GevSCPSPacketSize feature would provide minimum, maximum and increment such that rounding is not necessary.

29.10 AcquisitionMode Feature

The AcquistionMode feature is used to determine the sequencing of images during acquisition. This typically refers to the number of images captured after the acquisition has been started. It could represent uninterrupted acquisition or the acquisition of a pre-determined number of frames. Note that this is totally independent from the stream channel being opened or not.

[CR-523st] AcquisitionMode feature MUST be readable. [CR28-15st]

[CR-524st] When AcquisitionMode provides multiple values in its enumeration, then it has to be writable. [CR28-16st]

[CR-525st] AcquisitionMode feature MUST be represented by the IEnumeration interface of the GenICam specification. It is mandatory to list at least the following mode of acquisition:

1. "Continuous"

By default, Acquisition mode MUST take the "Continuous" state. [CR28-17st]

Note that other acquisition modes are recommended by the "GenICam Standards Feature Naming Convention". The string used to represent a given mode of acquisition is provided in Table 29-3. The value associated to each string is specific to the camera.

Table 29-3: AcquisitionMode Strings

Mode	String
Uninterrupted acquisition	"Continuous"

AcquisitionMode should only be changed when image acquisition is stopped. Result of changing AcquisitionMode after issuing an AcquisitionStart command (but before an AcquisitionStop command) is not defined by this specification. This feature should only be changed after an AcquisitionStop command has been fired.

[CR-526st] When AcquisitionMode is set to "Continuous", image acquisition is initiated by the "AcquisitionStart" command and will execute until the "AcquisitionStop" command is issued by the application. It is allowed to have other commands (such as an optional AcquisitionAbort command) end the acquisition. [CR28-18st]

29.11 AcquisitionStart Feature

The AcquisitionStart command begins image acquisition using the mode specified by AcquisitionMode. It is expected that image acquisition starts as soon as possible after the AcquisitionStart command has been issued.

[CR-527st] AcquisitionStart feature MUST be realized using the ICommand feature of the GenICam specification. [CR28-19st]

Note the stream channel must be setup through the bootstrap registers and the other camera features must be initialized prior to start an acquisition.

[CR-528st] Re-issuing an AcquisitionStart command after it has already been initiated, but before an AcquisitionStop command, MUST not affect the image acquisition status. Therefore, the camera must remain with acquisition active. [CR28-21st]

29.12 AcquisitionStop Feature

The AcquisitionStop command terminates image acquisition after the current frame is completed.

[CR-529st] AcquisitionStop feature MUST be realized using the ICommand feature of the GenICam specification. [CR28-22st]

[CR-530st] Image acquisition MUST stop as soon as possible on a frame boundary (i.e. after the data trailer) when issuing the AcquisitionStop command. [CR28-23st]

[CR-531st] Re-issuing an AcquisitionStop command after it has already been initiated, but before an AcquisitionStart command, MUST not affect the image acquisition status. Therefore, the camera must remain with acquisition inactive. [CR28-24st]

29.13 Link to Naming Convention

This section only described the mandatory features provided in the XML camera description file. Other recommended features are covered in the "GenICam Standard Feature Naming Convention" (available from http://www.genicam.org). This naming convention should be respected when implementing features not described in this section.

Proposal for new recommended feature names shall be addressed to the GenICam committee.

30 Appendix 1 –Requirements Reference Tables

Table 30-1 : Requirements and Objectives Reference Table

Req.	Version 1.x cross-reference	Introduced in version
[R-001cd]	[R12-1cd]	1.0
[R-002cd]	-	2.0
[R-003cd]	[R12-2cd]	1.0
[CR-004cd]	[CR3-3cd]	1.0
[CR-005cd]	[CR12-3cd]	1.0
[R-006cd]	[R12-5cd]	1.0
[CR-007cd]	[CR10-7s]	1.0
[CR-008cd]	-	2.0
[CO-009cd]	-	2.0
[CR-010c]	-	2.0
[CO-011st]	-	2.0
[CO-012cd]	-	2.0
[R-013cd]	[R3-1cd]	1.0
[R-014cd]	[R3-2cd]	1.0
[R-015cd]	[R3-5cd]	1.0
[R-016cd]	[R3-6cd]	1.0
[R-017cd]	[R3-7cd]	1.0
[CR-018cd]	[CR3-8cd]	1.0
[CR-019cd]	[CR3-9cd]	1.0
[CO-020cd]	[CO3-11cd]	1.0
[R-021cd]	[R3-12cd]	1.0
[O-022cd]	[O3-13cd]	1.0
[CR-023cd]	[CR3-14cd]	1.0
[CR-024cd]	[CR3-15cd]	1.0
[CR-025cd]	[CR3-16cd]	1.0
[O-026cd]	[O3-18cd]	1.0
[O-027cd]	[O3-19cd]	1.0
[R-028cd]	[R3-20cd]	1.0
[R-029cd]	[R3-21cd]	1.0
[R-030cd]	[R4-1cd]	1.0
[R-031cd]	[R4-2cd]	1.0

[R-032cd]	[R4-5cd]	1.0
[R-033cd]	[R4-6cd]	1.0
[O-034cd]	[O4-7cd]	1.0
[O-035cd]	-	2.0
[O-036cd]	-	2.0
[CR-037cd]	-	2.0
[CR-038cd]	-	2.0
[O-039ca]	[O5-1ca]	1.0
[R-040c]	[R7-1c]	1.0
[R-041c]	[R7-2c]	1.0
[R-042c]	[R7-3c]	1.0
[R-043c]	[R7-4c]	1.0
[R-044c]	[R7-6c]	1.0
[R-045c]	[R7-7c]	1.0
[O-046ca]	[O7-8ca]	1.0
[R-047ca]	[R7-9ca]	1.0
[O-048ca]	[O7-10ca]	1.0
[O-049ca]	[O7-11ca]	1.0
[R-050ca]	[R7-12ca]	1.0
[R-051cd]	[R7-13cd]	1.0
[R-052cd]	[R7-14cd]	1.0
[CR-053cd]	-	2.0
~~[R-054ca]~~	~~[R7-18ca]~~	~~1.0~~
[R-055ca]	[R7-19ca]	1.0
[R-056cd]	-	2.0
[R-057cd]	[R9-1cd]	1.0
[R-058ca]	[R8-3ca]	1.0
[O-059cd]	[O9-2cd]	1.0
[R-060ca]	[R9-3ca]	1.0
[R-061cd]	[R9-4cd]	1.0
[R-062cd]	[R9-5cd]	1.0
[R-063cd]	[R9-6cd]	1.0
[CO-064cd]	[CO9-7cd]	1.0
[CR-065cd]	[CR9-8cd]	1.0
[CO-066cd]	[CO9-17cd]	1.2
[CR-067cd]	[CR9-18cd]	1.2
[R-068cd]	[R9-9cd]	1.0

[R-069cd]	[R9-10cd]	1.0
[R-070cd]	[R9-11cd]	1.0
[R-071ca]	[R27-30ca]	1.0
[R-072cd]	[R9-12cd]	1.0
[O-073ca]	[O9-13ca]	1.0
[R-074cd]	[R9-14cd]	1.0
[R-075cd]	[R9-15cd]	1.0
[R-076cd]	[R9-16cd]	1.0
[CR-077cd]	-	2.0
[CR-078cd]	-	2.0
[R-079s]	[R8-1s]	1.0
[R-080s]	[R10-2s]	1.0
[R-081s]	-	2.0
[R-082s]	-	2.0
[CR-083s]	-	2.0
[CO-084s]	-	2.0
[CO-085st]	-	2.0
[O-086st]	[O10-3st]	1.0
[R-087ca]	[R27-36ca]	1.0
[R-088st]	-	2.0
[CR-089cd]	[CR27-45cd]	1.1
[CR-090cd]	[CR27-46cd]	1.1
[R-091ca]	[R10-4ca]	1.0
[R-092st]	[R10-5st]	1.0
[R-093st]	[R10-6st]	1.0
[CR-094st]	[CR10-8st]	1.1
[CR-095ca]	[CR10-9ca]	1.1
[R-096cd]	[R8-2cd]	1.0
[CR-097cd]	[CR11-1cd]	1.0
[CO-098cd]	[CO11-2cd]	1.0
[CR-099ca]	[CR27-31ca]	1.0
[CR-100ca]	[CR11-7ca]	1.1
[CR-101cd]	[CR27-33cd]	1.0
[CR-102cd]	[CR27-35cd]	1.0
[CR-103cd]	[CR27-34cd]	1.0
[CR-104cd]	[CR27-43cd]	1.1
[CR-105cd]	[CR27-44cd]	1.1

[CR-106ca]	[CR11-3ca]	1.0
[CR-107cd]	[CR11-4cd]	1.0
[CR-108ca]	[CR11-5ca]	1.0
[CR-109cd]	[CR11-6cd]	1.0
[CR-110ca]	[CR11-8ca]	1.1
[R-111c]	[R12-6c]	1.0
[R-112cd]	[R12-7cd]	1.0
[R-113st]	[R12-8st]	1.0
[CR-114sr]	[CR12-11sr]	1.2
[R-115ca]	[R12-9ca]	1.0
[CR-116c]	[CR12-10c]	1.0
[R-117cd]	[R17-1cd]	1.0
[R-118ca]	[R17-3ca]	1.0
[R-119cd]	[R17-5cd]	1.0
[O-120ca]	[O17-6ca]	1.0
[R-121ca]	[R17-4ca]	1.0
[R-122ca]	-	2.0
[CR-123cd]	-	2.0
[CR-124cd]	-	2.0
[CR-125c]	-	2.0
[CR-126c]	-	2.0
[CR-127cd]	-	2.0
[CR-128cd]	-	2.0
[R-129c]	[R13-1c]	1.0
[O-130c]	[O13-2c]	1.0
[R-131c]	[R13-3c]	1.0
[R-132c]	[R13-4c]	1.0
[R-133c]	-	2.0
[O-134c]	[O13-5c]	1.0
[R-135c]	[R13-6c]	1.0
[O-136c]	[O13-7c]	1.0
[R-137c]	[R13-9c]	1.0
[R-138cd]	[R14-1cd]	1.0
[R-139cd]	[R14-2cd]	1.0
[R-140c]	[R14-3c]	1.0
[R-141ca]	[R14-62ca]	1.1
[R-142cd]	[R14-5cd]	1.0

[O-143cd]	-	2.0
[O-144cd]	[O14-6cd]	1.0
[R-145c]	[R14-7c]	1.0
[R-146cd]	[R14-8cd]	1.0
[R-147cd]	[R14-9cd]	1.0
[CR-148ca]	[CR14-10ca]	1.0
[R-149cd]	[R14-11cd]	1.0
[R-150cd]	[R14-12cd]	1.0
[R-151cd]	[R14-13cd]	1.0
[R-152cd]	[R14-14cd]	1.0
[R-153cd]	[R14-15cd]	1.0
[CR-154ca]	[CR14-76ca]	1.1
[O-155cd]	-	2.0
[R-156cd]	[R14-16cd]	1.0
[R-157cd]	[R14-17cd]	1.0
[R-158cd]	[R14-18cd]	1.0
[CR-159cd]	[CR14-19cd]	1.0
[CR-160cd]	[CR14-20cd]	1.0
[R-161cd]	[R14-21cd]	1.0
[R-162cd]	[R14-22cd]	1.0
[R-163c]	[R14-23c]	1.0
[R-164c]	[R14-24c]	1.0
[R-165cd]	[R14-25cd]	1.0
[R-166cd]	[R14-26cd]	1.0
[R-167cd]	[R14-27cd]	1.0
[CR-168cd]	[CR14-28cd]	1.0
[CR-169cd]	[CR14-29cd]	1.0
[R-170cd]	[R14-30cd]	1.0
[R-171cd]	[R14-31cd]	1.0
[R-172cd]	[R14-32cd]	1.0
[R-173c]	[R14-33c]	1.0
[R-174c]	[R14-34c]	1.0
[R-175cd]	[R14-35cd]	1.0
[R-176cd]	[R14-36cd]	1.0
[R-177c]	[R14-37c]	1.0
[R-178cd]	-	2.0
[R-179c]	[R14-38c]	1.0

[R-180cd]	[R14-39cd]	1.0
[R-181cd]	[R14-40cd]	1.0
[R-182c]	[R14-41c]	1.0
[R-183c]	[R14-42c]	1.0
[R-184cd]	[R14-43cd]	1.0
[R-185cd]	-	2.0
[O-186ca]	[O14-44ca]	1.0
[R-187c]	[R14-45c]	1.0
[R-188cd]	-	2.0
[R-189c]	[R14-46c]	1.0
[R-190cd]	[R14-47cd]	1.0
[R-191cd]	[R14-48cd]	1.0
[R-192c]	[R14-49c]	1.0
[R-193c]	[R14-50c]	1.0
[R-194cd]	[R14-51cd]	1.0
[R-195cd]	[R14-52cd]	1.0
[O-196cd]	[O14-53cd]	1.0
[R-197ca]	-	2.0
[CR-198ca]	[CR14-55ca]	1.0
[CR-199cd]	[CR14-56cd]	1.0
[CR-200c]	[CR14-59c]	1.0
[CR-201cd]	[CR14-60cd]	1.0
[CO-202st]	[CO14-63cd]	1.1
[CR-203st]	[CR14-61cd]	1.0
[CR-204st]	-	2.0
[CR-205st]	-	2.0
[CO-206st]	-	2.0
[CR-207sr]	-	2.0
[CR-208st]	-	2.0
[CR-209cd]	[CR14-64cd]	1.1
[CR-210cd]	[CR14-65cd]	1.1
[CR-211cd]	[CR14-66cd]	1.1
[CR-212cd]	[CR14-67cd]	1.1
[CR-213c]	[CR14-68c]	1.1
[CR-214cd]	[CR14-69cd]	1.1
[CR-215cd]	[CR14-70cd]	1.1
[O-216ca]	[O14-71ca]	1.1

[O-217ca]	-	2.0
[CR-218cd]	[CR14-72cd]	1.1
[CR-219c]	[CR14-73c]	1.1
[CR-220cd]	-	2.0
[CR-221cd]	-	2.0
[CR-222cd]	[CR14-74cd]	1.1
[CR-223cd]	-	2.0
[CR-224c]	[CR14-75c]	1.1
[O-225ca]	[O15-15ca]	1.2
[CR-226cd]	[CR15-1cd]	1.0
[CO-227cd]	[CO15-2cd]	1.0
[O-228cd]	[O15-3cd]	1.0
[CR-229cd]	[CR15-5cd]	1.0
[CO-230cd]	[CO15-6cd]	1.0
[CR-231c]	[CR15-7c]	1.0
[CR-232c]	[CR15-8c]	1.0
[CR-233cd]	[CR15-10cd]	1.0
[CR-234cd]	[CR15-11cd]	1.0
[CO-235cd]	[CO15-12cd]	1.0
[CR-236c]	[CR15-13c]	1.0
[CR-237c]	[CR15-14c]	1.0
[R-238c]	[R16-1c]	1.0
[O-239cd]	[O18-1cd]	1.0
[R-240cd]	[R18-2cd]	1.0
[O-241st]	[O18-3st]	1.2
[R-242st]	[R18-4st]	1.2
[CR-243cd]	[CR19-1cd]	1.0
[R-244cd]	[R20-1cd]	1.0
[R-245s]	[R22-1s]	1.0
[R-246s]	[R22-2s]	1.0
[R-247sr]	[R22-3sr]	1.0
[R-248st]	[R22-4st]	1.0
[R-249st]	-	2.0
[R-250st]	-	2.0
[R-251st]	[R23-2st]	1.0
[CR-252st]	[CR24-10st]	1.0
[CR-253st]	[CR24-11st]	1.0

[CR-254st]	-	2.0
[R-255s]	-	2.0
[R-256s]	[R23-1s]	1.0
[R-257s]	[R24-2s]	1.0
[R-258s]	[R24-1s]	1.0
[R-259st]	-	2.0
[R-260st]	[R24-3st]	1.0
[R-261st]	[R24-4st]	1.0
[R-262s]	-	2.0
[R-263st]	[R24-12st]	1.0
[R-264st]	-	2.0
[R-265s]	-	2.0
[R-266st]	[R24-18st]	1.0
[R-267st]	[R24-19st]	1.0
[R-268st]	[R24-20st]	1.0
[R-269s]	[R24-21s]	1.0
[CR-270st]	-	2.0
[CR-271st]	-	2.0
[CR-272st]	-	2.0
[CR-273st]	-	2.0
[CR-274st]	-	2.0
[CR-275s]	-	2.0
[CR-276s]	[CR23-12s]	1.1
[CR-277s]	[CR23-13s]	1.1
[CR-278st]	[CR23-9st]	1.0
[CO-279st]	[CO23-10st]	1.0
[CR-280st]	[CR24-29st]	1.1
[CO-281sr]	[CO24-30sr]	1.1
[R-282s]	[R24-28st]	1.0
[CR-283st]	-	2.0
[CR-284st]	[CR23-3st]	1.0
[CR-285st]	[CR23-4st]	1.0
[CR-286s]	[CR23-5s]	1.0
[CR-287st]	[CR23-6st]	1.0
[CR-288s]	[CR24-5s]	1.0
[CR-289s]	[CR24-13s]	1.0
[CR-290s]	[CR24-22s]	1.0

[CR-291s]	-	2.0
[CR-292st]	-	2.0
[CR-293st]	[CR23-7st]	1.0
[CR-294s]	[CR24-6s]	1.0
[CR-295s]	[CR24-14s]	1.0
[CR-296s]	[CR24-23s]	1.0
[CR-297s]	-	2.0
[CR-298s]	-	2.0
[CR-299st]	[CR23-8st]	1.0
[CR-300s]	[CR24-7s]	1.0
[CR-301s]	[CR24-15s]	1.0
[CR-302s]	[CR24-24s]	1.0
[CR-303s]	-	2.0
[CR-304st]	-	2.0
[CR-305s]	[CR24-8s]	1.0
[CR-306s]	[CR24-16s]	1.0
[CR-307s]	[CR24-25s]	1.0
[CR-308s]	-	2.0
[CR-309s]	[CR23-14s]	1.2
[CR-310s]	[CR24-31s]	1.2
[CR-311s]	[CR24-16s]	1.1
[CR-312s]	[CR24-32s]	1.2
[CR-313st]	-	2.0
[CR-314s]	-	2.0
[CR-315s]	-	2.0
[CR-316s]	-	2.0
[CR-317s]	-	2.0
[CR-318st]	-	2.0
[CR-319s]	-	2.0
[CR-320s]	-	2.0
[CR-321s]	-	2.0
[CR-322s]	-	2.0
[CR-323st]	-	2.0
[CR-324s]	-	2.0
[CR-325s]	-	2.0
[CR-326s]	-	2.0
[CR-327s]	-	2.0

[CR-328st]	-	2.0
[CR-329st]	-	2.0
[CR-330st]	-	2.0
[CR-331st]	-	2.0
[CR-332st]	-	2.0
[CR-333st]	-	2.0
[CR-334st]	-	2.0
[CR-335st]	-	2.0
[CR-336s]	-	2.0
[CR-337s]	-	2.0
[CR-338s]	-	2.0
[CR-339s]	[CR24-9s]	1.0
[CR-340s]	[CR24-17s]	1.0
[CR-341s]	[CR24-26s]	1.0
[CR-342s]	-	2.0
[O-343st]	[O25-1st]	1.0
[R-344st]	[R25-2st]	1.0
[O-345s]	[O25-3s]	1.0
[CR-346s]	-	2.0
[CR-347s]	-	2.0
[CR-348s]	-	2.0
[CR-349s]	[CR25-4s]	1.0
[CR-350s]	[CR25-5s]	1.0
[CR-351s]	[CR25-6s]	1.0
[CR-352s]	[CR25-7s]	1.0
[CR-353s]	[CR25-8s]	1.0
[CR-354s]	[CR25-9s]	1.0
[CR-355s]	[CR25-41s]	1.1
[CR-356s]	[CR25-10s]	1.0
[CR-357st]	[CR25-11st]	1.0
[CR-358s]	[CR25-12s]	1.0
[CR-359s]	[CR25-13s]	1.0
[CR-360s]	[CR25-14s]	1.0
[CR-361s]	[CR25-15s]	1.0
[CR-362s]	[CR25-16s]	1.0
[CR-363s]	[CR25-17s]	1.0
[CR-364s]	[CR25-18s]	1.0

[CR-365s]	[CR25-19s]	1.0
[CR-366s]	[CR25-20s]	1.0
[CR-367s]	[CR25-21s]	1.0
[CR-368s]	[CR25-22s]	1.0
[CR-369s]	[CR25-23s]	1.0
[CR-370s]	[CR25-42s]	1.1
[CR-371s]	[CR25-43s]	1.1
[CR-372s]	[CR25-44s]	1.1
[CR-373s]	[CR25-45s]	1.1
[CR-374s]	[CR25-46s]	1.1
[CR-375s]	[CR25-47s]	1.1
[CR-376s]	[CR25-48s]	1.1
[CR-377s]	[CR25-49s]	1.1
[CR-378s]	[CR25-50s]	1.1
[CR-379s]	[CR25-51s]	1.1
[CR-380s]	[CR25-52s]	1.1
[CR-381s]	[CR25-53s]	1.1
[CR-382s]	[CR25-24s]	1.0
[CR-383s]	[CR25-25s]	1.0
[CR-384s]	[CR25-26s]	1.0
[CR-385s]	[CR25-27s]	1.0
[CR-386s]	[CR25-28s]	1.0
[CR-387s]	[CR25-29s]	1.0
[CR-388s]	[CR25-30s]	1.0
[CR-389s]	[CR25-31s]	1.0
[CR-390s]	[CR-25-54s]	1.1
[CR-391s]	[CR25-32s]	1.0
[CR-392s]	[CR25-33s]	1.0
[CR-393s]	[CR25-55s]	1.1
[CR-394s]	[CR25-57s]	1.2
[CR-395s]	[CR25-58s]	1.2
[CR-396s]	[CR25-34s]	1.0
[CR-397s]	[CR25-35s]	1.0
[CR-398s]	[CR25-56s]	1.1
[CR-399s]	[CR25-36s]	1.0
[CR-400s]	-	2.0
[CR-401s]	-	2.0

[CR-402s]	-	2.0
[CR-403s]	-	2.0
[CR-404s]	-	2.0
[CR-405s]	-	2.0
[CR-406s]	-	2.0
[CR-407s]	-	2.0
[CR-408s]	-	2.0
[CR-409s]	[CR25-37s]	1.0
[CR-410s]	[CR25-38s]	1.0
[CR-411s]	[CR25-39s]	1.0
[CR-412s]	[CR25-40s]	1.0
[R-413cd]	[R27-1cd]	1.0
[O-414cd]	[O27-37cd]	1.1
[R-415cd]	[R27-2cd]	1.0
[R-416cd]	-	2.0
[R-417ca]	[R27-4ca]	1.0
[R-418cd]	[R27-5cd]	1.0
[R-419cd]	[R27-6cd]	1.0
[R-420ca]	-	2.0
[R-421cd]	[R27-7cd]	1.0
[R-422cd]	-	2.0
[R-423cd]	[R27-8cd]	1.0
[O-424ca]	[O27-9ca]	1.0
[CR-425cd]	[CR27-16cd]	1.0
[CR-426cd]	[CR27-17cd]	1.0
[O-427cd]	-	2.0
[O-428cd]	-	2.0
[R-429cd]	-	2.0
[R-430cd]	-	2.0
[R-431cd]	[R27-10cd]	1.0
[R-432cd]	[R27-11cd]	1.0
[R-433cd]	[R27-12cd]	1.0
[R-434cd]	[R27-13cd]	1.0
[R-435cd]	[R27-14cd]	1.0
[R-436cd]	[R27-15cd]	1.0
[R-437cd]	-	2.0
[R-438cd]	-	2.0

[R-439cd]	-	2.0
[R-440cd]	-	2.0
[O-441cd]	-	2.0
[O-442cd]	-	2.0
[CR-443cd]	[CR27-18cd]	1.0
[R-444cd]	-	2.0
[R-445cd]	-	2.0
[CR-446cd]	-	2.0
[CR-447cd]	[CR27-21cd]	1.0
[CR-448cd]	[CR27-22cd]	1.0
[CR-449cd]	[CR27-23cd]	1.0
[CR-450cd]	[CR27-38cd]	1.1
[R-451cd]	-	2.0
[R-452cd]	-	2.0
[CR-453cd]	-	2.0
[CR-454cd]	-	2.0
[R-455cd]	-	2.0
[R-456cd]	-	2.0
[R-457cd]	-	2.0
[R-458cd]	-	2.0
[R-459cd]	-	2.0
[O-460cd]	[O27-24cd]	1.0
[CR-461cd]	[O27-25cd]	1.0
[CR-462cd]	-	2.0
[CR-463ca]	[CR27-26ca]	1.0
[CR-464cd]	[CR27-27cd]	1.0
[CR-465cd]	-	2.0
[CR-466ca]	[CR27-28ca]	1.0
[CR-467cd]	[CR27-29cd]	1.0
[O-468cd]	-	2.0
[CR-469cd]	[CR27-41cd]	1.1
[CR-470cd]	[CR27-42cd]	1.1
[O-471cd]	-	2.0
[R-472cd]	-	2.0
[CR-473cd]	-	2.0
[O-474cd]	-	2.0
[R-475cd]	-	2.0

[R-476cd]	-	2.0
[CR-477cd]	-	2.0
[CR-478cd]	-	2.0
[R-479cd]	-	2.0
[O-480cd]	-	2.0
[O-481cd]	-	2.0
[CR-482cd]	-	2.0
[CR-483cd]	-	2.0
[CR-484cd]	-	2.0
[CR-485cd]	[CR27-32cd]	1.0
[CR-486cd]	-	2.0
[CO-487cd]	-	2.0
[CR-488cd]	-	2.0
[CR-489cd]	-	2.0
[R-490st]	-	2.0
[CR-491cd]	-	2.0
[CR-492cd]	-	2.0
[CO-493cd]	-	2.0
[CO-494cd]	-	2.0
[CO-495cd]	-	2.0
[CR-496cd]	-	2.0
[CR-497cd]	-	2.0
[O-498cd]	-	2.0
[R-499ca]	-	2.0
[CR-500cd]	[CR27-47cd]	1.1
[CR-501cd]	[CR27-48cd]	1.1
[CR-502cd]	[CR27-49cd]	1.1
[CR-503cd]	-	2.0
[CR-504cd]	-	2.0
[R-505cd]	[R28-1cd]	1.0
[CR-506st]	[CR28-2st]	1.0
[CR-507st]	[CR28-3st]	1.0
[CR-508st]	[CR28-4st]	1.0
[CR-509st]	[CR28-5st]	1.0
[CR-510st]	[CR28-6st]	1.0
[CR-511st]	[CR28-7st]	1.0
[CR-512st]	[CR28-8st]	1.0

[CR-513st]	[CR28-9st]	1.0
[CR-514st]	[CR28-11st]	1.0
[CR-515st]	[CR28-12st]	1.0
[CR-516st]	[CR28-13st]	1.0
[CR-517st]	[CR28-14st]	1.0
[CR-518st]	-	2.0
[CR-519st]	-	2.0
[CR-520st]	-	2.0
[CR-521st]	-	2.0
[CR-522st]	-	2.0
[CR-523st]	[CR28-15st]	1.0
[CR-524st]	[CR28-16st]	1.0
[CR-525st]	[CR28-17st]	1.0
[CR-526st]	[CR28-18st]	1.0
[CR-527st]	[CR28-19st]	1.0
[CR-528st]	[CR28-21st]	1.0
[CR-529st]	[CR28-22st]	1.0
[CR-530st]	[CR28-23st]	1.0
[CR-531st]	[CR28-24st]	1.0
[CR-532s]		2.0.03
[CR-533s]		2.0.03
[CR-534s]		2.0.03

Table 30-2 : Deprecated Requirements and Objectives Table

Req.	Introduced in version	Deprecated in version
[CO3-10d]	1.0	1.1
[CO3-17d]	1.0	1.1
[O3-22d]	1.0	1.1
[R4-3d]	1.0	2.0
[O4-4d]	1.0	2.0
[O7-5]	1.0	2.0
[O7-15d]	1.0	1.2
[R7-16]	1.0	1.2
[R7-17a]	1.0	1.2
[R10-1s]	1.0	2.0
[O12-4d]	1.0	2.0

[O13-8a]	1.0	1.1
[R14-4d]	1.0	2.0
[O14-54a]	1.0	2.0
[CO14-57d]	1.0	2.0
[R14-58d]	1.0	2.0
[O15-4a]	1.0	1.2
[O15-9a]	1.0	1.2
[CR17-2d]	1.0	2.0
[CO23-11d]	1.0	1.1
[R24-27st]	1.0	2.0
[O27-3a]	1.0	2.0
[R27-19d]	1.0	2.0
[R27-20d]	1.0	2.0
[CR27-39cd]	1.1	2.0
[CR27-40cd]	1.1	2.0
[CR28-10d]	1.0	1.1
[CR28-20d]	1.0	1.2
[R-054ca]	1.0	2.0.03

31 Appendix 2 – Status Codes Explained

This appendix provides additional explanation about the status code provided in Table 19-1.

31.1 GEV_STATUS_SUCCESS

The GEV_STATUS_SUCCESS status code is returned by the device whenever the operation executed successfully. For GVCP commands, this means the command was successfully completed. For GVSP, this means the data transfer contains the correct data.

31.2 GEV_STATUS_PACKET_RESEND

The GEV_STATUS_PACKET_RESEND status code is returned in a GVSP packet when this packet is due to a retransmission request. This helps the GVSP receiver confirms that the PACKETRESEND_CMD was correctly processed by the device as no PACKETRESEND_ACK is permitted.

Note that this status code must be enabled through the GVCP Configuration bootstrap register.

31.3 GEV_STATUS_NOT_IMPLEMENTED

The GEV_STATUS_NOT_IMPLEMENTED status code is set in the acknowledge message when the specified GVCP command is not supported by the device. This indicates to the application a possible mismatch in the GigE Vision version supported by the device.

31.4 GEV_STATUS_INVALID_PARAMETER

The GEV_STATUS_INVALID_PARAMETER status code is set in the acknowledge message of a GVCP request when the data provided in the request is not valid (or out of range) for the current situation. This can also be used when there are too many parameters provided. As an example, trying to write an unsupported value to a bootstrap register would return GEV_STATUS_INVALID_PARAMETER.

31.5 GEV_STATUS_INVALID_ADDRESS

The GEV_STATUS_INVALID_ADDRESS is returned by the device when the GVCP request tried to access an invalid address location because no register exists at that address

For instance, this could be an unsupported bootstrap register or an optional register not supported by the device. It can also be used when accessing a section in the manufacturer-specific register space where no register is mapped (as per the XML device description file).

31.6 GEV_STATUS_WRITE_PROTECT

The GEV_STATUS_WRITE_PROTECT status code is returned when a GVCP request tried to write to a read-only register or to a read-only memory area, or write to a register that cannot be written to. For instance, an application trying to write to the Version register would receive this status code.

The GEV_STATUS_WRITE_PROTECT shall not be used when only some fields of a register are read-only and the other fields are writable.

31.7 GEV_STATUS_BAD_ALIGNMENT

The GEV_STATUS_BAD_ALIGNMENT status code is returned when a GVCP request performed a memory access that is not aligned to the appropriate memory boundary (address or data). For instance, READMEM_CMD and WRITEMEM_CMD require the amount of data to be aligned to 32-bit boundary. If an application performs such a request with a quantity of bytes to read or write that is not a multiple of 4 bytes, then the device will return GEV_STATUS_BAD_ALIGNMENT. Moreover, this status code is returned when the address specified by the read or write access is not aligned to a 32-bit boundary.

31.8 GEV_STATUS_ACCESS_DENIED

The GEV_STATUS_ACCESS_DENIED status code is returned when a GVCP request could not be executed because the application does not have the required privilege or because it is trying to read from a write-only register or memory location. For instance, a secondary application cannot write to a device and it cannot read from a device when a primary application has the exclusive access mode.

31.9 GEV_STATUS_BUSY

The GEV_STATUS_BUSY status code is returned when a device cannot execute a GVCP request at this moment because it is currently not in the proper state. It tells the application to retry the request at a later time. For instance, the device is currently performing an operation that renders the memory location temporarily not accessible. This could be the case if an asynchronous request to calibrate a camera was in progress and the application would ask to capture an image. The device could then return GEV_STATUS_BUSY to inform the application that this request cannot be executed at this moment and to retry later.

31.10 GEV_STATUS_PACKET_UNAVAILABLE

The GEV_STATUS_PACKET_UNAVAILABLE status code is set in a GVSP packet if a PACKETRESEND_CMD asks for a packet that is not available anymore in the device memory, typically because it has been overwritten with new data.

This status code was introduced before GEV_STATUS_PACKET_NOT_YET_AVAILABLE, GEV_STATUS_PACKET_AND_PREV_REMOVED_FROM_MEMORY and GEV_STATUS_PACKET_REMOVED_FROM_MEMORY. As such it is recommended to use the latter 3 to provide better feedback about packet availability issues during a packet resend.

31.11 GEV_STATUS_DATA_OVERRUN

The GEV_STATUS_DATA_OVERRUN status code can be used in 2 scenarios.

First, it can be returned in the data trailer if the payload data has to be truncated because of insufficient internal memory to buffer the data. This can be the case if data acquisition is faster than data transfer over GVSP. Note that when a full block is discarded, the GEV_FLAG_PREVIOUS_BLOCK_DROPPED flag should be used instead.

Second, it can be returned during All-in transmission mode if not enough room is available to regroup the leader, payload and trailer information in a single packet.

31.12 GEV_STATUS_INVALID_HEADER

The GEV_STATUS_INVALID_HEADER status code is returned by the device if the validation of the GVCP packet header finds a problem other than an invalid key (0x42) or unsupported GVCP command. For instance, many commands require additional data on top of the header. If the header *length* field is set to 0, then the device should return a GEV_STATUS_INVALID_HEADER to indicate this GVCP command is malformed.

31.13 GEV_STATUS_PACKET_NOT_YET_AVAILABLE

The GEV_STATUS_PACKET_NOT_YET_AVAILABLE status code is linked to the packet resend logic and it is used to indicate that a packet that was asked for retransmission has not yet been acquired by the device. This can be the case in linescan cameras when the external line trigger rate is lower than the application timeout for GVSP.

Note that this status code must be enabled through the GVCP Configuration bootstrap register.

31.14 GEV_STATUS_PACKET_AND_PREV_REMOVED_FROM_MEMORY

The GEV_STATUS_PACKET_AND_PREV_REMOVED_FROM_MEMORY status code is linked to the packet resend logic and it is used to indicate that the requested packet and all the ones before have been removed from the GVSP transmitter memory. It is therefore not possible to retransmit any of them. This status code can appear in the GVSP header when the data acquisition is faster than data transmission and by the time the application determined that a packet was missing, this packet was already overwritten in the device memory.

Note that this status code must be enabled through the GVCP Configuration bootstrap register.

31.15 GEV_STATUS_PACKET_REMOVED_FROM_MEMORY

The GEV_STATUS_PACKET_REMOVED_FROM_MEMORY status code is linked to the packet resend logic and it is used to indicate that the requested packet has been removed from the GVSP transmitter memory. It is therefore not possible to retransmit it. Contrary to GEV_STATUS_PACKET_AND_PREV_REMOVED_FROM_MEMORY, this status code does not take stance on previous packets. Hence the application should not assume the previous packets are not available. This status code is put in the GVSP header of the data packet.

Note that this status code must be enabled through the GVCP Configuration bootstrap register.

31.16 GEV_STATUS_PACKET_TEMPORARILY_UNAVAILABLE

The GEV_STATUS_PACKET_TEMPORARILY_UNAVAILABLE status code is linked to the packet resend logic and it is used to indicate that the requested packet is momentarily not available.

This is to accommodate the functionality on cameras where they reserve some finite amount of bandwidth for resends and do not allow resends to consume more than that amount. This allows the application to intelligently hold off on requesting resends for a while rather than assuming its resend requests were lost (which if the camera just ignored the request would be what would happen).

The GVSP transmitter provides a hold-off time as guideline to the receiver to re-issue its packet resend request. The format of this hold-off time is modeled similarly to the Ethernet flow control PAUSE frame.

The transmitter suggestion for hold-off does not have to be precise and could just be a constant. This is left as a quality of implementation. A camera vendor could tie this into his internal bandwidth reservation system such that it could precisely indicate when the next packet could be sent while maintaining the specified bandwidth limits.

Note that this status code must be enabled through the GVCP Configuration bootstrap register.

31.17 GEV_STATUS_OVERFLOW

The GEV_STATUS_OVERFLOW status code is used to indicate that an internal queue of the device could not take additional content and it had to discard the request. It is up to the application to take proper action to ensure the queue does not overflow.

This is used by the Scheduled Action Commands queue when it overflows. In this case, the ACTION_ACK will return GEV_STATUS_OVERFLOW and the ACTION_CMD is not queued. The size of the Scheduled Action Command queue is available through the Scheduled Action Command Queue Size register.

Note that this status code must be enabled through the GVCP Configuration bootstrap register.

31.18 GEV_STATUS_NO_REF_TIME

The GEV_STATUS_NO_REF_TIME status code is used to indicate that the device does not have a reference time and hence it cannot schedule action commands to be executed at a precise time in the future.

The status of the IEEE 1588 clock can be obtained through the IEEE 1588 Status register.

Note that this status code must be enabled through the GVCP Configuration bootstrap register.

31.19 GEV_STATUS_ACTION_LATE

The GEV_STATUS_ACTION_LATE status code is used in the context of the Scheduled Action Command when the time at which the action command is to be executed (*action_time* field) is already past. This is used as an indicator to the application requesting the ACTION_CMD that mis-sychronization might have occured.

31.20 GEV_STATUS_ERROR

The GEV_STATUS_ERROR status code is a default status code to use when no other available status code adequately represents the cause of the problem. It should be avoided whenever possible as it does not provide much information to the receiver about the nature of the problem.

32 Document History

Version	Date	Editor	Description of Changes
1.0	2006-05-03	Eric Carey, DALSA	-Initial release from GigE Vision committee
1.1 draft A	2007-03-21	Eric Carey, DALSA	-Added CCP to list of acronym. -Added reference to DHCP RFC in [R3-12d]. -Added reference to LLA RFC in [R3-20d] -Clarified to use application dynamic port as destination for GVCP ACK message. -Changed CPP to CCP in figure 9-1 and figure 9-2. -[R9-12d] now clearly indicate MCP and SCPx must be cleared on Control Channel Connection closure. -[O13-2] now indicates the device should validate the 0x42 key. -Clarification to [O13-4d] to indicate the acknowledge message value must be the unsupported command message value + 1. -New requirement [R14-62a] to force ACKNOWLEDGE bit to be set in flag field of a DISCOVERY_CMD. -Added the GEV_STATUS_PACKET_RESEND status code. Note this is an informational status code. -New conditional objective [CO14-63d] to recommend usage of GEV_STATUS_PACKET_RESEND status code on a packet resend. -Removed various references to feature not supported in XML (Device-specific message, Device-specific status code, Device-specific pixel format). -Clarified inter-packet delay is from end to start of packets. -Added support for GVSP_PIX_MONO14. -Corrected a typo in figure for GVSP_PIX_YUV422_PACKED. -Clarified that FC can be use in compliancy matrix for a conditional requirement/objective if the condition is false. -Removed [O3-22d] since LLA cannot fail (it retries infinitely). -Added a note to [R3-5d] to indicate 2 acceptable interpretations possible for DHCP client when moving to LLA. -Add note above [CO3-10d] to indicate Persistent IP 0.0.0.0 and 255.255.255.255 are not valid when interface is configured. -Clarify behavior when DHCP lease expire. Added a new section for this. -Remove [O3-17d]. -New conditional requirement [R11-7a] to use the UDP source port from the message channel as the UDP destination port in the acknowledge message. -Note highlighted the spec. does not define an acknowledge timeout. -Change wording in [R9-6d] and [R9-15d] to reference FORCEIP_CMD. -Change to [CR14-48d] to remove length = 0. -Adjusted the definition of .ZIP file format in section 17. -Add a note about ICMP Destination Unreachable in section 20. -Removed [CO23-11d]. -Change to [O27-8d] to differentiate error returned on a read vs write access.
1.1 draft B	2007-05-19	Eric Carey, DALSA	-Clarify usage of CPP register when both exclusive_access and control_access are set. -Change [R24-10d] into a conditional requirement CR24-10d. -Possibility to broadcast FORCEIP_ACK.

			-Rewording of [CR14-61d] -Change GVSP_PIX_RGB (now deprecated) to GVSP_PIX_COLOR. -Remove [CR28-10d] -AcquisitionStart and AcquisitionStop features are optionally readable. -New status code: GEV_STATUS_WRONG_CONFIG. -Add a new bootstrap register at 0x0950 to represent the max. randomized delay for DISCOVERY_ACK -Change Version bootstrap register to refer to version 1.1 of this specification. -Clarification to [R4-3d] to support any type of broadcast request. -Remove [CO3-10d]. -Factory default of MCRC and MCTT are now device-specific. -New bootstrap register at 0x0954 for Heartbeat Control. New capability bit to indicate if Heartbeat can be disabled. -New optional bootstrap register to report current Link Speed of each supported network interface. A capability bit indicates if these registers are supported. -Deprecate events related to image acquisition. These should now be specified by GenICam Standard Naming Convention for GigE Vision. -Update reference to GenICam and Standard Feature Naming Convention to reflect most recent release. -New objective indicating bootstrap registers should be part of the XML device configuration file and respect the rules of GenICam Standard Feature Naming Convention for GigE Vision. -Further clarifications about .ZIP usage: only DEFLATE and STORE compression algorithms are permitted. -Added clarification to Chunk Data Payload format and an example. -New bootstrap registers to retrieve Primary Application Port and IP address. -Support for PENDING_ACK message and Pending Timeout bootstrap register. -Introduction of the Manifest Table. -Added support for the following pixel types: GVSP_PIX_BAYBG10_PACKED, GVSP_PIX_BAYGB10_PACKED, GVSP_PIX_BAYGR10_PACKED, GVSP_PIX_BAYRG10_PACKED, GVSP_PIX_BAYBG12_PACKED, GVSP_PIX_BAYGB12_PACKED, GVSP_PIX_BAYGR12_PACKED, GVSP_PIX_BAYRG12_PACKED, GVSP_PIX_BAYGR16_PACKED, GVSP_PIX_BAYRG16_PACKED, GVSP_PIX_BAYGB16_PACKED, GVSP_PIX_BAYBG16_PACKED, GVSP_PIX_YUV422_YUYV_PACKED, GVSP_PIX_RGB12V1_PACKED and GVPS_PIX_RGB16_PACKED. -Capability to support LFSR-generated data to fill the payload data of the test packet. -Option to support SCSPx and MCSP to help firewall traversal. Addition of stream channel and message channel capability registers. -Add SCCx bootstrap registers for per stream channel capabilities. Introduction of big endian support flag. -Add GEV_STATUS_PACKET_NOT_YET_AVAILABLE status code that can be used by linescan cameras when the line trigger rate is slower than the application timeout for packet retransmission. -Device Mode register factory default is now device-specific. -Clarified definitions of FC, NC and NS for conditional requirement and conditional objectives. Put them in tables with clear definition. -Remove [O13-8a]. -[R14-10a] is now a conditional requirement since application don't have requirement to support FORCEIP_CMD. [R14-15] and [R14-16] now only apply to devices. -[R24-27] and [R24-28] now only apply to devices. -[R25-2] now only applies to devices. -Reword [CR27-28a] to clarify the condition since an application does not have to use this feature.

1.1 draft C	2007-06-10	Eric Carey, DALSA	-Add support for ACTION_CMD and ACTION_ACK. Up to 128 actions can be defined. -Add new contributors to technical committee (members attending the technical meetings). -Fix pixel order for GVSP_PIX_BAYBG8 and GVSP_PIX_BAYBG10 and GVSP_PIX_BAYBG12. -Change example in [O4-4d] to remove an incorrect subnet for a PersistentIP address since we do not validate classes anymore. -Add indication for optional registers. -Put unique numbers to new requirements. -Add new requirements to the compliancy matrices. Indicates requirements no longer applicable.
1.1 RC1	2008-10-14	Eric Carey, DALSA	-Add note under [R14-5d] to highlight fact that device might not be able to adequately verify if it reside on same subnet as the application. -Insert clarification for READREG and WRITEREG when accessing string register. Little-endian device will convert data to big-endian as required by [R3-19]. -Highly recommend supporting WRITEMEM for device used through GenICam interface. -Bit 0 of Device Mode register indicates the endianess of all device registers. -Remove reference to chair and vice-chair of GigE Vision committee. This should be part of other AIA documentation. -Clarify device should move to next IP configuration scheme if PersistenIP is 0.0.0.0. -Remove use case from [O4-4d]. -Revisit the graphics for the various Pixel Formats. -Change to Manifest Table to support only XML documents. Other documents will be supported by entries within the XML file. - Indicate that Primary Application Source port and IP address must be reset when control channel is closed in [R9-12d]. -Add enable bit to GVCP Configuration register to enable extended status codes introduced in version 1.1. -Modify CR27-46d to only discard packets targeted at the device SCSPx port and MCSP port.
1.1 RC2	2008-10-29	Eric Carey, DALSA	-Correction to figure representing YUV411. -Set reference to GenICam version 1.1.1. -Add note to indicate why ACTION_CMD does not reset the heartbeat. -Discovery ACK Delay register is optional. Change [R27-42d] to [CR27-42d]. Add a capability bit for availability. Add capability bit when it is writable. -GVCP Configuration register default is 0 to allow backward compatibility to version 1.0 Application software. -Add 2 status codes: GEV_STATUS_PACKET_AND_PREV_REMOVED_FROM_MEMORY and GEV_STATUS_PACKET_REMOVED_FROM_MEMORY. -Clarify length of LFSR algorithm. -Link speed registers expressed in Mbps in a single 32-bit register. -Add explanation for multi-word to access the lowest address first. -For compliancy matrix, allow FC to be used for unsupported conditional requirements. -Remove one of the conditions allowing a device to broadcast a DISCOVERY_ACK. -Remove action_id field in ACTION_CMD.
1.1 RC3	2008-12-10	Eric Carey, DALSA	-Added capability bit in GVCP capability register to indicate if extended status codes are supported. -Add note below [CR14-61d] to indicate a device is to use GEV_STATUS_PACKET_UNAVAILABLE if it does not support extended packet resend status codes. -Change [O14-6d] to reflect the case when the Discovery ACK delay register is not supported. -Switch order of the conditions in Table 9-1.

1.1	2009-02-04	Eric Carey, DALSA	Official release of GigE Vision 1.1 specification
1.2 draft A	2009-03-29	Vincent Rowley, Pleora Technologies	Updated the first 2 pages of each section as a proof of concept for the new product classification initiative.
1.2 draft B	2009-05-07	Vincent Rowley, Pleora Technologies	Updated full bootstrap registers proposal for new product classification initiative.
1.2 draft C	2009-07-03	Vincent Rowley, Pleora Technologies	First draft of the new product classification proposal made available for public review. It also includes a number of the maintenance work identified for the release of version 1.2 of this specification. This includes: - Clarified requirements [O4-4cd], [R9-3ca], [R9-6cd], [R9-16cd], [CR14-61cd], [O15-4ca] and [O15-9ca]. - Clarified section describing device control (Section 10.6). - Updated the description of group_key and group_mask in Section 14.1 in order to be consistent with other sections of the standard text. - Described GEV_STATUS_PACKET_AND_PREV_REMOVED_FROM_MEMORY and GEV_STATUS_PACKET_REMOVED_FROM_MEMORY in the core of the packet resend text (Section 16.7.2). - Removed PACKETRESEND_ACK from Section 18. - Clarified GEV_STATUS_WRONG_CONFIG status code in Section 19. - Clarified recommendation on how to handle destination unreachable messages. - Removed PACKETRESEND_ACK from Data Resend Flowchart. - Clarified that the last data packet of a block can be padded so that all data packet have the same size. - Added the “Image Format” field in the table below [CR25-42]. - Fixed typos in requirements [CR27-45cd] and [CR27-46cd]. SCPx is now referenced as opposed to SCPSx and SCSPx. - Deleted minimal GenICam XML file. - Fixed link to GenICam Standard Features Naming Convention to point to GenICam web site. - Fixed some typos.
1.2 RC1	2009-08-30	Vincent Rowley, Pleora Technologies	- Defined management entity. - Added primary application switchover concept. - Better documented suffix terminology for requirements and objectives. - Removed [O7-15cd], [R7-16c] since considered as redundant. - Augmented registers lists in Section 10.2, 0 and 12.1. - Added unconditional streaming concept. - Reworked Section 10.6 in order to improve its technical accuracy. - Added a note to clarify PENDING_ACK behavior with respect to secondary applications. - Updated Figure 14-5. - Moved note indented to promote transceivers compliance with GigE Vision 1.0 and 1.1 applications to Section 11. - Added a note with respect to the usage of GEV_STATUS_PACKET_UNAVAILABLE, GEV_STATUS_PACKET_AND_PREV_REMOVED_FROM_MEMORY and GEV_STATUS_PACKET_REMOVED_FROM_MEMORY. - Renamed [CR14-68cd] to [CR14-68c]. - Replaced [O15-4ca] and [O15-9ca] by [O15-15ca]. - Added extended chunk data payload type. - Replaced [CR23-12st] and [CR23-13st] by [CR23-12s] and [CR23-13s].

			- Added RGB565 and BGR565 pixel formats. - Deprecated BayerGR16Packed, BayerRG16Packed, BayerGB16Packed and BayerBG16Packed pixel format names and replaced them by BayerGR16, BayerRG16, BayerGB16 and BayerBG16. - Removed [CR28-20st] since not testable. - Clarified some ambiguous text and fixed some typos. - Improved layout (deleted some carriage returns, etc.).
1.2 RC2	2009-10-09	Vincent Rowley, Pleora Technologies	- Defined primary application switchover request concept. - Updated LSB and MSB definitions. - Updated GenICam version. - Removed [R7-17ca]. - Replaced [O13-4cd] and [O13-5cd] by [O13-4c] and [O13-5c]. - Clarified some ambiguous text and fixed some typos.
1.2	2009-12-23	Vincent Rowley, Pleora Technologies	Official release of GigE Vision 1.2 specification. - Removed RC2 tag. - Removed line numbers. - Added copyrights page. - Inserted page breaks in order to make the document more printer friendly. - Updated licensing statement in Section 1.1. - Updated list of contributors in Section 1.2.
2.0 draft A	2010-09-06	Eric Carey, DALSA	- Fix many typos and grammatical style issues reported onto mailing list. - Change [R14-55ca] to a conditional requirement. - With GigE Vision 2.x, strings in the bootstrap registers are not required to be NULL-terminated if they occupy the full memory space allocated to the register. This mainly affects [R27-5cd]. - New character encoding for string added: ASCII. Recommended character set for GenICam support is ASCII. Previously, only UTF-8 was supported. - New appendix to clarify usage of status code. - Recommendation that Bayer CFA pixel formats have OffsetX and OffsetY aligned to the size of the Bayer tile. And that camera deals with ReverseX and ReverseY by internally adding an offset of 1 pixel. These 2 recommendations ensure the PixelFormat configured by the device match the pixel format field of the Image Data Leader. - Add trademark symbol to GenICam™ references. - Add [R22-5st] to define how a GVSP transmitter deals with data payload packet being discarded internally. - Add [R22-6st] to define how a GVSP transmitter deals with one or more data blocks being discarded internally. - Factory default of Persistent IP, subnet mask and default gateway bootstrap registers are now device-specific. They used to be 0. - Stream Channels and Message Channel Capability registers are now mandatory. Consequently, [CR27-39cd] and [CR27-40cd] are removed. - Clarify how DISCOVERY_ACK should be sent (unicast or broadcast). Removed [R4-3cd] and [O4-4cd] as they duplicate information in the DISCOVERY_ACK section. Add [O14-77cd] to specify when DISCOVERY_ACK is broadcasted. - Clarify that size x and size y fields in the Image data leader represent the dimensions of the image transported in the current data block. - Create appendix 3 that contains information about connectors.
2.0 draft B	2010-12-10	Eric Carey, DALSA	- Explicitly state the any speed grade supported by IEEE 802.3 can be used by GigE Vision.

			- Add support for unconditional ACTION where ACTION_CMD are processed even though the primary control channel is closed. Provide a capability and an enable bit in the corresponding GVCP register. Update the list of 4 conditions to assert the ACTION_CMD. - Introduction of PacketSize mandatory feature to represent the range of value supported for SCPS packet size field. - The packet size field of SCPS can only be changed when the application access the SCPS register. It is allowed for the device to round the value that is written, but it is not allowed to change this value without having the application directly writing into this register. - Change Align_565Packed figure so that each color component starts with bit 0. - Clarify that PixelFormat feature represents the format of the pixel output on the stream channel. - Clarification to [R24-20st] that block_id must be incremented to next valid value. - For a given stream channel, a GVSP transmitter must reset the block_id to 1 when the stream channel is opened. - Correct address of Control Switchover Key in section 27.36. - Increase Pending Timeout register from 16 to 32 bits. Make it mandatory. Clarify it must consider the worst case command execution time, including concatenation. - Change [R3-19cd] to [O3-19cd] to allow a device to use RFC2131 retransmission strategy. This helps devices based on a standard communication stack, such as a GigE Vision camera simulator that can be run from a PC. - Clarification to [R12-8st] and [R12-11sr] to use the network interface index field of SCPx. - Remove [CR17-2cd] since it is already covered by chapter on Standard Feature List. - Let factory default value of SCPSx be device-specific for both GVSP transmitters and receivers. - For RGB planar pixel format, clarify that 3 consecutive stream channels must be used, with R using the first, G the second and B the last. - Explicitly state that at least one register must be accessed by READREG and WRITEREG command. - Change layout of chapter about Data Block and Data Packet Headers. First chapter now contains the common information about Data block. Second chapter contains the details of each payload format. - Creation of Standard Transmission mode and All-in Transmission mode. The standard transmission mode uses different packets for Data Leader, Data Payload and Data Trailer. The All-in Transmission mode combines them in a single packet. - Addition of ALL_IN_FORMAT packet format for GVSP where data leader, data trailer and data payload are combined in one packet. - Introduction of new Payload Type for image compression: 0x0006 for JPEG data, 0x0007 for JPEG 2000 data and 0x0008 for H.264 data. - Add references for image compression standard and connector standard. - Provide short description of IEC/PAS 61076-2-110 pre-standard for M12 connector working up to 10 Gb/s. - Introduction of 64-bit block_id. Called them block_id64 to distinguish from the 16-bit block_id. - Add capability bit in Stream Channel Capability register to announce 16-bit block_id support. - Add a global GVSP Configuration register to allow enabling 64-bit block_id mode. - Adapt PACKETRESEND_CMD, EVENT_CMD and EVENTDATA_CMD to support 64-bit block_id. - Adjust Version register to reflect this version of the specification. - Add requirement for Application to ignore reserved field in bootstrap registers. This will ease future interoperability problem when we start using these bits. - Explicitly state that reserved fields in packets are set to 0 on transmission and ignored on reception. - Added msb and lsb definition to represent bit alignment. MSB and LSB are reserved for byte alignment. - Introduce bitfield size information for all packet definition tables. - Add support for interlaced scan for Image, Extended Chunk, JPEG and JPEG 2000 Payload Type. 2

			bitfields are added in the Data Leader packet. - Introduce concept of extended chunk mode to enable appending chunks to JPEG, JPEG 2000 and H.264 payload types. - Convert the Reserved field in the Data Leader to a payload type specific field since this was already in usage for Extended Chunk Payload Type and it did not match the generic definition for the Data Leader. Proper definitions of this specific field were added to the payload type sections. - Standardize all fields in Data Leader, Data Payload and Data Trailer packet to use lowercase. - Provide an out-of-order flag for Image Payload type to allow a GVSP transmitter to send Data Payload packets out-of-order within a data block. But Data Leader must be the first packet of a data block, and Data trailer must be the last packet. This can be used with multi-tap sensors. - New chapter about Physical Link Configuration describing the 4 configurations: Single Link, Multiple Links, static Link Aggregation and dynamic Link Aggregation. - Clarification to the following bootstrap register to account for link aggregation: MAC address, supported IP configuration, current IP configuration, current IP address, current subnet mask, current default gateway, persistent IP address, persistent subnet mask, persistent default gateway, link speed. - Clarification to SCPDx to indicate the inter-packet delay is on a given physical link. - Add the following bootstrap registers: Number of Active Links, Physical Link Configuration Capability and Physical Link Configuration. - Change to Device Mode register to indicate the current link configuration. - Change [CR3-3cd], [CR3-4cd] and [CR10-7s] to be conditional to ML Configuration instead of multiple network interface (since LAG only shows a single interface even though there are multiple physical interfaces). - Set unused bit of Number of Network Interfaces register as reserved for possible future usage. - Move introduction of section “Device with Multiple Network Interfaces” to the “Physical Link Configuration” chapter. Corresponding requirements have also been moved. - Add GEV_EVENT_LINK_SPEED_CHANGE to standard list of events. - Add section to describe IEEE 1588 Precision Time Protocol. Update the GVCP Capability and GVCP Configuration registers. Provide equations to convert between IEEE 1588 80-bit timestamp and GigE Vision 64-bit timestamps. Update the description of Timestamp Value register to reflect IEEE 1588 usage. Include IEEE 1588 profile information. - Augment the ACTION_CMD to allow for action to be executed at a specific time in the future. Provide description in the section describing Action Commands. - Indicate that ACTION_ACK must be sent back as soon as the action command is queued for execution at a future time. Introduce GEV_EVENT_ACTION_LATE to be fired if action time is past. A bootstrap register added to indicate the queue size. - Creation of the “Additional Concepts” chapter to regroup sections that don’t directly fit with the Control Channel chapter. Move Retrieval of XML file, Device Synchronization, Action Commands and Primary Application Switchover in that new chapter. - Rework GVSP section describing Flow Control. Introduce usage of IEEE PAUSE mechanism. Rename Supported IP Configuration register to Network Interface Capability register. Rename Current IP Configuration procedure register to Network Interface Configuration register. Provide 2 bits in these register to report and enable PAUSE reception and PAUSE generation. - New requirement that all device registers must visible be in big-endian (OK if the implementation is little endian as long as they are shown and accessible as big-endian). - Promote WRITEMEM to mandatory command to support. This will ease interfacing to GenICam. Remove [O27-3ca]. - Remove [O7-5c] as GVCP packets are too small to be fragmented. - For [O10-3st], provide the IANA range for dynamic ports. - Remove [O12-4cd] since it is redundant with the more generic [O27-7cd]. Convert [O27-7cd] into requirement [R27-7cd] to have consistency in accessing unsupported bootstrap registers. - Convert [O13-4c] into requirement [R13-4c]. - Add requirement for application to check device Packet Resend capability before asking for packet

			resend. This eliminates the need for [CO14-57cd] which has been removed. - State the application should use the Pending Timeout register to determine suitable GVCP transmission timeout value. - Convert [CO27-16cd] into conditional requirement [CR27-16cd] to favor error handling. - Convert [CO27-17cd] into conditional requirement [CR27-17cd] to favor error handling.
2.0 draft C	2011-02-04	Eric Carey, DALSA	- For Packet Resend, add option to resend packet to IP source address of the PACKETRESEND_CMD instead of the address specified by SCDAx register. Provide capability in SCCx and configuration bit in SCCFGx for this feature. - [O14-54ca] obsolete due to new requirement asking for application to check Packet resend support by the device before issuing such commands. - Remove [R14-58cd] since it duplicates the more general [R9-15cd]. This is about Packet Resend command not resetting the heartbeat. - Add a resend range error flag by claiming one bit from the payload format field. Change the Packet Resend answer by allowing sending only a single GVSP packet to report an error status code affecting a range of stream channel packets. - Introduction of GEV_STATUS_PACKET_TEMPORARILY_UNAVAILABLE. Add GVCP capability and configuration bit to enable generation of GEV_STATUS_PACKET_TEMPORARILY_UNAVAILABLE. Optional field in GVSP packet to offer hold-off time. - When device is configured for 64-bit block_id, then the legacy block_id field is converted into a 16-bit flag field. The upper 8 bits are device-specific, the lower 8 bits are defined by GigE Vision. Bit 15 must be sent when the GVSP packet is the result of a packet resend request. - Reformatting section on Pixel Format to link it to the Pixel Format Naming Convention document. - Added Mono1p, Mono2p and Mono4p pixel formats. - Updated #define for GVSP_PIX_xxx pixel definition. Updated the PixelFormat list. - Convert [R12-11sr] into conditional requirement [CR12-11sr] where the condition is the GVSP receiver to be a device. - For JPEG and JPEG 2000 Payload Format, add a color space selection bit to allow using either GigE Vision pixel format or JPEG 2000 EnumCS color space values. Change the “color space” field to “data format”. - Add clarification to the JPEG, JPEG 2000 and H.264 sections following an update of the Image Compression proposal (after review by domain experts). - Allow interlaced scan support for H.264 by adding field ID and field count in the data leader. - Illustrate the pixel format using little-endian convention with byte 0 on the right of the figures. This is in-line with the Pixel Format Naming Convention. - Correct mix-up in GEV 1.x where GVSP_PIX_BGR10V1_PACKED and GVSP_PIX_BGR10V2_PACKED where in the text, but replaced for RGB10V1 and RGB10V2 in the #defines and in the PixelFormat Feature. RGB ordering prevails for these 2 cases. - Change to Pending Timeout definition introduce in draft B so it does NOT consider concatenation. This is better left to the application since it knows how many read or write commands it put in the request. - Convert [R7-12ca] into [O7-12ca] and state that req_id does not necessarily increment in the case of broadcasted commands. Add new requirement that successive req_id on control channel must be different (but not necessarily increment). Provide a note to explain that device should not expect req_id to increment. Make some adjustment throughout the text to explain the impact of this broadcast case on req_id. - Change mandatory feature “PacketSize” to “GevSCPSPacketSize” to align to SFNC. - Corrections to chapter on “Standard Features List for Cameras” to align to most recent definition of SFNC for the non-mandatory features. - Standardize naming to “XML device *description* file” to avoid any confusion. There were instances referring to “XML device *configuration* file”. - Populate appendix 2 – Status Code Explained.

2.0 RC1	2011-03-14	Eric Carey, Teledyne DALSA	- Provide introductory text for LLA, as part of the zeroconf effort introducing mDNS and DNS-SD. - Add optional support for mDNS and DNS-SD. - Inserted some figures from Compression proposal and for All-in Transmission mode. - Correct Table 11-1 to compare 16-bit vs 64-bit block_id. - Add conditional requirement that DISCOVERY_CMD must be answer during a PENDING_ACK situation. This replaces a description in the text. - Add conditional requirement that PENDING_ACK cannot be used when receiving DISCOVERY_CMD, FORCEIP_CMD and PACKETRESEND_CMD. This replaces a description in the text. - Clarification to Pending Timeout register to indicate it can report 2 different values depending if PENDING_ACK is enabled or not. - Introduction of a new requirement numbering style to eliminate dependency to section. With GEV 2.0, new sections were introduced in the text and this does not match the numbering style of GEV 1.x. Actual requirement numbers will be introduced in last release candidate so they appear sequential in the final text. Right now, use 'xx' placeholder to facilitate retrieval. GEV 1.x requirement move at the end of requirement text (to help cross-reference).
2.0 RC2	2011-04-29	Eric Carey, Teledyne DALSA	- Clarify first byte to be sent in the test packet data when LFSR is used (0xFF is sent) in [CR24-29st]. - Add "Meaning" column to suffix table (table 1-2). - Remove reference to "previous version of this specification". Replaced with "versions prior to 1.2". - Reword [R12-2cd] to use the word MUST since it is a requirement. - Remove legacy strikethrough text as this is a new major version of the text. - Remove redundant info in section 3.2 about hard-coding stream channel to specific NIC. - In section 10, convert a note about device that have to answer discovery request into a requirement. - Move Appendix 4 (GVCP commands) to section 10.6 – "Controlling the Device". - Add requirement that PENDING_ACK cannot be used to answer a DISCOVERY_CMD. - Change a sentence into requirement: block_id must be reset to 1 upon stream channel open. - Clarification in section 14.1.2 that the XML store on the vendor web site can point to a specific folder location. The example already showed that possibility. - Correction to maximum index in WRITEREG_ACK. For WRITEREG_CMD, 67 registers can be written in one command. Therefore, the 0-based index of WRITEREG_ACK can go from 0 to 66. - Add GEV_STATUS_PACKET_TEMPORARILY_UNAVAILABLE to list of status code (both in the table and in the appendix). These were overlooked when this new status code was introduced. - Indicate that maximum number of EVENT in one packet is 33 for 16-bit block_id and 22 for 64-bit block_id. - Standardize synthax of big-endian and little-endian throughout the document (with a hyphen). - Fix some invalid cross-references. - Standardize bit breakdown in packet header description to start from bit 0 (msb) and progressing towards bit 31 (lsb). - Fix error in Extended Chunk mode to state that length field must have bits 30 and 31 clear to align to 32-bit boundary. - Refactoring presentation of bootstrap registers. Use a standard template to show bit position. Ensure a requirement or objective is present for each bootstrap register (along the same line as for pixel formats). Remove the constant reference to [R27-7cd]. - Generalize [CR27-16cd] for READMEM so it applies to all unsupported string bootstrap registers. Same for [CR27-17cd] and WRITEMEM to unsupported string bootstrap registers. - Remove [R27-19cd] and [R27-20cd] that are redundant with the generic [R27-5cd]. This is to indicate the character set used by the strings. - Use Arial font to highlight bootstrap register names throughout the text.

			- Revise names in Table 27-1 (Bootstrap registers) to ensure consistency with the actual bootstrap name listed in section 27.
2.0 RC3	2011-06-16	Eric Carey, Teledyne DALSA	- Add note in SCDA and MCDA about the types of ARP reply that might be sent back. - Improvement to the Gratuitous ARP definition. - Rewording of [CO3-11cd] to align it to RFC5227 (IPv4 Address Conflict Detection). Add RFC5227 to the list of references. - Multiple fix for typos and text improvements. - [R14-13cd] now indicates to restart IP configuration on all network interfaces when using FORCEIP with IP address of 0. - Extended Chunk Mode can be used with All-in Transmission mode. - For JPEG and JPEG2000, remove restriction that both fields must have the same timestamp. This was overlooked from final compression proposal. - Add comment that H.264 is unlikely to carry interlaced video. - Re-introduce a note with the text of former [O27-3ca]. - Clarify in some sections that inter-packet delay represents the “minimal” delay to insert between packets on a given stream channel. This was not consistent across the text. - Remove the following unused status codes: GEV_STATUS_MSG_MISMATCH, GEV_STATUS_INVALID_PROTOCOL, GEV_STATUS_NO_MSG. - Indicate in Table 10-1 if unicast or broadcast is allowed for read and write commands. - Manifest Table is optional for device, but mandatory for application. The Manifest Table is preferred over the First URL and Second URL registers. - For Planar pixel formats, highlight that each color plane uses a different stream channel. - Remove the recommendation designation from chapter 18 as this does not match the requirement/objective terminology. - Remove note about BOOTP and RARP as this conflicts with IP Configuration section. - Add support for 3-field interlacing. Clarify that Field 1 corresponds to the first line of the de-interlaced image. - Clarify that timestamp register and fields are limited to 63-bit when using a GenICam interface (signed integer in GenICam). - Change requirement to objective for factory default of 16-bit block_id. - Rename “Future Time Action Commands” to “Scheduled Action Commands”. - Add a capability bit in GVCP capability register for Scheduled Action Commands. There is no corresponding enable bit as the application decides if it puts a future execution time in the ACTION_CMD. - Introduction of GEV_STATUS_OVERFLOW to indicate the Scheduled Action Command queue is full and cannot take the additional request. Add corresponding conditional requirement for Scheduled Action Command. - Introduction of GEV_STATUS_NO_REF_TIME to indicate that the device does not have a time reference and hence cannot schedule action commands. Add corresponding conditional requirement for Scheduled Action Command. - Re-introduce all 1.x GigE Vision-specific pixel formats that don’t have an equivalent in PFNC. - Add new pixel format for YCbCr, YCbCr422, YCbCr411, YCbCr601, YCbCr601_422, YCbCr601_411, YCbCr709, YCbCr709_422, YCbCr709_411. - Re-introduce that for pixel data, first byte transmitted on the wire is Byte 0. - Add requirement that reserved field of GVCP command are set to 0 on transmission and ignored on reception. - Provide note that Unconditional Action Commands enable is necessary to enable protection and security measure (ex: in industrial control systems). - Add clarification that Persistent IP and User-defined Name registers must persist the value as soon as it

			is set by the application. - Remove redundant [CR3-4cd]. [CR12-3cd] already covers this. - Remove recommendation to use same GVSP packet size in ML configuration. - Indicate for [R14-1cd] that length field must be set to 0. - Add objective to broadcast FORCEIP_ACK when broadcast bit is set. This was only text in previous versions. - Revise list of new status codes for GVCP Capability and GVCP Configuration registers. - Adjust requirement to allow rounding of the GevSCPSPacketSize feature in the same manner as the SCPS register. - Introduction of 32-bit packet_id to be used when 64-bit block_id are enabled. Impact the GVSP header and PACKETRESEND_CMD. - Change lower limit of GevSCPSPacketSize to 46 bytes since this value does not include the header. - Add clarification to the 2 ways to support multiple ROI: same stream channel vs different stream channel. - Indicate that only chunk image data is transmitted in little-endian for the chunk data payload type. - Regroup under [CR27-34cd] the acknowledge generation requirements (part was in [CR27-33cd] before). - Add requirement stating that the range of address from 0x0000 to 0x9FFF is strictly reserved for bootstrap registers. Manufacturer-specific register must be allocated after that range. - Clarification to Figure 16-4 so it references the proper GVSP packet header definition. - Add a section about line and image boundaries. Add note that Width increment can be made coarser to avoid packing pixels from different lines together (Mono1p being a good example). - For URL, put reference to RFC3986 and indicate which part of the URL strings are case-sensitive and case-insensitive based on this RFC. - Replace GEV_STATUS_PREVIOUS_BLOCK_DROPPED status code with GEV_FLAG_PREVIOUS_BLOCK_DROPPED bit flag. - Introduce a table to list all the GEV_FLAG_xxx supported by the flag field of the GVSP header. - Remove a duplicate requirement in the section about Pending Timeout register. - Move [R27-30ca] to GVCP section describing how to close a control channel. - Move [CR27-31ca], [CR27-33cd], [CR27-34cd], [CR27-35cd], [CR27-43cd] and [CR27-44cd] to GVCP section about message channel. - Move [R27-36ca], [CR27-45cd] and [CR27-46cd] to the stream channel section. - Generalize the concept of Extended Chunk Mode. Deprecate Extended Data Chunk Payload. Move [CO23-10st] to the Chunk Data section. Move generic chunk information to the Chunk Data section. - Allow device manufacturer to either force Interface #0 to a given physical interface or to dynamically detect it (first cable connected). - Allow a LAG to be one of the links in ML configuration. A single aggregator is still permitted, but it can be seen as a “faster” links and occupy any of the 4 interfaces allocated for ML. - Add requirement that PAUSE configuration in the Network Interface Configuration register must persist across device reset. Add note that the setting of PAUSE in this same register might be hard-coded (not configurable). - Clarify that FORCEIP cannot be used to apply the new physical link configuration. - Indicate that for H.264, not all payload packets have the same size. NAL starts on a new packet boundary. Receiver to use the packet size to determine the gap to next packet when a NAL unit does not fill a payload packet. - Align the text to the last image compression proposal which contained some changes with respect to initial text included in this draft. - Use trademark symbol ™ only on the first instance of each chapter for a given trademark name. This makes it obvious to the reader who owns the trademark, but does not clutter the text needlessly. Use a trademark name as an adjective whenever possible, as recommended by standard practices on trademark

			usage. - Introduce template for Requirements and Objectives Reference Table (to replace compliancy matrix). Start populating the table. Create a table of deprecated requirements for reference. - Add use case to explain why requirement to have sequential req_id has been relaxed for the broadcast scenario. - Change to [R7-3c] to allow using a different GVCP port when it is advertised using mDNS. - For DNS-SD, state that hexadecimal value must be expressed using uppercase. - Clarify that user-defined keys are allowed for DNS-SD. - Clarify that from a DNS-SD perspective, a service is defined as a GVCP control channel. - For DNS-SD, clarify that host name and service name do not follow the same set of rules. - Add a DISCOVERY broadcast summary table to show the possible use cases. - Rework list of participants to the technical committee. - Remove GEV_STATUS_DATA_TRUNCATION since the All-in transmission mode specify to use GEV_STATUS_DATA_OVERRUN. - Add explicit requirements about usage of GEV_STATUS_BAD_ALIGNMENT and GEV_STATUS_BUSY.
2.0 RC4	2011-08-07	Eric Carey, Teledyne DALSA	- Minor adjustment to multiple ROI definition. - Align Width and Height feature with latest graphic from SFNC. - Clarify the description of padding x and padding y for image payload type. - Change the definition of the IEEE 1588 clock status field to exactly match IEEE 1588 specification. - Deprecate GEV_STATUS_LOCAL_PROBLEM and GEV_STATUS_WRONG_CONFIG. Mark other status code removed in RC3 as deprecated since they are valid for 1.x device. - Specify status code to return if an application tries to reset the Timestamp Control register. - Packet format field of GVSP header supports various valued for DATA_PAYLOAD_FORMAT to help a dissector interpret the packet header by simply looking at its content. Add an objective for GVSP to have packet self-describing. For the moment, H.264 adds a new packet format. - Grammatical improvement from tech. writer. - For chunk, indicated that the chunk data payload length field in the data trailer is necessary to find the location of the tag when the last GVSP payload packet is padded. - Combine [R17-4ca] with a new requirement for the usage of manifest table to avoid confusion. - Add fields in H.264 data leader to indicate profile and sequence parameter set data attributes. - Adjustments to JPEG 2000 codestream figure. - Clarify that endianess of the chunk data is defined by the XML file. - Remove out-of-order data transmission for Image payload type. This is going to be replace with the multi-zone proposal. - Remove appendix about connector since it is incomplete. A specific connector document will be generated by the committee. - Add support for multi-zone payload type. - Add All-in Packet for Device-specific payload type. - Introduction of a Document Typographic Convention. Adjust text accordingly. - Adjustments to GVCP and GVSP packet figures and fields to follow typographic convention. - Add note that field order is a de-interlacer control parameter and it is not provided as part of the stream. - Create GEV_FLAG_RESEND_RANGE_ERROR flag to replace a specific bit field in the GVSP packet. - Clarification to why a timestamp field is necessary in each data payload packet for H.264. - For Generic Data Leader, move timestamp field to the Payload Type specific section to accommodate

			H.264. - Add use case examples to mDNS section. Introduce Zeroconf Discovery to designate the combination of mDNS and DNS-SD. - Explanation of how to use Extended Chunk for H.264 and Multi-zone Image where packet format ID differs from the generic packet format ID (3). - Breaking down [CR14-61cd] for PACKETRESEND response to distinguish the resend range and hold-off scenarios. - Resend range only possible for Extended ID mode. - For GVSP packets, provide reference to a another payload type for fields that are the same instead of cut-and-pasting the information. - Add note that Planar pixel formats deviate from PFNC since they don't put an underscore before the Planar suffix. - Fix typo in string designator for RGB565p and BGR565p. - Define the acronym the first time it is used in the text. - Eliminate duplicate of [CR3-3cd]. - Provide hyperlinks to all RFC references. - Remove [R10-1s] which is a duplicate of [R8-1s]. - Move [R8-3ca] to Control Channel chapter. Move [R8-2cd] to Message Channel chapter. Move [CR10-7s] to Multiple Link Configuration Section. - Explicitly state *block_id/block_id64* and *packet_id/packet_id32* throughout the text. - Fix H.264 *packetization_mode* field to 2 bits. - Change YUV422_8_YUYV to YUV422_8, as per PFNC newly introduced component ordering. - Fix new YCbCr pixel format by adding the # of bits per component (8 bits) to align to PFNC. - Align Planar pixel format name to PFNC by adding an underscore before Planar.
2.0 RC5	2011-10-26	Eric Carey, Teledyne DALSA	- Clearly state that for Multi-zone Image, the chunk tags associated to the first zone are put in the additional zone. Confirm that 2 zones are necessary even if Multi-zone Image itself only occupies a single zone. - Remove erroneous payload formats 0x4004 and 0x4005. - Indication that instance key of TXT record typically starts at index 0. - Fix many typos and inconsistencies from external reviewers. - Generalisation of text referring to *block_id* to the more generic "block ID". - Add note to clarify pixel padding at a zone boundary. - End-of-zone flag is don't care for single zone images. - Fix t_{GEV} upper limit to 2^64 – 1 in IEEE 1588 section. - For all YCbCr 4:1:1 pixel formats, change Cb and Cr indexes to use co-sited positioning, as specified in PFNC. - Fix bit mismatch for RGB12V1Packed pixel format (10 bits were illustrated originally). - Use Device MAC Address register instead of Device MAC register. - Stream Channel Capability register only mandatory for GVSP transmitter or receiver. - Change PENDING field of GVCP capability register to PENDING_ACK. - Revert back GVSP_PIX_BAYERxxx to GVXP_PIX_BAYxxx. - Indicate how to construct unique host_name for mDNS. - New payload types introduced by GEV 2.0 are only supported in Extended ID mode (EI = 1). - Rename "Stream Channels Capability" register to "GVSP Capability" register to avoid name clash with SCCx. This aligns with "GVCP Capability" register. - Use packet_id32 in the examples of Multi-zone Image payload format.

			- Generalize last packet ID description below Figure 16-13 to cover both 24 bits (*last_packet_id*) and 32 bits (*last_packet_id32*) cases. - Revert to [R7-12ca] from GEV 1.2. req_id MUST increment, but this is not necessarily by 1. - Adjusted the "Number of Network Interface" register to account for virtual interfaces under LAG. This is not anymore the number of physical interfaces, but simply the number of interfaces. - Reword first paragraph of Zeroconf section. - With mDNS, this first instance must use the GVCP port 3956 to allow backward compatibility. - Remove duplicate requirement for Scheduled Action Command when the time tag is past. Add a GEV_STATUS_ACTION_LATE status code to be returned in this situation. This code does not need to be enabled in the GVCP Configuration register since application using *action_time* is GEV 2.0 ready. - Use GEV_STATUS_BAD_ALIGNMENT for GVCP read or write access not aligned to 32-bit. - Change format of PACKETRESEND_CMD in Extended ID mode to leverage existing reserved fields. - In Standard ID mode, revert back to GEV 1.2 behavior (each packet must be resent separately). - padding_y is possible with variable frame size. - Add graphic in Multi-zone Image with ROI where data from a sensor tap is shared across two different zones. - Change the following Pixel Formats to reflect the component order of GEV 1.2: YCbCr8_CbYCr, YCbCr411_8_CbYYCrYY, YCbCr601_8_CbYCr, YCbCr601_411_8_CbYYCrYY, YCbCr709_8_CbYCr and YCbCr709_411_8_CbYYCrYY. - Define the Test Packet to be neutral to Extended ID mode. Eliminate [R24-27st] since the layout is clearly provided by [R24-28st]. - GEV_STATUS_WRITE_PROTECT only applies to full register that cannot be written to. - Adjust IEEE 1588 timestamp structure to represent the full 80 bits (epoch_number, seconds, nanoseconds). - Restore [O14-44ca] to facilitate interoperability with devices based on GEV 1.x. - Add capability bit in SCCx to indicate that multi-zone image stream registers are available. - Adjust Bayer tile definition to represent the pixel format of the full image with no ROI. Used to be the pixel format of the sensor. - Update list of participants to reflect latest technical meeting. - Augment the tables for each payload format packet to list all the "white" fields and provide proper reference. - GEV timestamp is a 64-bit signed value when using IEEE 1588 as time reference. - Renumber requirements following GEV 2.0 scheme. - Update Requirements and Objectives table. - Update Deprecated Requirements and Objectives table. - Errata for pixel formats 0x0220001C and 0x0220001D where GEV 1.2 introduced a different format by swapping the R and B components. Revert to definition from GEV 1.0/1.1. Clearly mark the errata for both of these pixel formats. - Added following pixel formats: YCbCr422_8_CbYCrY, YCbCr601_422_8_CbYCrY, YCbCr709_422_8_CbYCrY.
2.0 RC6	2011-11-04	Eric Carey, Teledyne DALSA	- Added liability disclaimer. - Removed line numbers. - Adjust suffix for the following requirements: [CR-010c], [CO-011st], [CO-012cd], [CR-204st], [CR-205st], [CO-206st], [CR-207sr], [CR-208st], [CO-202st], [CR-203st], [R-262s], [R-265s], [CR-275s], [R-282s], [CR-488cd], [CR-489cd], [R-490st], [CR-491cd], [CR-492cd], [CO-493cd], [CO-494cd], [CO-495cd], [CR-496cd], [CR-497cd]. Update requirement matrix accordingly.
2.0	2011-11-21	Eric Carey, Teledyne DALSA	Official release of GigE Vision 2.0 specification. - Removed RC6 tag.

			- Inserted page breaks in order to make the document more printer friendly.
2.0.03 draft A with errata	2013-02-06	Eric Carey, Teledyne DALSA	The following errata have been corrected in this version: -Multiple typos throughout the document. -Renumbered duplicate requirement numbers for [CR-401s], [CR-404s] and [CR-407s] to [CR-532s], [CR-533s] and [CR-534s] respectively. -Removed sentence in section 17.2 suggesting that device-specific data is limited to 540 bytes as this is not true in all situations. -Added a note in section 3.3.3 and section 23.1.4 that all physical MAC part of the same aggregator use the same PAUSE configuration since they are presented as a single logical link. This was not clear before. -Removed inconsistency in endianess for the manufacturer-specific registers. This impacts [R-415cd], [R-416cd] and bit 0 of Device Mode register. Add a note that endianess of the manufacturer-specific registers is not defined by this text. -Clarification for the 2nd interpretation of [R-015cd] to follow more closely DHCP RFC3927 for possible IP address loss upon renewal and retransmission delay after 2 retransmission failures. Adjustments to Figure 4-1 to reflect that 2nd interpretation. -Augmented [R-013cd] to include Ethernet link negotiation which happens when Ethernet cable is unplugged then re-plugged. This scenario was missing. -Eliminate [R-054ca] since duplicate of [R-073ca]. Convert [R-073ca] to an objective for an application heartbeat frequency parameter. Adjust surrounding text to reflect that change. Therefore, the device heartbeat timeout register becomes the reference throughout the text to control the GVCP connection heartbeat. -Clarify the exclusive access mode provide a certain level of protection, not security. And this relates to primary control channel. -In section 25.1, clarify that chunk_data_payload_length represents the sum for all the chunks. -In section 26.3, replace the note about device-specific pixel format with a more comprehensive paragraph that references PFNC and explains how to create custom pixel formats. -Fix IEEE 1588 formulas and timestamp structure to match version 2 of IEEE 1588 (IEEE 1588-2008) instead of version 1 (IEEE 1588-2002). This eliminates mismatch between version 1 and 2 of IEEE 1588. This has the side effect of removing negative timestamps, though the GigE Vision timestamp remains 64-bit signed to maintain compatibility with GigE Vision 2.0.02. -Fix typo in status code values for GEV_STATUS_NO_REF_TIME, GEV_STATUS_PACKET_TEMPORARILY_UNAVAILABLE, GEV_STATUS_OVERFLOW and GEV_STATUS_ACTION_LATE. They now follow the sequential numerical value and have the *Severity* bit set. -Add note below [R-152cd] to recommend sending a gratuitous ARP anytime a new IP address is assigned to the device. This enables remote hosts to update their ARP cache. -In the Requirement Reference Table, put [O-428cd] instead of [R-428cd].
2.0.03 draft B with errata	2013-03-13	Eric Carey, Teledyne DALSA	The following errata have been corrected in this version: -For ACTION_CMD, correct that bit 4 to 7 of the flag field must use the standard definition from GVCP header. They were marked as reserved. -Indicate the GEV_STATUS_INVALID_PARAMETER can be used when too many parameters are provided in the command. -Clarify that application cannot use concatenation for READREG and WRITEREG when accessing a device if this device has the Concatenation bit (bit 31) cleared in the GVCP Capability register. -Add text to Link Speed, Control Switchover Key, Primary Application Port, Primary Application IP Address and Discovery ACK delay to indicate an application can query the GVCP Capability register to determine if these optional registers are supported. -Indicate in [R-076cd] that bit 31 of the GVCP Configuration register can be used to disable the heartbeat. -In section 14.4.1, highlight it is bit 10 of the GVCP Capability register that must be checked.
2.0.03	2013-04-16	Eric Carey, Teledyne DALSA	Official release of GigE Vision 2.0.03 (with errata). -Adjust page formatting

			-Regenerate table of content