INTERNATIONAL STANDARD

ISO 17215-3

First edition
2014-04-15

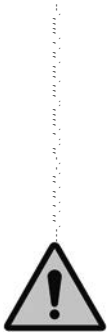
Road vehicles — Video communication interface for cameras (VCIC) —

Part 3: Camera message dictionary

Véhicules routiers — Interface de communication vidéo pour caméras (ICVC) —

Partie 3: Dictionnaire de message de caméra

Reference number
ISO 17215-3:2014(E)



COPYRIGHT PROTECTED DOCUMENT

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Contents

Page

Foreword ..... **iv**

Introduction ..... **v**

1 Scope **1**

2 Normative references **2**

3 Terms and definitions, symbols, and abbreviated terms **2**
3.1 Terms and definitions 2
3.2 Abbreviated terms 3

4 Conventions **3**

5 Overview of ISO 17215 ..... **3**
5.1 General 3
5.2 Document overview and structure 3
5.3 Open Systems Interconnection (OSI) model 4
5.4 Document reference according to OSI model 4

6 Camera application interface (OSI layer 7) **5**
6.1 Specific properties 5
6.2 API principles 6
6.3 API data types 7
6.4 API Return codes 7
6.5 API enumerations 7
6.6 API structures 11
6.7 API reference 26
6.8 Programming model for SOME/IP 40
6.9 PDU examples for some/IP 45

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular the different approval criteria needed for the different types of ISO documents should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation on the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the WTO principles in the Technical Barriers to Trade (TBT) see the following URL: Foreword - Supplementary information

The committee responsible for this document is ISO/TC 22, *Road vehicles,* Subcommittee SC 3, *Electrical and electronic equipment*.

ISO 17215 consists of the following parts, under the general title *Road vehicles — Video communication interface for cameras (VCIC):*

— *Part 1: General information and use case definition*

— *Part 2: Service discovery and control*

— *Part 3: Camera message dictionary*

— *Part 4: Implementation of communication requirements*

Introduction

Driver assistance systems are more and more common in road vehicles. From the beginning, cameras were part of this trend. Analogue cameras were used in the beginning because of lower complexity of the first systems. With increasing demand for more advanced functionality, digital image processing has been introduced. So-called one box design cameras (combining a digital image sensor and a processing unit) appeared in the vehicles.

Currently, the market demands such systems with multiple functions. Even different viewing directions are in use. It seems to be common sense that 6 up to 12 cameras in a single vehicle will be seen in the next future. Out of this and the limitation in size, power consumption, etc. it will lead to designs where the cameras are separated from the processing unit. Therefore, a high performance digital interface between camera and processing unit is necessary.

This International Standard has been established in order to define the use cases, the communication protocol, and the physical layer requirements of a video communication interface for cameras, which covers the needs of driver assistance applications.

The video communication interface for cameras

— incorporates the needs of the whole life cycle of an automotive grade digital camera,

— utilizes existing standards to define a long-term stable state-of-art video communication interface for cameras usable for operating and diagnosis purpose,

— can be easily adapted to new physical data link layers including wired and wireless connections by using existing adaption layers, and

— is compatible with AUTOSAR.

This part of ISO 17215 is related to the general information and use case definition. This is a general overview International Standard which is not related to the OSI model.

To achieve this, it is based on the Open Systems Interconnection (OSI) basic reference model specified in ISO/IEC 7498-1 and ISO/IEC 10731, which structures communication systems into seven layers. When mapped on this model, the protocol and physical layer requirements specified by this International Standard, in accordance with Table 1 are broken into following layers:

— application (layer 7), specified in ISO 17215-3;

— presentation layer (layer 6), specified in ISO 17215-2;

— session layer (layer 5), specified in ISO 17215-2;

— transport protocol (layer 4), specified in ISO 17215-4, ISO 13400-2;

— network layer (layer 3), specified in ISO 17215-4, ISO 13400-2;

— data link layer (layer 2), specified in ISO 17215-4, ISO 13400-3;

— physical layer (layer 1), specified in ISO 17215-4, ISO 13400-3.

Table 1 — Specifications applicable to the OSI layers

Applicability	OSI 7 layers	Video communication interface for cameras		Camera diagnostics
Seven layers according to ISO 7498–1 and ISO/IEC 10731	Application (layer 7)	ISO 17215-3		
	Presentation (layer 6)	ISO 17215-2		
	Session (layer 5)	ISO 17215-2		
	Transport (layer 4)	ISO 17215-4	Other future interface standards	ISO 13400-2
	Network (layer 3)			
	Data link (layer 2)	ISO 17215-4		ISO 13400-3
	Physical (layer 1)			

ISO 17215-1 has been established in order to define the use cases for vehicle communication systems implemented on a video communication interface for cameras; it is an overall International Standard not related to the OSI model.

ISO 17215-3 covers the application layer implementation of the video communication interface for cameras; it includes the API.

ISO 17215-2 covers the presentation layer implementation of the video communication interface for cameras.

ISO 17215-4 is the common standard for the OSI layers 1 to 4 for video communication interface for cameras. It complements ISO 13400-2 and ISO 13400-3 and adds the requirement for video transmission over Ethernet.

ISO 17215-2 and ISO 17215-3 (OSI layer 5 to 7) services have been defined to be independent of the ISO 17215-4 (OSI layer 1 to 4) implementation. Therefore, ISO 17215-4 could be replaced by other future communication International Standard.

Road vehicles — Video communication interface for cameras (VCIC) —

Part 3:
Camera message dictionary

1 Scope

This part of ISO 17215 specifies the standardized camera messages and data types used by a VCIC camera (OSI Layer 7).

The scope of the camera application interface (API) and its context are shown in Figure 1.

Applications hosted on ECUs want to communicate with one or more cameras (e.g. "Ask camera for parameters."). If the applications can use standardized services supported by the cameras (API layer 7), the development of a vision application should be independent of the camera used. The services can be implemented by general libraries.

The definition of streaming data are not an issue of this API.

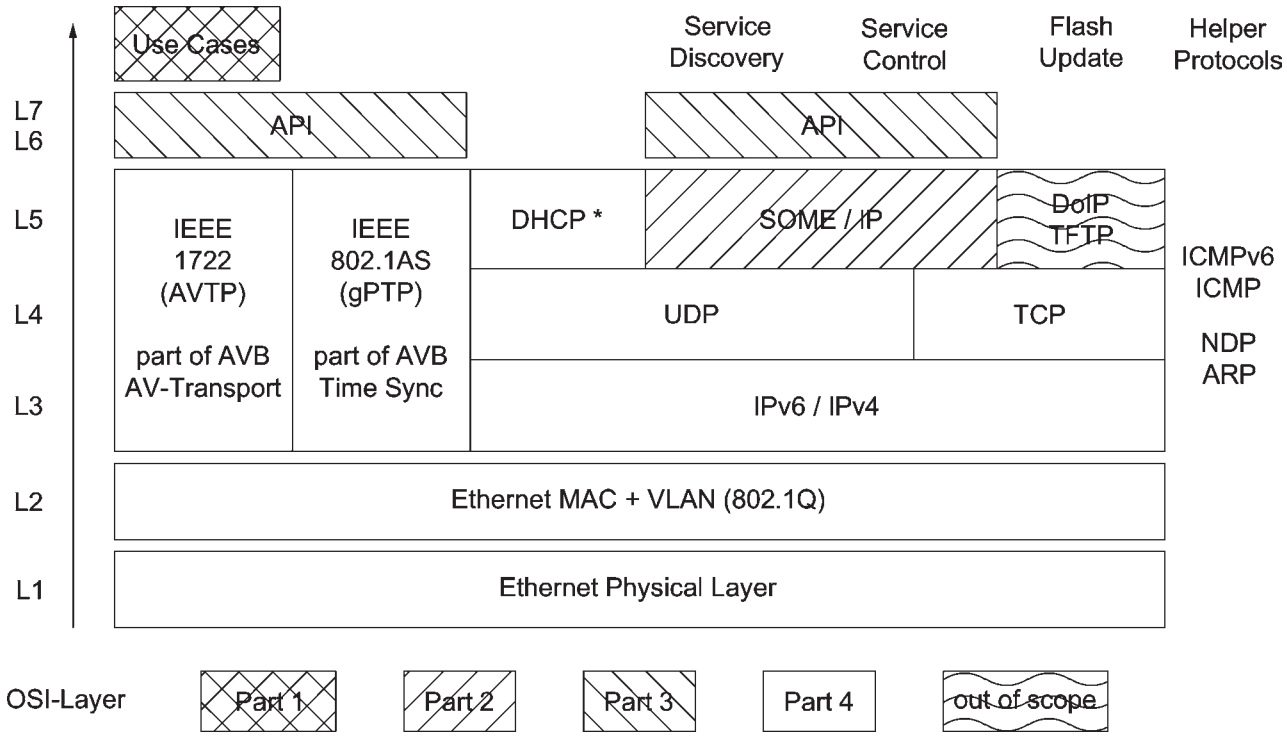


Figure 1 — Overview of ISO 17215

The general terminology defined in ISO 17215-1 is also used in this part of ISO 17215.

2 Normative references

The following documents, in whole or in part, are normatively referenced in this document and are indispensable for its application. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO°7498-1, *Information processing systems — Open systems interconnection — Basic reference model*

ISO/IEC 10731, *Information technology — Open Systems Interconnection — Basic Reference Model — Conventions for the definition of OSI services*

ISO 17215 (all parts), *Road vehicles —Video communication interface for cameras (VCIC)*

3 Terms and definitions, symbols, and abbreviated terms

3.1 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

3.1.1
camera identification number
individual camera identification number that identifies the supplier, camera type, and individual camera (e.g. MEE-32140-194565432-DD2RT supplier, camera type, serial number)

3.1.2
camera register
internal HW registers of the camera

3.1.3
extrinsic parameters
denotes the coordinate system transformations from 3D world (vehicle) coordinates (m,°) to 3D-camera coordinates (m,°)

3.1.4
focal length
distance over which initially collimated rays are brought to a focus

3.1.5
frame rate
update rate per time of camera images

3.1.6
global shutter
exposure that exposes all pixels at the same time

3.1.7
histogram
type of histogram that acts as a graphical representation of the tonal distribution in a digital image

3.1.8
intrinsic camera parameters
denote the coordinate system transformations from 3D camera (m) to 2D pixel coordinates (pixel)

3.2 Abbreviated terms

Term	Description
API	Application Programming Interface
AEC	Automatic Exposure Control
AGC	Automatic Gain Control
DAS	Driver Assistance System
ECU	Electronic Control Unit
HDR	High Dynamic Range
HMI	Human Machine Interface
ISO	International Organization for Standardization
LDR	Low Dynamic Range
MAC	Media Access Control
OSI	Open Systems Interconnection
PSE	Persistent storage entry
ROI	Region of Interest, i.e. sub-part of overall image
RPC	Remote Procedure Call

4 Conventions

This International Standard is based on the conventions specified in the OSI service conventions (ISO/IEC°10731) as they apply for physical layer, protocol, network, and transport protocol and diagnostic services.

5 Overview of ISO 17215

5.1 General

This International Standard has been established in order to implement a standardized video communication interface for cameras on a communication data link.

The focus of this International Standard is using existing protocols.

— Figure 1 specifies the relation to the other parts of this International Standard.

— Figure 2 specifies the relation of this International Standard to existing protocols.

5.2 Document overview and structure

This International Standard consists of a set of four sub-documents, which provide all references and requirements to support the implementation of a standardized video communication interface for cameras according to the standard at hand.

— ISO 17215-1 provides an overview of the document set and structure along with use case definitions and a common set of resources (definitions, references) for use by all subsequent parts.

— ISO 17215-2 specifies the discovery and control of services provided by a VCIC camera.

— ISO 17215-3 specifies the standardized camera messages and data types used by a VCIC camera (OSI Layer 7).

— ISO 17215-4 specifies standardized low-level communication requirements for implementation of the physical layer, data link layer, network layer, and transport layer (OSI Layers 1 to 4).

5.3 Open Systems Interconnection (OSI) model

This International Standard is based on the Open Systems Interconnection (OSI) basic reference model as specified in ISO/IEC 7498 which structures communication systems into seven layers.

All parts of this International Standard are guided by the OSI service conventions as specified in ISO/IEC 10731 to the extent that they are applicable to diagnostic services. These conventions define the interaction between the service user and the service provider through service primitives.

The aim of this subclause is to give an overview of the OSI model and show how it has been used as a guideline for this part of ISO 17215. It also shows how the OSI service conventions have been applied to this International Standard.

The OSI model structures data communication into seven layers called (from top to bottom) the application layer (layer 7), presentation layer, session layer, transport layer, network layer, data link layer, and physical layer (layer 1). A subset of these layers is used in this International Standard.

The purpose of each layer is to provide services to the layer above. The active parts of each layer, implemented in software, hardware, or any combination of software and hardware, are called entities. In the OSI model, communication takes place between entities of the same layer in different nodes. Such communicating entities of the same layer are called peer entities.

The services provided by one layer are available at the Service Access Point (SAP) of that layer. The layer above can use them by exchanging data parameters

This International Standard distinguishes between the services provided by a layer to the layer above it and the protocol used by the layer to send a message between the peer entities of that layer. The reason for this distinction is to make the services, especially the application layer services and the transport layer services, reusable also for other types of networks than the video communication interface for cameras. In this way, the protocol is hidden from the service user and it is possible to change the protocol if demanded by special system requirements.

5.4 Document reference according to OSI model

Figure 2 illustrates the document references.

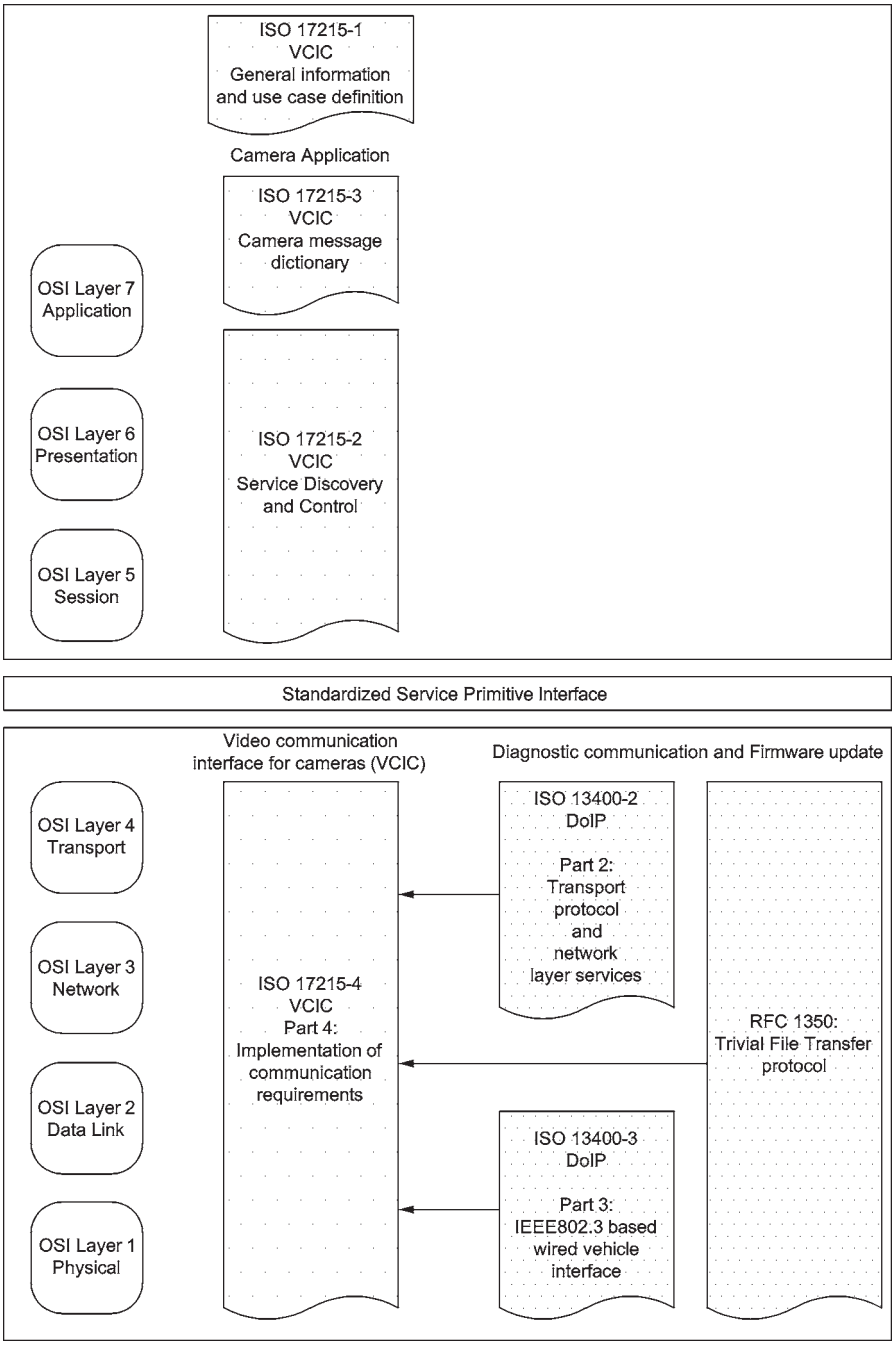


Figure 2 — Document reference according to OSI model

6 Camera application interface (OSI layer 7)

6.1 Specific properties

In the automotive environment, the network topologies are semi-static and the characteristics of all components, including cameras, are bound to a specific car platform design. Components and characteristics of components not included in the design need not to be supported.

There is no requirement for a least common video mode. The minimum compatibility requirement is to recognize the reason of unexpected behaviour. Compatibility is ensured during the design phase. Incompatibility will be detected during development, assembling, or repair of a car.

Consequently, the standard specifies the interface to an automotive camera, but doesn't specify the characteristics of automotive cameras.

For instance, no mandatory or standard video formats are specified. However, all provisions are made to implement standard video formats.

6.2 API principles

The camera API consists of a variety of data structures describing video modes, camera controls, and stored items and a set of API functions.

The API is independent of specific programming languages. It is an abstract list of data structures and functions to be offered by all implementations. A fundamental principle of all camera API functions is the usage of remote procedure calls (RPC).

The addressing of multiple cameras can be expressed by:

Camera_function = f { camera instance, api_function { MethodID, arg1, .. argn } }.

The implementation depends highly on the programming language used. Therefore, this part of ISO 17215 only covers the api_function itself.

Camera data structures can be read, written, and deleted using the associated camera API functions.

Camera API functions can be grouped by the mechanism they are using in

— set, get, and erase functions, and

— subscribe and unsubscribe functions.

All API functions starting with set, get, and erase are used to modify the cameras data structures and the underlying functionality, for instance setting the exposure time of the imager. They are using a request/response mechanism. A request is followed by a single response.

Functions are generally identified by its Method ID.

The Method IDs are specified in 6.5.1.

All API functions starting with unsubscribe/subscribe are used to acquire cyclically data from the camera, for instance, a video stream. They are using subscribe/notification mechanism. After subscribing an event, multiple notification packages will follow.

Events are identified by its respective Eventgroup ID.

The Eventgroup IDs are specified in 6.5.2.

The SOME/IP protocol defined in ISO 17215-2 provides the mechanism for such a RPC-based implementation.

Camera functions and the associated structures can be grouped in functional context in

— general camera functions (MethodID 0x0001 – 0x00FF),

— video format functions (MethodID 0x0101 – 0x01FF),

— image control functions (MethodID 0x0201 – 0x02FF), and

— imager functions (MethodID 0x0301 – 0x03FF).

6.2.1 Image cropping and windowing

The definition of the image windows are shown below.

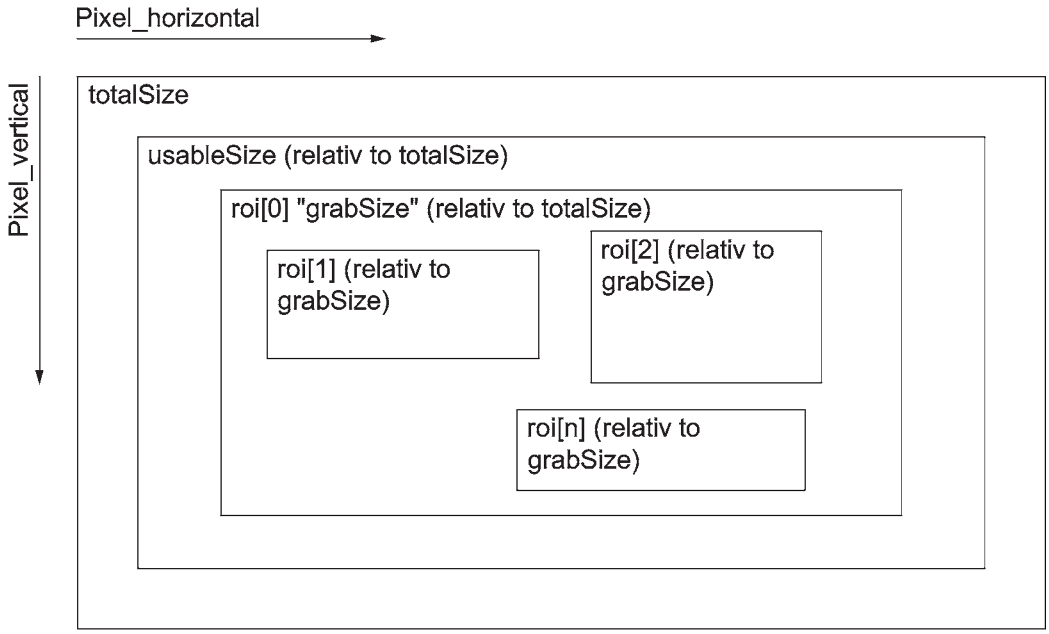


Figure 3 — Image cropping and windowing

6.3 API data types

All used data types are specified in 6.3.1 of ISO 17215-2

6.4 API Return codes

Each function of the API returns a byte (8 bits) to signalize the status of the operation.

— The return codes (0x00 to 0x1F) are defined in 6.1.1.7 of ISO 17215-2.

— The return codes (0x20 to 0x3F) are function-specific and are described in 6.5.3.

6.5 API enumerations

6.5.1 enumeration eMethodID

Purpose: This enumeration is used to identify the methods supported by the camera.

This enumeration is based on uint16 data type.

Table 2 — enumeration eMethodID

Name	Value	Description
getDataSheet	0x0001	returns the datasheet of the camera
getCamStatus	0x0002	returns the current status of the camera
setCamMode	0x0003	start, stops, and restarts the camera application in the camera
setCamExclusive	0x0011	assigns the control of the camera exclusively to the requesting client
eraseCamExclusive	0x0019	removes the exclusive control look for the requesting client
setHostParameters	0x0022	sets the host parameters using persistent storage entries
getHostParameters	0x0024	the requested host parameter by reading persistent storage entries
eraseHostParameters	0x0029	forces the camera to erase the requested host parameters addressed by the PSE ID

Table 2 *(continued)*

Name	Value	Description
setRegionOfInterest	0x0101	sets the parameter for a region of interest addressed by index
setRegionsOfInterest	0x0102	sets the parameter for all supported regions of interest
getRegionOfInterest	0x0103	returns the parameter for region of interest addressed by ROI index
getRegionsOfInterest	0x0104	returns the parameter for all supported region of interest
eraseRegionOfInterest	0x0109	erases all parameter for the requested ROI
setVideoFormat	0x0111	sets the video format for a ROI addressed by ROI index
getVideoFormat	0x0113	reads the parameter of the current video format, addressed by the ROI index
eraseVideoFormat	0x0119	erases all video format parameter for the requested ROI
setHistogrammFormat	0x0121	the histogram format parameter for a ROI addressed by ROI index
getHistogrammFormat	0x0123	the parameter of the current histogram format, addressed by the ROI index
eraseHistogrammFormat	0x0129	erases all histogram format parameter for the requested ROI
SubscribeROIVideo	0x0131	starts the transmission of a video stream for the requested ROI
UnSubscribeROIVideo	0x0132	stops the video streaming for the requested ROI
SubscribeROIHistogram	0x0133	starts the transmission of the histograms for the requested ROI
UnSubscribeROIHistogram	0x0134	the transmission of the histograms for the requested ROI
setCamControl	0x0201	sets the parameter for a camera control addressed by the control index
setCamControls	0x0202	sets the parameters for all camera controls
getCamControl	0x0203	returns the current parameter for a camera control addressed by the control index
getCamControls	0x0204	returns all current camera control parameter
setCamRegister	0x0301	writes the content of a register of the cameras imager addressed by physical register address
setCamRegisters	0x0302	writes (atomic access) the content of a register block of the camera imager
getCamRegister	0x0303	reads the content of a register of the camera imager address by the physical register address
getCamRegisters	0x0304	reads the content of a register block of the camera imager
setUsedRegisterSet	0x0305	forces the camera to write the imager register set stored in the requested PSE to the imager

6.5.2 enumeration eEventGroupType

Purpose: This enumeration is used to identify the event groups supported by the camera.

This enumeration is based on uint16 data type.

Table 3 — enumeration eEventGroupType

Name	Value	Description
AllEvents	0xB000	Event group ID for all events
VideostreamEvents	0x9000	Event group ID for video streams
HistogrammEvents	0xA000	Event group ID for histograms

6.5.3 enumeration eCamErrorCodes

Purpose: This enumeration defines the camera API specific return codes. The values are in the range of 0x20 to 0x3F.

Table 4 — enumeration eCamErrorCodes

Name	Value	Description
E_LOCKED_BY_FOREIGN_INSTANCE	0x20	Camera service is already locked by another client
E_LOCK_ EXPIRED	0x21	The camera lock has expired
E_NOT_LOCKED	0x22	Camera is not locked
E_INVALID_PS_ENTRY	0x24	The requested PSE ID is unknown
E_INVALID_PS_OPERATION	0x25	The requested PSE operation is not allowed, e.g. store on a RO PSE
E_INVALID_PS_DATA	0x26	The PSE contains a CRC16 error
E_NO_MORE_SPACE	0x27	No more space available to store the PSE
E_INVALID_ROI_INDEX	0x30	The requested ROI is out of range
E_INVALID_ROI_NUMBER	0x31	The requested number of ROIs is out of range, defined by sDatasheet.numOfRegionOfInterest
E_INVALID_VIDEO_FORMAT	0x32	Invalid value in video format
E_INVALID_HISTOGRAM_FORMAT	0x33	Invalid value in histogram format
E_INVALID_CONTROL_INDEX	0x35	The requested camControlIndex is out of range or not supported by the camera
E_INVALID_CONTROL_MODE	0x36	The requested control mode is not supported by the camera
E_INVALID_CONTROL_VALUE	0x37	The requested control value is out of the range, defined in sCamControl
E_INVALID_REGISTER_ADDRESS	0x38	The register address is not supported by the imager
E_INVALID_REGISTER_VALUE	0x39	The value for the given register address is not supported by the imager
E_INVALID_REGISTER_OPERATION	0x3A	The requested operation (read or write) for the given register address is not supported by the imager

6.5.4 enumeration eCameraMode

Purpose: This enumeration type defines the camera operating modes which can be set by using the setCamMode method.

Table 5 — enumeration eCameraMode

Name	Value	Description
StartCameraService	1	The camera application shall be started
StopCameraService	2	The camera application shall be stopped. Power-intensive components like imager should be switched off.
ReStartCameraService	3	The camera application shall restart. When restarting, the camera application shall use the PSE 'UseAtBootTime'
StopCamera	4	The camera device shall enter standby mode (uC OFF, Communication-Controller OFF) and stores the settings. This state can only left by repowering or wake up on LAN

6.5.5 enumeration eControlIndex

Table 6 — enumeration eControlIndex

Name	Value	Description
exposureTime	1	Controls the integration time
Brightness	2	Controls the black level offset

Table 6 *(continued)*

Name	Value	Description
Gain	3	Controls the gain circuits of the camera
Hue	4	Controls the hue circuits of the camera
Saturation	5	Controls the saturation circuits of the camera
Gamma	6	Controls the gamma correction circuit of the camera
whiteBalance	7	Controls the white balance circuit of the camera
synchronization	8	Controls the synchronization method
Heater	9	Controls the method to use the heater

6.5.6 enumeration eControlSupportedModes

Purpose: This enumeration defines the valid values for the supported modes of operation of camera controls.

Table 7 — enumeration eControlSupportedModes

Name	Value	Description
StatnotSupported	0	Control not supported
StatRW_nA_nO	1	Read/write, no continuous automatic, no OnePush automatic
StatRO_nA_nO	3	Read only, no continuous automatic, no OnePush automatic
StatRO_A_nO	5	Read only, continuous automatic, no OnePush automatic
StatRW_A_nO	7	Read/write, continuous automatic, no OnePush automatic
StatRO_nA_O	9	Read only, no continuous automatic, OnePush automatic
StatRW_nA_O	11	Read/write, no continuous automatic, OnePush automatic
StatRO_A_O	13	Read only, continuous automatic, OnePush automatic
StatRW_A_O	15	Read/write, continuous automatic, OnePush automatic

Manual mode: The written value will be taken.

One push automatic mode: The built-in automatic control works until the automatic reaches the desired value. Then, the mode switches automatically back to manual mode.

Automatic mode: Built-in automatic control works until the mode of operation changes.

6.5.7 enumeration eControlMode

Purpose: This enumeration type is used to set the mode of operation of a certain camera control.

Table 8 — enumeration eControlMode

Name	Value	Description
ModeManual	2	Manual mode, value not modified
ModeManualModified	3	Manual mode, value modified
ModeAuto	4	Continuous automatic mode
ModeOnePush	8	onePush automatic mode

6.5.8 enumeration ePersistentStorageID

Purpose: Persistent storage entry identifiers are used to identify the payload of the Set/Get/Restore host parameter functions. The four LSB bits are used to allow up to 16 entries of the same type.

NOTE Predefined persistent storage entry identifiers are in the range of 0x0000 to 0x7FFF. The manufacturer specific identifiers are in the range of 0x8000 to 0xFFFF.

Table 9 — enumeration ePersistentStorageID

Name	Value	Description
UseAtBootTime	0x0000	The payload consists of a list of persistent storage entries. The structure sPersistentEntryList shall be used. Only existing entries with the ID ROISetting and ImagerRegisterBlock are valid.
ImagerCharacteristic	0x0010	The payload consists of imager type specific information. The structure sImagerCharacteristic shall be used and be implemented as read only entry.
Defectpixelmap	0x0020	The payload consists of the imager exemplar specific defect pixel map. The structure sPixelMap shall be used and might be implemented as read only entry.
intrinsicCamParam	0x0030	The payload consists of the intrinsic calibration parameter. The struct sIntrinsicCamParam shall be used and might be implemented as read only entry.
ExtrinsicCamParam	0x0040	The payload consists of the extrinsic calibration parameter. The struct sExtrinsicCamParam shall be used.
ROISetting	0x1yyx	The payload consists of the parameter set for a ROI. The structure sRegionOfInterest shall be used, y = roiIndex, x = number of entry belonging to the roiIndex.
ImagerRegisterBlock	0x20xx	The payload consists of an block of imager registers,. The structure sImagerRegisterBlock shall be used xx = number of entry.

Each ROI has its own ID, and up to 16 PSEs for each ROI are allowed. The ID shall fit in the range given by numOfRegionOfInterest in the camera data sheet. The PersistentStorageID 0x1012, for example, identifies the second entry for the ROI with the index 1. Up to 256 PSEs for imager register parameter are allowed.

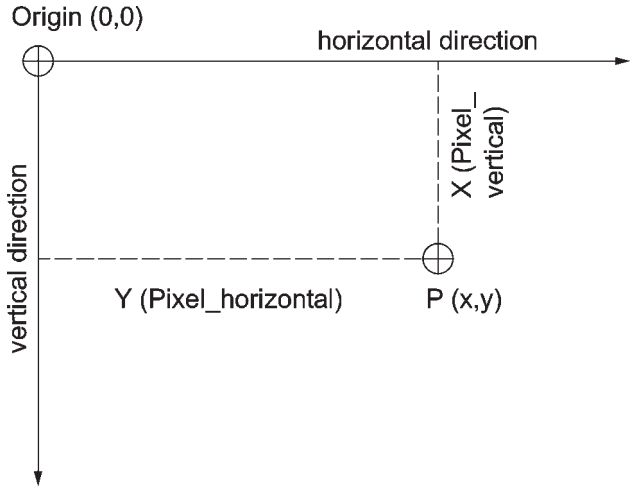
The boot parameter entry contains a list of PSEs. Only existing entries with the ID ROISetting and ImagerRegisterBlock are valid for this PSE ID. If this entry exists, the camera shall use content of the listed PSE as initial parameter set.

6.6 API structures

6.6.1 structure sPixelPosition

Purpose: Definition of the position of a single pixel relative to a window defined in 6.2.1. The effects of the optical system (e.g. lens) are included.

Figure 4 illustrates the single pixel evaluation.



NOTE Looking along the optical axis of camera to the object.

Figure 4 — sPixel

Table 10 — structure sPixelPosition

Name	Type	Description
X	unit16	Number of pixels in vertical direction
Y	unit16	Number of pixels in horizontal direction

6.6.2 structure sPixelMap

Purpose: This structure defines a group of independent pixels. This structure is used, for instance, to report the position of defect pixel in the imager.

Table 11 — structure sPixelMap

Name	Type	Description
numberOfEntries	unit32	Number of used entries counted from the first entry
pixelPositions	sPixelPosition[numberOfEntries]	Array of pixel positions

6.6.3 structure sRectangle

Purpose: Definition of a rectangular area using two pixels coordinates (see Figure 5).

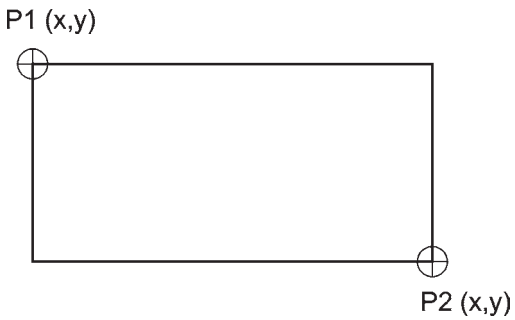


Figure 5 — sRectangle

Table 12 — structure sRectangle

Name	Type	Description
P1	sPixelPosition	Left upper pixel of the rectangle
P2	sPixelPosition	Right lower pixel of the rectangle

6.6.4 structure sImageDimension

Purpose: Definition of the image dimension.

Table 13 — structure sImageDimension

Name	Type	Description
Width	uint16	Width in pixel
Height	unit16	Height in pixel

6.6.5 structure sImagerRegister

Purpose: Definition of an address/value pair of imager register.

Table 14 — structure sImagerRegister

Name	Type	Description
regAddress	uint16	Address of the register
regValue	unit16	Value of the register

Each register is addressed by a 16-bit number. Each register is 16 bits wide. Smaller physical register shall be LSB bound.

6.6.6 structure sImagerRegisterBlock

Purpose: Definition of a block of address/value pairs of imager register.

Table 15 — structure sImagerRegisterBlock

Name	Type	Description
regCount	uint16	Number of registers to be written
imagerRegister	sImagerRegister[regCount]	Array of regCount register address and value pairs of the imager

6.6.7 structure sImagerCharacteristic

Purpose: Defines the characteristics of the camera.

Table 16 — structure sImagerCharacteristic

Name	Type	Description
totalSize	sImageDimension	Maximum size of the imagers pixel array, P1 is always 0,0; see 6.2.1
usableSize	sImageDimension	Usable size of the imagers pixel array, coordinates are relative to total size; see 6.2.1
pixelSizeX	uint16	Horizontal pixel size [µm] = pixelSizeX/100
pixelSizeY	uint16	Vertical pixel size [µm] = pixelSizeY/100

Table 16 *(continued)*

Name	Type	Description
shutterType	unit8	0: global shutter 1: rolling shutter
sensorIdentifier	uint8[32]	Sensor identifier UTF-8 encoded

6.6.8 struct sIntrinsicCamParam

Purpose: The intrinsic camera parameters determine the projection from 3-D camera coordinates to 2-D pixel coordinates. The intrinsic parameters of the camera can be read out by application software and can be used for back-projection form a 2-D image position to a line in 3-D space.

The sIntrinsicCamParam data structure containing the intrinsic camera parameters is defined in Table 17.

Table 17 — struct sIntrinsicCamParam

Name	Type	description
distortionModel	unit8	This parameter indicates in later editions of this International Standard the distortion model used. It is fixed to 0 in this part of ISO 17215.
pad1	unit8	Unused
pad2	unit8	Unused
pad3	unit8	Unused
principalPointX	float64	X coordinate of the principal point [pixel]
principalPointY	float64	Y coordinate of the principal point [pixel]
ScaleX	float64	Dimensionless scale factor (distance, measured in units of the pixel width of the imager, between imager plane and the imager-side projection centre of the camera)
scaleY	float64	Dimensionless scale factor (distance, measured in units of the pixel height of the imager, between imager plane and the imager-side projection centre of the camera)
k_1	float64	Radial distortion coefficient 1
k_2	float64	Radial distortion coefficient 2
k_3	float64	Radial distortion coefficient 3
k_4	float64	Radial distortion coefficient 4
k_5	float64	Radial distortion coefficient 5
k_6	float64	Radial distortion coefficient 6
P_1	float64	Tangential distortion coefficient 1
P_2	float64	Tangential distortion coefficient 2

Using the constants contained in the above data structure, a 2-D vector (u, v) can be computed from a given 3-D vector (x, y, z) as defined by the below formulae.

$$x' = \frac{x}{z} \tag{1}$$

$$y' = \frac{y}{z} \tag{2}$$

$$r^2 = x'^2 + y'^2 \tag{3}$$

$$x'' = x' * \frac{1 + k_1 r^2 + k_2 r^4 + k_3 r^6}{1 + k_4 r^2 + k_5 r^4 + k_6 r^6} + 2p_1 x' y' + p_2 \left(r^2 + 2r'^2 \right) \tag{4}$$

$$y'' = y' * \frac{1 + k_1 r^2 + k_2 r^4 + k_3 r^6}{1 + k_4 r^2 + k_5 r^4 + k_6 r^6} + p_1 \left(r^2 + 2y'^2 \right) + 2p_2 x' y' \tag{5}$$

$$u = principalPo\,\mathrm{int}\,X * x'' + scaleX \tag{6}$$

$$v = principalPo\,\mathrm{int}\,Y * y'' + scaleY \tag{7}$$

The vector (x, y, z) shall define a position in a right-hand orthogonal 3-D camera coordinate system with the x-axis pointing to the right, the y-axis pointing down, and the z-axis being parallel to the optical axis, that is pointing to the view direction of the camera.

The vector (u, v) shall be the 2-D pixel position of the projected image of (x, y, z), where u is the horizontal pixel position and v is the vertical pixel position with $(u, v) = (0, 0)$ denoting the top left pixel of the image.

6.6.9 struct sExtrinsicCamParam

Purpose: The extrinsic camera parameters determine the mapping from 3-D vehicle coordinates to 3-D camera coordinates. The extrinsic parameters of the camera can be written and read out by application software and can be used, in conjunction with the intrinsic camera parameters, for back-projection form a 2-D image position to a line in 3-D space that is defined in terms of the vehicle coordinate system.

The sExtrinsicCamParam data structure containing the extrinsic camera parameters is defined in Table 18.

Table 18 — struct sExtrinsicCamParam

Name	Type	Description
rotationMat	float64[9]	3-by-3-element rotation matrix
t_{Vec}	float64[3]	3-D translation vector

Using the constants contained in the above data structure, a 3-D vector (x, y, z) can be computed from a given 3-D vector (X, Y, Z) as defined by the below formulae.

$$x = rotationMat[0] * X + rotationMat[1] * Y + rotationMat[2] * Z + t_{Vec}[0] \tag{8}$$

$$y = rotationMat[3] * X + rotationMat[4] * Y + rotationMat[5] * Z + t_{Vec}[1] \tag{9}$$

$$z = rotationMat[6] * X + rotationMat[7] * Y + rotationMat[8] * Z + t_{Vec}[2] \tag{10}$$

The vector (X, Y, Z) shall define a position in a right-hand orthogonal 3-D vehicle coordinate system.

NOTE The orientation and offset of this coordinate system relative to the vehicle body is not defined by this International Standard.

The vector (x, y, z) shall define a position in a right-hand orthogonal 3-D camera coordinate system with the x-axis pointing to the right, the y-axis pointing down, and the z-axis being parallel to the optical axis, that is pointing to the view direction of the camera.

6.6.10 structure sPersistentEntryList

Purpose: Definition of the extrinsic parameters of the camera.

Table 19 — structure sPersistentEntryList

Name	Type	Description
numberOfEntries	uint16	Number of persistent entries
listOfEntries	uint16[numberOfEntries]	List of persistent storage entry IDs

6.6.11 structure sPersistentStorageEntry

Purpose: This structure defines the content of a persistent storage entry (PSE) used in the Set/Get/Restore host parameter functions.

Table 20 — structure sPersistentStorageEntry

Name	Type	Description
Flag	uint32	0 = PSE is R/W 1 = PSE is RO
Length	uint16	Length of persistent data in byte
PersistentStorageID	uint16	Use enumeration ePersistentStorageID, see 6.5.8
PersistentData	uint8(Length)	Persistent data, the structure of the persistent data are identified by the PersistentStorageID
CRC16	uint16	CRC16 calculated using 16 LSB of the CCITT X.25 Polynome

Persistent storage entries are used to store various types of data in the camera's flash or EEPROM. The structure of data type to be stored is identified by the persistent storage ID. They can be stored as RW or RO entry. When writing to an existing RO PSE, the cameras shall return an error. When writing to an existing RW PSE, the existing PSE shall be replaced. The length of an entry shall not exceed maxLengthOfPersistenEntry reported in the camera data sheet.

For the following data types, predefined identifier exists:

— boot parameter;

— characteristic of the imager;

— defect pixel map of the imager;

— intrinsic calibration parameter;

— extrinsic calibration parameter;

— ROI parameter;

— imager register parameter.

6.6.12 structure sTimeStamp

Purpose: Definition of a Timestamp.

Table 21 — structure sTimeStamp

Name	Type	Description
802ASExtension	uint16	Additional 16 bits for compatibility to IEEE 802.AS
secondsField	uint32	Portion of the timestamp in units of seconds
nanosecondsField	uint32	Fractional portion of the timestamp in units of nanoseconds. The nanosecondsField member is always less than 10e9

EXAMPLE +2,0000 000 01 s is represented by secondsField = 0000 0000 000216 and nanosecondsField = 0000 000116.

6.6.13 structure sDatasheet

Purpose: Defines the characteristics of the camera.

Table 22 — structure sDatasheet

Name	Type	Description
deviceIdentification	uint8[256]	String mirrored to the camera API but managed by the generic device, UTF-8 encoded
standardVersion	unit32	Number of the highest supported International Standard
imagerCharacteristic	sImagerCharacteristic	Description of the cameras imager
intrinsicCamParam	sIntrinsicCamParam	Intrinsic parameter of the camera
extrinsicCamParam	sExtrinsicCamParam	Extrinsic parameter of the camera
defectPixelMap	sPixelMap	Supplier information of defect pixels
numOfTemperatures	unit32	Number of temperature sensor of the device
numOfRegionOfInterest	unit32	NumOfRegionOfInterest > 0 is mandatory, see also 6.6.14
maxLengthOfPersistentEntry	unit32	Maximum allowed length of a PSE

6.6.14 structure sRegionOfInterest

Purpose: This structure definition of a rectangular region of interest. It is used to specify the parameter for an Event ID derived by the roiIndex.

Table 23 — structure sRegionOfInterest

Name	Type	Description
roiSizeAndPosition	sRect	Rectangular region of interest, the roiResolution in roiIndex 0 defines the grab size of the camera; roiResolutions for a roiIndex > 0 shall fit within the roiResolution given by roiIndex = 0
videoFormat	sVideoFormat	For more details, see 6.6.15.
histogrammFormat	sHistogrammFormat	For more details, see 6.6.16.

The ROIs (regions of interest) are used to acquire data from the camera image using subscribe/notification mechanism. The data can be the image content of the region itself or a histogram of the region.

ROIs are interpreted as an array of structures and referenced by its index (roiIndex). The related event group ID is derived from the roiIndex and the event group type.

$$EventgroupID = eEventGroupType + (roiIndex + 1) \tag{11}$$

Table 24 — Event ID

Name	Value	Description
VideoStream1	0x9001	Event ID for video events related to ROI index 0 (main video stream)
VideoStream2	0x9002	Event ID for video events related to ROI index 1
:		
VideoStreamN	0x900n	Event group ID for events related ROI index n, given by sDatasheet.numOfRegionOfInterest-1
Histogram1	0xA001	Event ID for histogram events related to ROI index 0
Histogram2	0xA002	Event ID for histogram events related to ROI index 1

Table 24 *(continued)*

Name	Value	Description
:		
HistogramN	0xA00n	Event ID for histogram events related ROI index n, given by sDatasheet.numOf-RegionOfInterest-1

Each camera shall support at least one ROI with the index 0. The number of supported ROIs is given numOfRegionOfInterest in the sDatasheet structure.

The region 0 represents the main video stream of the camera and its use is described in the video format structure (see 6.6.15). The content of region 0 shall be always transported via AVTP packages. Therefore, transmissionMethod shall be set always to 0. They can either be transmitted as compressed or uncompressed stream.

All other regions are optional. They can be used to read out details of the image. This could be the extra lines of the imager with black level or register information.

Also, the measurement fields for exposure control threaded as region of interest. If the ROIs are used as measurement window for automatic functions like AGC or AEC, the histogram format shall be defined. Histogram values shall be calculated for all enabled ROIs.

All ROIs with index 1 to 15 can be transported as AVTP or notification packages. The structure of the notification packages is defined in 6.6.18 and 6.6.16

Since notification package are limited in size, the decision which transport method is used is depending on the amount of data to be transmitted.

6.6.15 structure sVideoFormat

Purpose: This structure defines the parameter of the video format associated to certain ROI.

Table 25 — structure sVideoFormat

Name	Type	Description
videoFormatEnabled	uint8	Disabled = 0 Enabled = 1
transmissionMethod	uint8	0 use AVTP packages 1 use notification packages
transmissionCycle	uint16	Only once = 0 Every *n* images > 0
WidthAndHeight	sImageDimension	Actual size of the transmitted video image. In case the frame size given by resolution does not match the size given by the associated roiResolution (resolution.x < > grabSize.P2.x – grabSize.P1.x or resolution.y < > grabSize.P2.y – grabSize. P1.y), the image is scaled by the imager.
frameRate	uint32	Frame rate (interlaced: not field rate!) * 2^{16}, TODO: Check if precision is sufficient
Interlaced	uint8	0: progressive scan video 1: interlaced video

Table 25 *(continued)*

Name	Type	Description
colourSpace	uint8	0: Grayscale8 1: YUV411 2: YUV422 3: YUV444 4: RGB8 5: Grayscale16 6: RGB16 7: signed Grayscale16 8: signed RGB16 9: RAW8 0: RAW16 11: Grayscale12 2: RAW12 vendor specific codes start at 128
maxBitrate	uint32	upper limit for bit rate after compression, not applicable to uncompressed transmission in MBit/s
videoCompression	uint8	0: UNCOMPRESSED 1: JPEG 2: H264 vendor codes starting at 128 can be used for specific codec profiles

6.6.16 struct sHistogramFormat

Purpose: This structure defines the histogram format used for a certain ROI.

Table 26 — struct sHistogramFormat

Name	Type	Description
histogramEnabled	uint8	Disabled = 0 Enabled = 1
histogramUpdateCycle	uint8	Defines how often a histogram will be calculated. only once = 0 every *n* images > 0
usedVideoComponent	uint8	Defines from which video component the histogram is derived. 0: Y, 1: R, 2: G, 3: B, manufacturer specific codes starting at 128
dataType	uint8	2: unit16 4: unit32
binSize	uint8	Defines the resolution of a histogram. With a bin size of 1, a bin for each value in the range of the selected component is used. If the bin size is 4, a bin for four consecutive values in the range of the component is used.
numberOfBins	uint16	Number of histogram bins

The resolution of a histogram and the size of the resulting histogram need to be considered when setting the histogram format. Theoretically, all pixels of a ROI can fall into one single bin. Therefore, the dataType shall be selected accordingly. This number of bins is calculated with Formula (12):

$$numberOfBins = \frac{(2 \exp binDepth)}{binSize} \tag{12}$$

EXAMPLE 1 binSize = 1, bitDepth = 10, ROI width = 128, and ROI height = 128

$$numberOfPixels = 128 \times 128 = 16\,384 \tag{13}$$

Therefore, the histogram data type has to be chosen as uint16:

dataType = 2

The number of histogram bins is then given by:

$$numberOfBins = \left(\frac{2 \exp 10}{1}\right) \times 2 = 1\,024 \times 2 = 2\ 048 \tag{14}$$

EXAMPLE 2 binSize = 2, bitDepth = 10, ROI with = 1 024, and ROI height = 480

$$numberOfPixels = (1\,024 \times 480) = 4\ 915\ 20 \tag{15}$$

Therefore, the histogram data type has to be chosen as uint32:

dataType = 4

The number of histogram values is then given by:

$$numberOfBins = \left(\frac{2 \exp 10}{2}\right) \times 4 = (1\,024 \div 2) \times 4 = 2\,048 \tag{16}$$

6.6.17 structure sHistogrammContent

Purpose: This structure defines the format of a histogram notification package.

Table 27 — structure sHistogrammContent

Name	Type	Description
roiIndex	uint8	ROI from which the histogram is derived
nBins	Uint16	The notification package contains nBins RoiValues
firstBin	Uint16	Index of the first roiValue in this package in the array of histogram values
timeStamp	sTimeStamp	Time stamp of the image in sequence from which the histogram is derived, for more details, see 6.6.12
numberOfROIValues	uint32	numberOfHistValues specified by the ROI defintion in bytes, see 6.6.16
histogramValues	uint8 [nBins]	Array of histogram values

If the size of a histogram exceeds the supported size of a notification package, the histogram shall be split in multiple notification packages.

For example, the supported package size is 1 400 and number of ROI values is 1 024. The notification, then, shall be transmitted as one package (nBins = numberOfROIValues).

1st package nBins = 1 024, firstBin = 0, numberOfROIValues = 1 024

For example, the supported package size is 1 400 and number of ROI Values is 2 048. The notification, then, shall be split in to two packages.

1st package nBins = 1 024, firstBin = 0, numberOfROIValues = 2 048

2nd package nBins = 1 024, firstBin = 1 024, numberOfROIValues = 2 048

6.6.18 structure sVideoContent

Purpose: This structure defines the format of a video content notification package.

Table 28 — structure sVideoContent

Name	Type	Description
roiIndex	uint8	ROI from which the video content is derived
nBins	Uint16	Notification package contains nBins ROIValues
firstBin	Uint16	Index of the first ROIValue in this package in the array of ROI values
timeStamp	sTimeStamp	Time stamp of the image, in sequence from which the histogram is derived; for more details, see 6.6.12.
NumberOfROIValues	uint32	Width × height × bitdepth given by sVideoFormat. colourSpace, see 6.6.15.
ROIContent	uint8 [nBins]	Uncompressed content of the ROI

If the sizes of the video content exceeds the supported size of a notification package, the video content shall be split in multiple notification packages.

For example, the supported package size is 1 400 and number of ROI values is 1 024. The notification, then, shall be transmitted as one package (nBins = numberOfROIValues).

1st package nBins = 1 024, firstBin = 0, numberOfROIValues = 1 024

For example, the supported package size is 1 400 and number of ROI Values is 2 048. The notification, then, shall be split in to two packages (nBins < numberOfROIValues).

1st package nBins = 1 024, firstBin = 0, numberOfROIValues = 2 048

2nd package nBins = 1 024, firstBin = 1 024, numberOfROIValues = 2 048

6.6.19 Structure sControlMode

Purpose: Definition of a camera control mode register.

Table 29 — Structure sControlMode

Name	Type	Description
SupportedModes	unit8	Use 6.5.6.
ControlMode	uint8	Use 6.5.7.

On read, the control mode field reports the current mode of operation for this control.

On write, the mode of operation will change according to the value written in this field. Only supported modes of operation are valid. If set to an unsupported mode, the cameras shall return an error code

6.6.20 structure sUnsignedCtl

Purpose: Definition of an unsigned camera control.

Table 30 — structure sUnsignedCtl

Name	Type	Description
ControlStatus	sControlMode	See 6.6.19.
ControlValue	uint16	For information on valid values, consider sCamControl.

6.6.21 structure sSignedCtl

Purpose: Definition of a signed camera control.

Table 31 — structure sSignedCtl

Name	Type	Description
ControlStatus	sControlMode	See 6.6.19.
ControlValue	sint16	For information on valid values, consider sCamControl.

6.6.22 structure sCombinedCtl

Purpose: Definition of a combined camera control.

Table 32 — structure sCombinedCtl

Name	Type	Description
ControlStatus	sControlMode	See 6.6.19.
ControlValueA	uint8	For information on valid values, consider sCamControl.
ControlValueB	uint8	For information on valid values, consider sCamControl.

6.6.23 structure sCamControl

Purpose: Defines the parameters to control the camera

Table 33 — structure sCamControl

Name	Type	Description
exposureTime	sUnsignedCtl	Controls the integration time Unit µs, range values within the frame period, resolution µs
Brightness	sSignedCtl	Controls the black level offset Init %, range –20 % - 20 %, resolution 0.1 %
Gain	sSignedCtl	Controls the gain circuits of the camera Unit db, range –20 - 20 dbB, resolution 0.2 db
Hue	sUnsignedCtl	Controls the hue circuits of the camera Unit deg, range 0 - 360 deg, resolution 1 deg
Saturation	sUnsignedCtl	Unit %, range 0 - 100 %, resolution 1 %
Gamma	sUnsignedCtl	Controls the gamma correction circuit of the camera Unit none, range 0 - 10, resolution 0.01
whiteBalance	sCombinedCtl	Controls the white colour, using R and B when in RGB mode, using U and V when in YUV mode Unit %, range 0 - 100 %, resolution 1 %

Table 33 *(continued)*

Name	Type	Description
Heater	sUnsignedCtl	Controls the heater of the camera Range on - off
Temperature	sUnsignedCtl	Controls the temperature of the camera See 6.6.25 for more information.

6.6.24 structure sCamStatus

Purpose: Defines the camera status information.

Table 34 — structure sCamStatus

Name	Type	Description
lockedBy	uint32	User of the camera (e.g. IP address) is set/unset by specific methods.
powerStatus	uint8	Value mirrored to the camera API, but managed by the generic device
Temperature	uint16	Actual temperature of the camera

6.6.25 Temperature

6.6.25.1 General

Application requests the internal temperature of the camera.

The camera has to respond always on request with the corresponding 16-bit temperature information. The camera puts the temperature information with the below defined resolutions (7 bits to 15 bits) on the 16-bit word. Bits not in use are mandatory to be set always to 0. This makes it possible that a 16-bit word can always be analysed at the application.

The temperature information is then transmitted in a 16-bit value over the communication channel (two's complement word).

In the application, the user can choose different resolutions for conversion, based on the application demand.

E.g. for less temperature accuracy, only 2°C/LSB resolution is necessary, therefore, only 8 bits (incl. signed indicator) will be evaluated. Or if a high accuracy is necessary, a resolution of 0.007 8°C/LSB can be used by evaluating the 16-bit value (incl. signed indicator) information.

Table 35 — Temperature resolution

Parameter	Resolution range
Min. resolution	0.007 8°C/LSB
Max. resolution	2°C/LSB

The chosen method allows realizing all relevant temperature ranges, as the following application example, from –55 °C to +125 °C, whereas, the theoretical temperature range, according to the resolution, could be –256 °C to +254 °C, to be open for future technologies.

Table 36 — Example for temperature ranges

Temperature range	Theoretical	Application example
Min. temperature	–256°C	–55°C
Max. temperature	+254°C	+125°C

6.6.25.2 Definition of 16-bit temperature value

Table 37 characterizes the definition of 16-bit temperature values.

Table 37 — Definition of 16-bit temperature value

Bits	Description	Values
15	MSB is signed indicator with two's complement format	Positive temperature value 0 Negative temperature value 1
14:8	Temperature value standard 7 bits standard resolution (°C/LSB)	7 bits = 2 °C/LSB
7:0	Temperature value extended 8 bits extended resolution (°C/LSB) Depends on the application, on how many bits will be used/analysed. Application can scale up to maximum resolution of 15 bits, if a high resolution has to be used.	8 bits = 1 °C/LSB 9 bits = 0,5 °C/LSB 10 bits = 0,25 °C/LSB 11 bits = 0,125 °C/LSB 12 bits = 0,0625 °C/LSB 13 bits = 0,03125 °C/LSB 14 bits = 0,015625 °C/LSB 15 bits = 0,0078125 °C/LSB

Not used resolution bits are mandatory to be set fixed to "0" at the camera (sender), to guarantee compatibility with the application (receiver).

6.6.25.3 Formula to calculate the temperature value

Formula (17) is used to calculate the temperature value:

$$Temperature[°C] = Resolution[°C / LSB] * TemperatureValue[LSB] \quad (17)$$

NOTE Selection of the resolution value is related to the resolution bits which are used for evaluation. See examples below.

6.6.25.4 Examples

In the following example, the possible resolutions are displayed for a positive temperature of +102 °C. The red marked bits are the evaluated resolution bits, which are mandatory to be linked to the corresponding resolution setting (e.g. 7-bit resolution is linked to 2 °C/LSB).

Table 38 — Temperature value calculation (positive values)

Setting		Temperature value calculation (positive values)		
Used bits resolution	Resolution (°C/LSB)	16-bit binary value * (incl. signed indicator)	Dec value (LSB)	Temperature (°C)
7 bits of 15 bits	2	0011 0011 0000 0000	51	+102
8 bits of 15 bits	1	0011 0011 0000 0000	102	+102
9 bits of 15 bits	0,5	0011 0011 0000 0000	204	+102

Table 38 *(continued)*

Setting		Temperature value calculation (positive values)		
10 bits of 15 bits	0,25	0011 0011 0000 0000	408	+102
11 bits of 15 bits	0,125	0011 0011 0000 0000	816	+102
12 bits of 15 bits	0,062 5	0011 0011 0000 0000	1 632	+102
13 bits of 15 bits	0,031 25	0011 0011 0000 0000	3 264	+102
14 bits of 15 bits	0,015 625	0011 0011 0000 0000	6 528	+102
15 bits of 15 bits	0,007 812 5	0011 0011 0000 0000	13 056	+102

For the given example, the temperature value is then calculated with Formula (18):

$$T = 2\ °C/LSB \times 51\ LSB = +\ 102\ °C \tag{18}$$

In this example, the possible resolutions are displayed for a negative temperature of –26 °C. The red marked bits are again the evaluated resolution bits, which are mandatory to be linked to the corresponding resolution setting (e.g. 7-bit resolution is linked to 2° C/LSB):

Table 39 — Temperature value calculation (negative values)

Setting		Temperature value calculation (negative values)		
Used bits resolution	Resolution (°C/LSB)	16-bit binary value * (incl. signed indicator)	Dec value (LSB)	Temperature (°C)
7 bits of 15 bits	2	1111 0011 0000 0000	–13	–26
8 bits of 15 bits	1	1111 0011 0000 0000	–26	–26
9 bits of 15 bits	0,5	1111 0011 0000 0000	–52	–26
10 bits of 15 bits	0,25	1111 0011 0000 0000	–104	–26
11 bits of 15 bits	0,125	1111 0011 0000 0000	–208	–26
12 bits of 15 bits	0,062 5	1111 0011 0000 0000	–416	–26
13 bits of 15 bits	0,031 25	1111 0011 0000 0000	–832	–26
14 bits of 15 bits	0,015 625	1111 0011 0000 0000	–1664	–26
15 bits of 15 bits	0,007 812 5	1111 0011 0000 0000	–3328	–26

The temperature value is then calculated with Formula (19):

$$T = 2\ °C/LSB \times (–13)\ LSB = –26\ °C \tag{19}$$

Table 40 shows the corresponding calculated temperature output value for a fixed resolution of 9 bits, which is linked to 0.5 °C/LSB, for some specific 16-bit temperature values.

Table 40 — Temperature value with fixed resolution

Setting		Temperature value with fixed resolution 0,5°C/LSB		
Used bits resolution	Resolution (°C/LSB)	16-bit binary value * (incl. signed indicator)	Dec value (LSB)	Temperature (°C)
9 bits of 15 bits	0,5	0011 1110 1000 0000	250	+125
9 bits of 15 bits	0,5	0010 1010 1000 0000	170	+85
9 bits of 15 bits	0,5	0000 0001 0000 0000	4	+2
9 bits of 15 bits	0,5	0000 0000 1000 0000	2	+1

Table 40 *(continued)*

Setting		Temperature value with fixed resolution 0,5°C/LSB		
9 bits of 15 bits	0,5	0000 0000 0000 0000	0	0
9 bits of 15 bits	0,5	1111 1111 1000 0000	–2	–1
9 bits of 15 bits	0,5	1111 1111 0000 0000	–4	–2
9 bits of 15 bits	0,5	1110 1100 0000 0000	–80	–40
9 bits of 15 bits	0,5	1110 0100 1000 0000	–110	–55

The temperature value is then calculated with Formula (20):

$$T = 0.5\ °C/LSB \times (250)\ LSB = +\ 125\ °C \quad (20)$$

Left hand side leading bit is the signed indicator bit. Red marked bits are used for the resolution (from 7 bits to max. 15 bits).

6.7 API reference

6.7.1 getDataSheet (MethodID 0x0001)

Purpose: This method returns the datasheet of the camera.

Input parameters: None

Output parameters:

Table 41 — Output parameters getDataSheet (MethodID 0x0001)

Name	Type	Description
Datasheet	sDatasheet	Datasheet of the camera

Specific error codes: No specific error codes

6.7.2 getCamStatus (MethodID 0x0002)

Purpose: This method returns the current status of the camera.

Input parameters: None

Output parameters:

Table 42 — Output parameters getCamStatus (MethodID 0x0002)

Name	Type	Description
camStatus	sCamStatus	Current status of the camera

Specific error codes: No specific error codes.

6.7.3 setCamMode (MethodID 0x0003)

Purpose: This method starts, stops, and restarts the camera application in the camera

A client shall look the camera using the setCamEclusive method before requesting this method.

Input parameters:

Table 43 — Input parameters setCamMode (MethodID 0x0003)

Name	Type	Description
cameraMode	uint32	Use enumeration eCameraMode

Specific error codes:

Table 44 — Error codes setCamMode (MethodID 0x0003)

Error code
E_LOCKED_BY_FOREIGN_INSTANCE
E_LOCK_ EXPIRED

6.7.4 setCamExclusive (MethodID 0x0011)

Purpose: This method assigns the control of the camera exclusively to the requesting client.

In an application with more than one ECU, one instance should be able to control the camera exclusively. This instance will issue the method "setCamLocked".

If the attribute lockedBy of the data structure sCamStatus is 0, the instance ID of the requesting client (e.g. IP address) shall be stored in the attribute lockedBy.

All set/erase methods can only be used by the instance with the ID stored in lockedBy. Request from other instances shall be answered with the E_LOCKED_BY_FOREIGN_INSTANCE error.

When lockedBy is not 0, the camera shall report the E_LOCKED_BY_FOREIGN_INSTANCE error.

If time to live was set to 0xFFFFFFFF, the camera shall use its default timeout parameter instead

After time to live or default time out is reached, the look expires. If this is the case, lockedBy shall be set to 0x00000000.

Input parameters:

Table 45 — Input parameters setCamExclusive (MethodID 0x0011)

Name	Type	Description
TTL	uint32	Time to live, after this time (in seconds) the look expires

Output parameters: None

Specific error codes:

Table 46 — Error code setCamExclusive (MethodID 0x0011)

Error code
E_LOCKED_BY_FOREIGN_INSTANCE

6.7.5 eraseCamExclusive (MethodID 0x0019)

Purpose: This method removes the exclusive control look for the requesting client.

The Instance ID of the requesting client shall fit the Instance ID stored in parameter lockedBy of the data structure sCamStatus. If this is the case, lockedBy shall be set to 0x00000000.

When the Instance ID doesn't match the lockedBy value, the camera shall report the E_LOCKED_BY_FOREIGN_INSTANCE error.

Input parameters: None

Output parameters: None

Specific error codes:

Table 47 — Error code eraseCamExclusive (MethodID 0x0019)

Error code
E_LOCKED_BY_FOREIGN_INSTANCE
E_NOT_LOCKED

6.7.6 setHostParameters (MethodID 0x0022)

Purpose: Sets the host parameters using persistent storage entries.

A client shall look the camera using the setCamEclusive method before requesting this method.

Input parameters:

Table 48 — Input parameters setHostParameters (MethodID 0x0022)

Name	Type	Description
numberOfEntries	uint32	Number of PSE
hostParams	sPersistentStorageEntry[numberOfEntries]	Array of PSEs

Specific error codes:

Table 49 — Error codes setHostParameters (MethodID 0x0022)

Error code
E_INVALID_PS_ENTRY
E_INVALID_PS_OPERATION
E_INVALID_PS_DATA
E_NO_MORE_SPACE
E_LOCKED_BY_FOREIGN_INSTANCE
E_LOCK_ EXPIRED

6.7.7 getHostParameters (MethodID 0x0024)

Purpose: Returns the requested host parameter by reading persistent storage entries.

Input parameters:

Table 50 — Input parameters getHostParameters (MethodID 0x0024)

Name	Type	Description
persistentEntryList	sPersistentEntryList	List of requested PSE IDs

Output parameters:

Table 51 — Output parameters getHostParameters (MethodID 0x0024)

Name	Type	Description
numberOfEntries	uint32	Number of PSE
persistentEntries	sPersistentStorageEntry[numberOfEntries]	Array of PSEs as in Table 48

Specific error codes:

Table 52 — Error codes getHostParameters (MethodID 0x0024)

Error code
E_INVALID_PS_ENTRY

6.7.8 eraseHostParameters (MethodID 0x0029)

Purpose: This method forces the camera to erase the requested host parameters addressed by the PSE ID.

A client shall look the camera using the setCamEclusive method before requesting this method.

Input parameters:

Table 53 — Input parameters eraseHostParameters (MethodID 0x0029)

Name	Type	description
persistentEntryList	sPersistentEntryList	List of requested PSE IDs

Output parameters: none

Specific error codes:

Table 54 — Error codes eraseHostParameters (MethodID 0x0029)

Error code
E_INVALID_PS_ENTRY
E_INVALID_PS_OPERATION
E_LOCKED_BY_FOREIGN_INSTANCE
E_LOCK_ EXPIRED

6.7.9 setRegionOfInterest (MethodID 0x0101)

Purpose: This method sets the parameter for a region of interest addressed by index.

A client shall look the camera using the setCamEclusive method before requesting this method.

Input parameters:

Table 55 — Input parameters setRegionOfInterest (MethodID 0x0101)

Name	Type	Description
roiIndex	uint32	Index of the requested ROI, shall be within the supported range given by sDatasheet.numOfRegionOfInterest
regionOfInterest	sRegionOfInterest	Parameter for the ROI given by roiIndex

Specific error codes:

Table 56 — Error parameters setRegionOfInterest (MethodID 0x0101)

Error code
E_INVALID_ROI_INDEX
E_INVALID_HISTOGRAM_FORMAT
E_INVALID_VIDEO_FORMAT
E_LOCKED_BY_FOREIGN_INSTANCE
E_LOCK_ EXPIRED

6.7.10 setRegionsOfInterest (MethodID 0x0102)

Purpose: This method sets the parameter for all supported regions of interest.

A client shall look the camera using the setCamEclusive method before requesting this method.

Input parameters:

Table 57 — Input parameters setRegionsOfInterest (MethodID 0x0102)

Name	Type	Description
numOfRegionOfInterest	uint32	Number of ROIs, shall not exceed numOfRegionOfInterest
roiParameter	sRegionOfInterest [numOfRegionOfInterest]	Array of ROI parameter

Output parameters: None

Specific error codes:

Table 58 — Error codes setRegionsOfInterest (MethodID 0x0102)

Error code
E_INVALID_ROI_INDEX
E_INVALID_ROI_NUMBER
E_LOCKED_BY_FOREIGN_INSTANCE
E_LOCK_ EXPIRED

6.7.11 getRegionOfInterest (MethodID 0x0103)

Purpose: This method returns the parameter for region of interest addressed by ROI index.

Input parameters:

Table 59 — Input parameters getRegionOfInterest (MethodID 0x0103)

Name	Type	Description
roiIndex	uint32	Index of the requested ROI, shall be within the supported range given by sDatasheet.numOfRegionOfInterest

Output parameters:

Table 60 — Output parameters getRegionOfInterest (MethodID 0x0103)

Name	Type	Description
regionOfInterest	sRegionOfInterest	Parameter of the requested ROI

Specific error codes:

Table 61 — Error codes getRegionOfInterest (MethodID 0x0103)

Error code
E_INVALID_ROI_INDEX

6.7.12 getRegionsOfInterest (MethodID 0x0104)

Purpose: This method returns the parameter for all supported region of interest.

Input parameters: None

Output parameters:

Table 62 — Output parameters getRegionsOfInterest (MethodID 0x0104)

Name	Type	Description
numOfRegionOfInterest	uint32	Number of returned RegionOfInterest
regionOfInterest	uint32[numOfRegionOfInterest]	Array of ROI indizes

Specific error codes:

Table 63 — Error codes getRegionsOfInterest (MethodID 0x0104)

Error code
E_INVALID_ROI_INDEX
E_INVALID_ROI_NUMBER

6.7.13 eraseRegionOfInterest (MethodID 0x0109)

Purpose: This method erases all parameter for the requested ROI.

Input parameters:

Table 64 — Input parameters eraseRegionOfInterest (MethodID 0x0109)

Name	Type	Description
roiIndex	uint32	Index of the requested ROI, shall be within the supported range given by sDatasheet.numOfRegionOfInterest

Output parameters: None

Specific error codes:

Table 65 — Error codes eraseRegionOfInterest (MethodID 0x0109)

Error code
E_INVALID_ROI_INDEX

6.7.14 setVideoFormat (MethodID 0x0111)

Purpose: This method sets the video format for a ROI addressed by ROI index.

A client shall look the camera using the setCamEclusive method before requesting this method.

Input parameters:

Table 66 — Input parameters setVideoFormat (MethodID 0x0111)

Name	Type	Description
roiIndex	uint32	Index of the requested ROI, shall be within the supported range given by sDatasheet.numOfRegionOfInterest
Format	sVideoFormat	Video format parameter for the addressed ROI

Specific error codes:

Table 67 — Error codes setVideoFormat (MethodID 0x0111)

Error code
E_INVALID_ROI_INDEX
E_INVALID_VIDEO_FORMAT
E_LOCKED_BY_FOREIGN_INSTANCE
E_LOCK_ EXPIRED

6.7.15 getVideoFormat (MethodID 0x0113)

Purpose: This method reads the parameter of the current video format, addressed by the ROI index.

Input parameters:

Table 68 — Input parameters getVideoFormat (MethodID 0x0113)

Name	Type	Description
roiIndex	uint32	Index of the requested ROI, shall be within the supported range given by sDatasheet.numOfRegionOfInterest

Output parameters:

Table 69 — Output parameters getVideoFormat (MethodID 0x0113)

Name	Type	Description
Format	sVideoFormat	video format addressed by the ROI index

Specific error codes:

Table 70 — Error codes getVideoFormat (MethodID 0x0113)

Error code
E_INVALID_ROI_INDEX

6.7.16 eraseVideoFormat (MethodID 0x0119)

Purpose: This method erases all video format parameter for the requested ROI.

A client shall look the camera using the setCamEclusive method before requesting this method.

Input parameters:

Table 71 — Input parameters eraseVideoFormat (MethodID 0x0119)

Name	Type	Description
roiIndex	uint32	Index of the requested ROI, shall be within the supported range given by sDatasheet.numOfRegionOfInterest

Output parameters: None

Specific error codes:

Table 72 — Error codes eraseVideoFormat (MethodID 0x0119)

Error code
E_INVALID_ROI_INDEX

Table 72 *(continued)*

Error code
E_LOCKED_BY_FOREIGN_INSTANCE
E_LOCK_ EXPIRED

6.7.17 setHistogramFormat (MethodID 0x0121)

Purpose: This method sets the histogram format parameter for a ROI addressed by ROI index.

A client shall look the camera using the setCamEclusive method before requesting this method.

Input parameters:

Table 73 — Input parameters setHistogramFormat (MethodID 0x0121)

Name	Type	Description
roiIndex	uint32	Index of the requested ROI, shall be within the supported range given by sDatasheet.numOfRegionOfInterest
Format	sHistogrammFormat	Histogram format parameter for the addressed ROI

Specific error codes:

Table 74 — Error codes setHistogramFormat (MethodID 0x0121)

Error code
E_INVALID_ROI_INDEX
E_INVALID_HISTOGRAM_FORMAT
E_LOCKED_BY_FOREIGN_INSTANCE
E_LOCK_EXPIRED

6.7.18 getHistogrammFormat (MethodID 0x0123)

Purpose: This method reads the parameter of the current histogram format, addressed by the ROI index.

Input parameters:

Table 75 — Input parameters getHistogrammFormat (MethodID 0x0123)

Name	Type	Description
roiIndex	uint32	Index of the requested ROI, shall be within the supported range given by sDatasheet.numOfRegionOfInterest

Output parameters:

Table 76 — Output parameters getHistogrammFormat (MethodID 0x0123)

Name	Type	Description
Format	sHistogrammFormat	video format addressed by the ROI index

Specific error codes:

Table 77 — Error codes getHistogrammFormat (MethodID 0x0123)

Error code
E_INVALID_ROI_INDEX

6.7.19 eraseHistogramFormat (MethodID 0x0129)

Purpose: This method erases all histogram format parameter for the requested ROI.

A client shall look the camera using the setCamEclusive method before requesting this method.

Input parameters:

Table 78 — Input parameters eraseHistogramFormat (MethodID 0x0129)

Name	Type	Description
roiIndex	uint32	Index of the requested ROI, shall be within the supported range given by sDatasheet.numOfRegionOfInterest

Output parameters: None

Specific error codes:

Table 79 — Error codes eraseHistogramFormat (MethodID 0x0129)

Error code
E_INVALID_ROI_INDEX
E_LOCKED_BY_FOREIGN_INSTANCE
E_LOCK_EXPIRED

6.7.20 SubscribeROIVideo (MethodID 0x0131)

Purpose: This method starts the transmission of a video stream for the requested ROI.

Input parameters:

Table 80 — Input parameters SubscribeROIVideo (MethodID 0x0131)

Name	Type	Description
roiIndex	uint32	Index of the requested ROI

The roiIndex shall be within the supported range given by sDatasheet.numOfRegionOfInterest. A valid video format needs to be set and enabled before calling this method

Output parameters:

When the transmissionMethod of the requested ROI video format indicates the use of notification packages, the requesting instance will receive a cyclic notification. The cycle is defined by the parameter transmissionCycle.

Table 81 — Output parameters SubscribeROIVideo (MethodID 0x0131)

Name	Type	Description
ROIContent	sVideoContent	Content of the registered ROI See 6.6.18.

When the transmissionMethod of the requested ROI video format indicates the use of AVTP packages, the requesting instance will receive cyclic AVTP packages. The cycle is defined by the parameter transmissionCycle.

Specific error codes:

Table 82 — Error codes SubscribeROIVideo (MethodID 0x0131)

Error code
E_INVALID_ROI_INDEX
E_INVALID_VIDEO_FORMAT

6.7.21 UnSubscribeROIVideo (MethodID 0x0132)

Purpose: This function stops the video streaming for the requested ROI.

Input parameters:

Table 83 — Input parameters UnSubscribeROIVideo (MethodID 0x0132)

Name	Type	Description
roiIndex	uint32	Index of the requested ROI, shall be within the supported range given by sDatasheet.numOfRegionOfInterest

6.7.22 SubscribeROIHistogram (MethodID 0x0133)

Purpose: This method starts the transmission of the histograms for the requested ROI.

Input parameters:

Table 84 — Input parameters SubscribeROIHistogram (MethodID 0x0133)

Name	Type	Description
roiIndex	uint32	Index of the requested ROI

The roiIndex shall be within the supported range given by sDatasheet.numOfRegionOfInterest. A valid histogram format needs to be set and enabled before calling this method.

Output parameters:

The requesting instance will receive cyclic a package. The cycle is defined by the parameter histogramUpdateCyle.

Table 85 — Output parameters SubscribeROIHistogram (MethodID 0x0133)

Name	Type	Description
ROIhistogram	sHistogrammContent	Histogram of the registered ROI See 6.6.16.

6.7.23 UnSubscribeROIHistogram (MethodID 0x0134)

Purpose: This method stops the transmission of the histograms for the requested ROI.

Input parameters:

Table 86 — Input parameters UnSubscribeROIHistogram (MethodID 0x0134)

Name	Type	Description
roiIndex	uint32	Index of the requested ROI, shall be within the supported range given by sDatasheet.numOfRegionOfInterest

6.7.24 setCamControl (MethodID 0x0201)

Purpose: This method sets the parameter for a camera control addressed by the control index.

A client shall look the camera using the setCamEclusive method before requesting this method.

Input parameters:

Table 87 — Input parameters setCamControl (MethodID 0x0201)

Name	Type	Description
camControlIndex	unit8	Use eControlIndex, see 6.5.5.
camControl	void	Use the type belonging to the camControlIndex, see 6.6.23.

Specific error codes:

Table 88 — Error codes setCamControl (MethodID 0x0201)

Error code
E_INVALID_CONTROL_INDEX
E_INVALID_CONTROL_MODE
E_INVALID_CONTROL_VALUE
E_LOCKED_BY_FOREIGN_INSTANCE
E_LOCK_ EXPIRED

6.7.25 setCamControls (MethodID 0x0202)

Purpose: This method sets the parameters for all camera controls.

A client must look the camera using the setCamEclusive method before requesting this method.

Input parameters:

Table 89 — Input parameters setCamControls (MethodID 0x0202)

Name	Type	Description
camControls	sCamControl	

Specific error codes:

Table 90 — Error codes setCamControls (MethodID 0x0202)

Error code
E_INVALID_CONTROL_MODE
E_INVALID_CONTROL_VALUE
E_LOCKED_BY_FOREIGN_INSTANCE
E_LOCK_ EXPIRED

For controls not intended to be changed, the ControlMode Field shall be set to appropriate.

6.7.26 getCamControl (MethodID 0x0203)

Purpose: This method returns the current parameter for a camera control addressed by the control index.

Input parameters:

Table 91 — Input parameters getCamControl (MethodID 0x0203)

Name	Type	Description
camControlIndex	unit8	Use eControlIndex, see 6.5.5.

Output parameters:

Table 92 — Output parameters getCamControl (MethodID 0x0203)

Name	Type	Description
camControl	sCamControl	Parameter of the requested camera control

Specific error codes:

Table 93 — Error codes getCamControl (MethodID 0x0203)

Error code
E_INVALID_CONTROL_INDEX

6.7.27 getCamControls (MethodID 0x0204)

Purpose: This method returns all current camera control parameters.

Input parameters: None

Output parameters:

Table 94 — Output parameters getCamControls (MethodID 0x0204)

Name	Type	Description
camControl	sCamControl	Data structure of all control parameters

Specific error codes:

Table 95 — Error codes getCamControls (MethodID 0x0204)

Error code
E_INVALID_CONTROL_INDEX

6.7.28 setCamRegister (MethodID 0x0301)

Purpose: This method writes the content of a register of the camera's imager addressed by physical register address.

A client shall look the camera using the setCamEclusive method before requesting this method.

The camera shall check if the imager register pair contains an unsupported register address/value pair and respond accordingly.

Input parameters:

Table 96 — Input parameters setCamRegister (MethodID 0x0301)

Name	Type	Description
imagerRegister	sImagerRegister	Pair of register address and value of the imager

Specific error codes:

Table 97 — Error codes setCamRegister (MethodID 0x0301)

Error code
E_INVALID_REGISTER_ADDRESS
E_INVALID_REGISTER_VALUE
E_INVALID_REGISTER_OPERATION
E_LOCKED_BY_FOREIGN_INSTANCE
E_LOCK_ EXPIRED

6.7.29 setCamRegisters (MethodID 0x0302)

Purpose: This method writes (atomic access) the content of a register block of the camera imager.

A client shall look the camera using the setCamEclusive method before requesting this method.

The camera shall check if the imager register block contains unsupported register address/value pairs and respond accordingly.

Input parameters:

Table 98 — Input parameters setCamRegisters (MethodID 0x0302)

Name	Type	Description
imagerRegisterBlock	sImagerRegisterBlock	Block of address/value pairs of imager register

Specific error codes:

Table 99 — Error codes setCamRegisters (MethodID 0x0302)

Error code
E_INVALID_REGISTER_ADDRESS
E_INVALID_REGISTER_VALUE
E_INVALID_REGISTER_OPERATION
E_LOCKED_BY_FOREIGN_INSTANCE
E_LOCK_ EXPIRED

6.7.30 getCamRegister (MethodID 0x0303)

Purpose: This method reads the content of a register of the camera imager address by the physical register address.

Input parameters:

Table 100 — Input parameters getCamRegister (MethodID 0x0303)

Name	Type	Description
regAddress	uint16	Address of the register

Output parameters:

Table 101 — Output parameters getCamRegister (MethodID 0x0303)

Name	Type	description
regValue	unit16	Value of the requested register

Specific error codes:

Table 102 — Error codes getCamRegister (MethodID 0x0303)

error code
E_INVALID_REGISTER_ADDRESS
E_INVALID_REGISTER_OPERATION

6.7.31 getCamRegisters (MethodID 0x0304)

Purpose: This method reads the content of a register block of the camera imager.

Input parameters:

Table 103 — Input parameters getCamRegisters (MethodID 0x0304)

Name	Type	description
regCount	uint16	Number of registers to be read
addressList	uint16 [regCount]	List of regCount imager register addresses

Output parameters:

Table 104 — Output parameters getCamRegisters (MethodID 0x0304)

Name	Type	description
imagerRegisterBlock	sImagerRegisterBlock	Block of address/value pairs of imager register

Specific error codes:

Table 105 — Error codes getCamRegisters (MethodID 0x0304)

error code
E_INVALID_REGISTER_ADDRESS
E_INVALID_REGISTER_OPERATION

6.7.32 setUsedRegisterSet (MethodID 0x0305)

Purpose: This method forces the camera to write the imager register set stored in the requested PSE to the imager.

A client shall look the camera using the setCamEclusive method before requesting this method.

When the PSE Identifier doesn't exist or is not an ImagerRegisterBlock entry, the camera shall respond with E_INVALID_PS_ENTRY.

The camera shall check if the imager register block contains unsupported register address/value pairs and respond accordingly.

Input parameters:

Table 106 — Input parameters setUsedRegisterSet (MethodID 0x0305)

Name	Type	Description
regSetID	uint16	PSE identifier, only ImagerRegisterBlock IDs are valid

Specific error codes:

Table 107 — Error codes setUsedRegisterSet (MethodID 0x0305)

error code
E_INVALID_PS_ENTRY
E_INVALID_REGISTER_ADDRESS
E_INVALID_REGISTER_VALUE
E_INVALID_REGISTER_OPERATION
E_LOCKED_BY_FOREIGN_INSTANCE
E_LOCK_ EXPIRED

6.8 Programming model for SOME/IP

This subclause describes the programming model for the implementation of the camera API using SOME/IP as specified in ISO 17215-2.

— An ISO 17215 camera shall use SOME/IP-SD protocol for service discovery.

— An ISO 17215 camera shall use SOME/IP protocol for remote procedure calls.

— All SOME/IP messages shall be transported via the UDP protocol.

— An ISO 17215 camera shall use the Audio Video Transport Protocol (AVTP) for video streaming.

SOME/IP-SD is used to determine which camera instances are available in the local network. A server broadcasts SOME/IP-SD OfferService messages with the ServiceID, the InstanceID, the IP address, and the chosen port number. Only UDP is used as transport protocol. A client sends SOME/IP-SD FindService messages with the ServiceID to collect information about all running instances of the service. It can also use RequestService and StopRequestService messages to start/stop certain instances.

SOME/IP request messages are used to call the Set/Get methods defined in the camera API. Each method is annotated with a MethodID, which allows the SOME/IP runtime to parse the payload of the message. The camera then replies with a SOME/IP response or error message.

For data that is to be sent periodically, SOME/IP offers notification messages. To publish the availability of notifications for an event, a server broadcasts SOME/IP-SD PublishEvent messages with the EventGroupID defined in the API. If an EventGroup is no longer available, a StopPublishEvent message is sent. To subscribe, a client sends a SOME/IP-SD Subscribe message with the corresponding EventGroupID to the server. From then on, it receives SOME/IP notification messages for all events in the group. These carry an EventID to allow the SOME/IP runtime to parse the payload. This mechanism is used to implement the Start/Stop methods of the API.

More detailed information can be found in ISO 17215-2. The subclauses below highlight the issues that are important for mapping camera API calls to SOME/IP messages.

6.8.1 General

The header fields in the SOME/IP RPC message header shall be assigned as follows:

— Service ID shall be set to 0x433F = 17215 (decimal);

— Protocol version shall be set to 0x01;

— Interface version shall be set to 0x01;

— Length, ClientID, SessionID, message type, and return code shall be set according to the rules in ISO 17215-2;

— Method ID and payload shall be set as specified in 6.5.1.

The header fields in the SOME/IP-SD message header shall be assigned as follows:

— Service ID shall be set to 0x433F = 17215 (decimal);

— InstanceID shall be set to the last octet of the camera's IP address;

— MajorVersion shall be 0x01;

— MinorVersion shall be 0x00000000;

— TTL and all other fields shall be set according to the rules of ISO 17215-2.

6.8.2 Startup behaviour

At start up a camera shall

— perform the IP address assignment procedure defined in ISO 17215-4,

— check if a PSE 'UseAtBootTime' is present and initialize the imager and ROIs accordingly. If no 'UseAtBootTime' parameter exists, factory defaults shall be used, and

— offer the ISO 17215 camera service in the network. Details are defined in ISO 17215-2.

6.8.3 Service discovery

For service discovery, the following requirements apply:

— a camera shall offer a service with the ID 0x433F = 17215 (decimal);

— a camera shall repeat the service offer with a CYCLIC_OFFER_DELAY of 100ms (as defined in ISO 17215-2);

— a camera shall answer to a FindService with the Service ID 0x433F = 17215 (decimal) with an OfferService message;

— setCameraMode(StartCameraService) shall be implemented with a SOME/IP-SD RequestService message;

— setCameraMode(StopCameraService) shall be implemented with a SOME/IP-SD StopRequestService message.

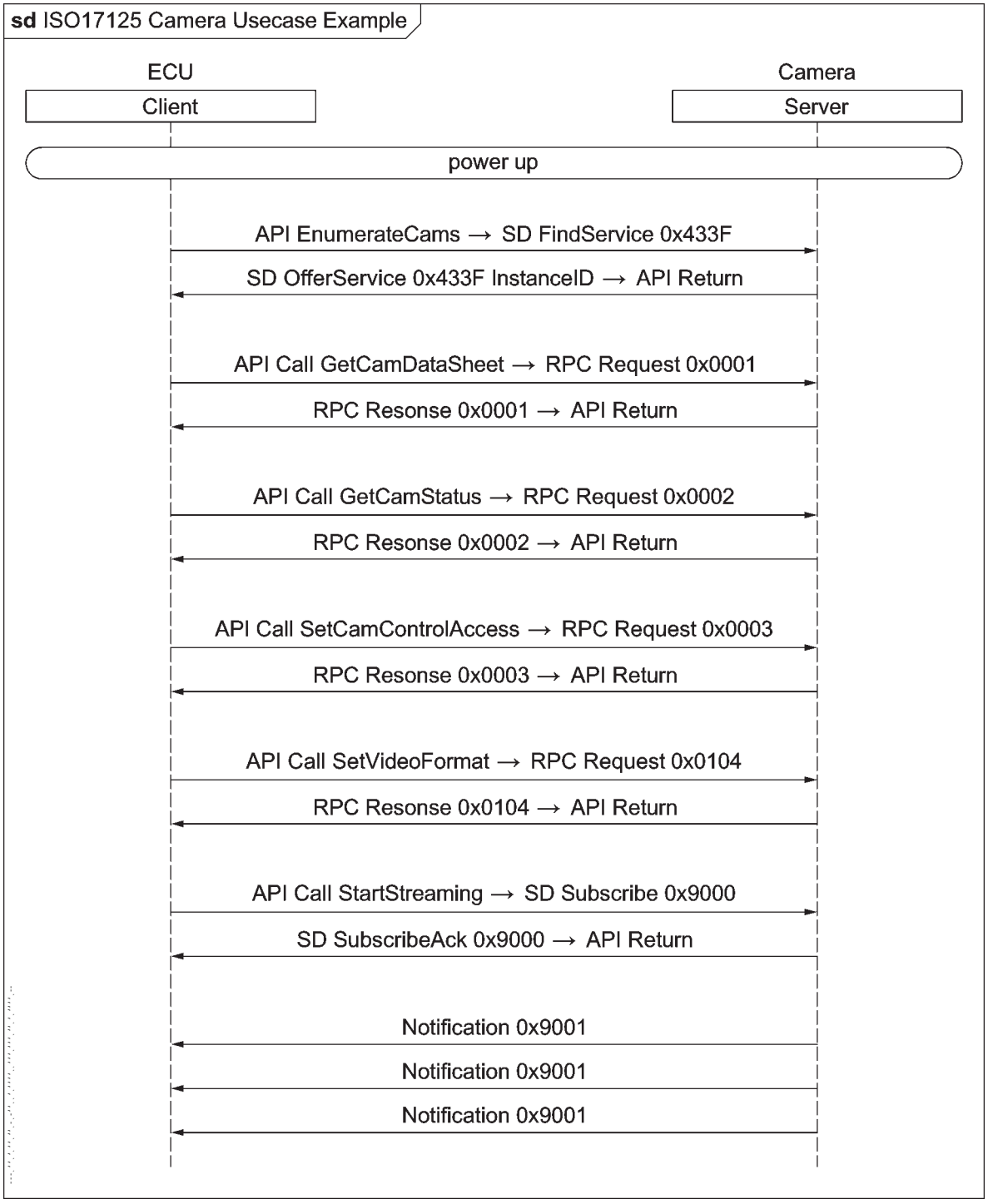


Figure 6 — Sequence diagram of a basic camera use case

Note that the diagram is simplified. Offer repetition, IP address assignment, timeouts, notification callbacks, and almost all parameters have been omitted for clarity. API calls also need not be blocking.

In Figure 6, a simplified sequence chart illustrates the principles of the API to SOME/IP mapping. For the sake of example, assume the service instance in question runs at IP address 192.168.23.42, UDP port 1234, and InstanceID 0x0007. Additionally, each SOME/IP message contains a ClientID field for AUTOSAR compatibility. Assume the camera service uses ClientID 0x0002 and the client has a ClientID of 0x0005. The PDU for an SD FindService message and an SD OfferService message under these conditions are shown in Figure 7 and Figure 8. The PDUs for the SetCamControlAccess call and the appropriate response or error is shown in Figure 9, Figure 10, and Figure 11.

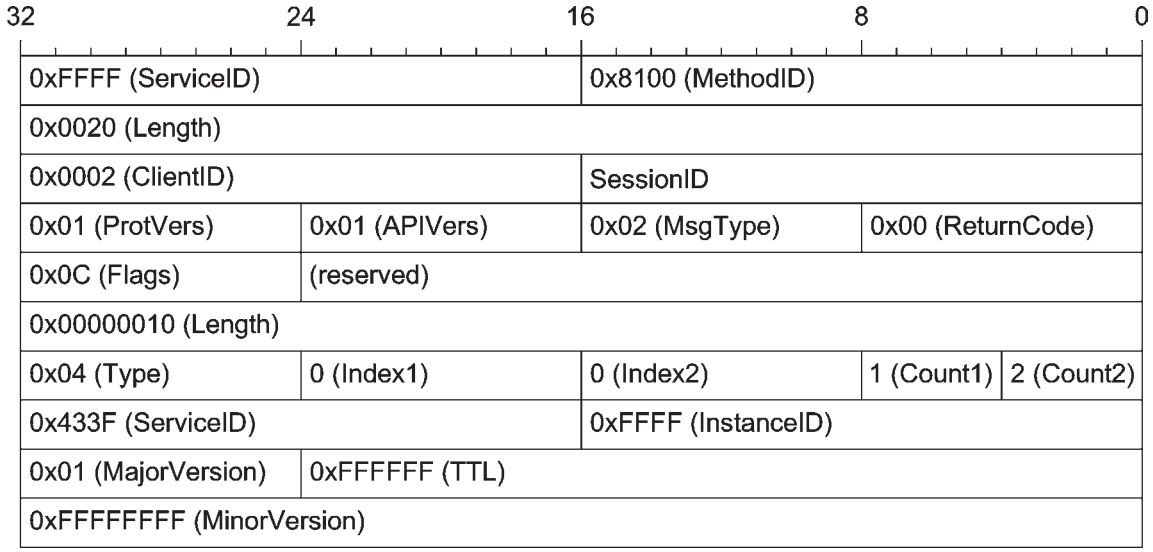


Figure 7 — Find service example (any instance, any minor version)

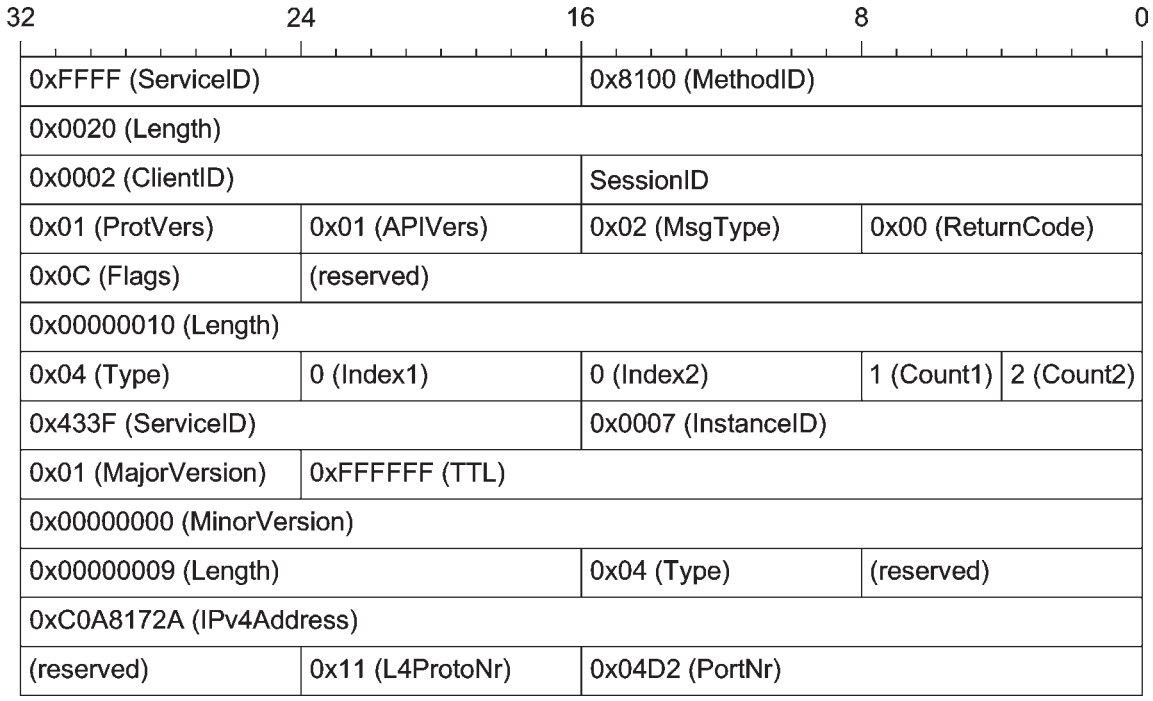


Figure 8 — OfferService example

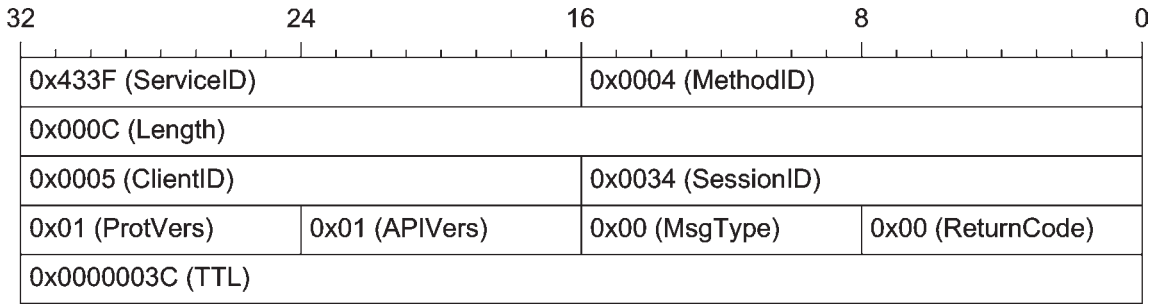


Figure 9 — SetCamControlAccess request

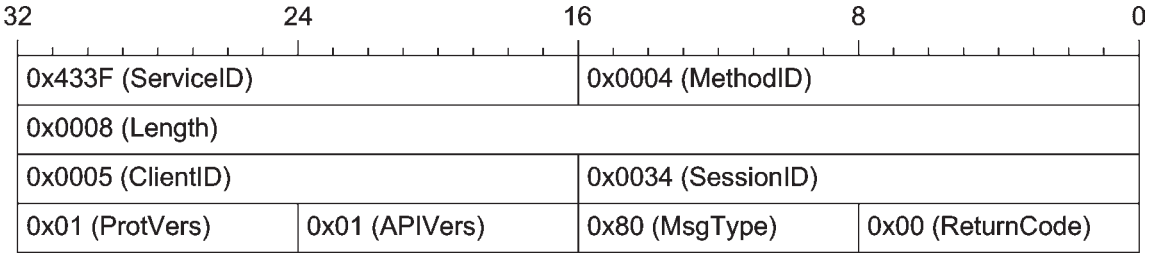


Figure 10 — SetCamControlAccess response

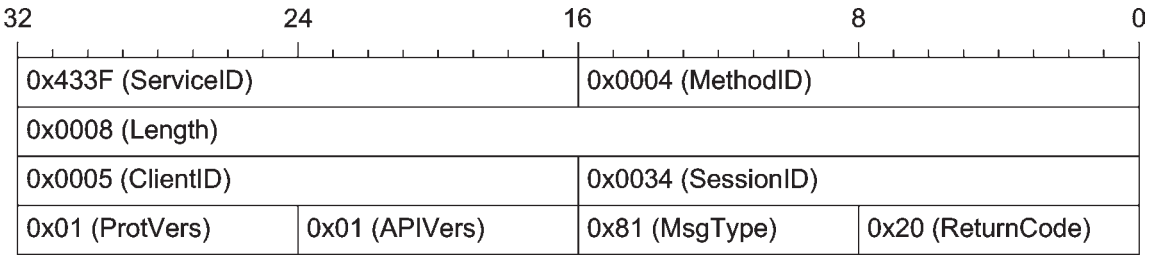


Figure 11 — SetCamControlAccess error (LOCKED_BY_FOREIGN_INSTANCE)

6.8.4 Event group handling

The event group handling shall follow the requirements below.

— An ISO 17215 camera shall support notifications for video data and histogram data. The corresponding EventIDs and EventGroupIDs are computed from the index of the underlying ROI to which the data pertains (see above).

— After a ROI was set up, at start-up (from a PSE) or at run time (by using the setRegionOfInterest and setVideoFormat or setHistogramFormat methods), the camera shall publish the corresponding eventgroups.

— If a ROI has been deleted, the camera shall inform other nodes in network about this, using a StopPublishEventgroup SD message.

— The eventgroups for “all video data”, “all histogram data”, and “all data” shall not be published explicitly.

— The client ECU shall send a Subscribe message with the EventGroupID of interest to subscribe to video or histogram data.

— The client ECU shall send a StopSubscribe message with the EventGroupID of interest to unsubscribe video or histogram data.

An example PDU for a Subscribe message is shown in Figure 12.

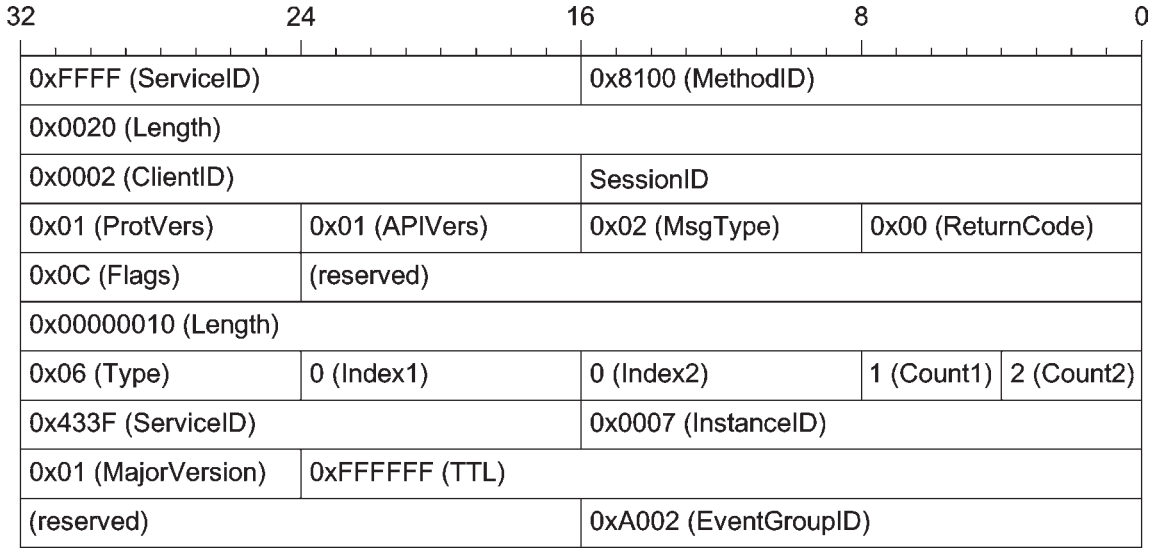


Figure 12 — Subscribe message example

6.9 PDU examples for some/IP

6.9.1 Request and response sequence (SOME/IP)

Example for the service, see 6.7.30.

— An ECU to read out the content of a certain register of the cameras imager

— The method is identified by its MethodID and can be described in a formal scheme

— ServiceID 0x433F, VideoCommunication

— MethodID 0x0303, getCamRegister

— Parameter, IN unit16 regAddress, OUT uint16 regValue

The PDU diagram for the request using SOME/IP looks like this. IP and UDP header are not visualized in the diagram. The client ID is provided by the supplier of the camera. The SessionID starts with 0 and is incremented by each further call.

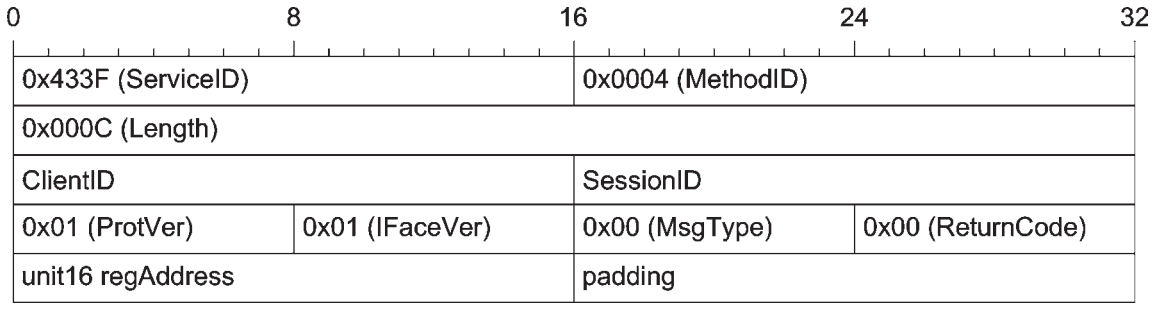


Figure 13 — PDU diagram for the request

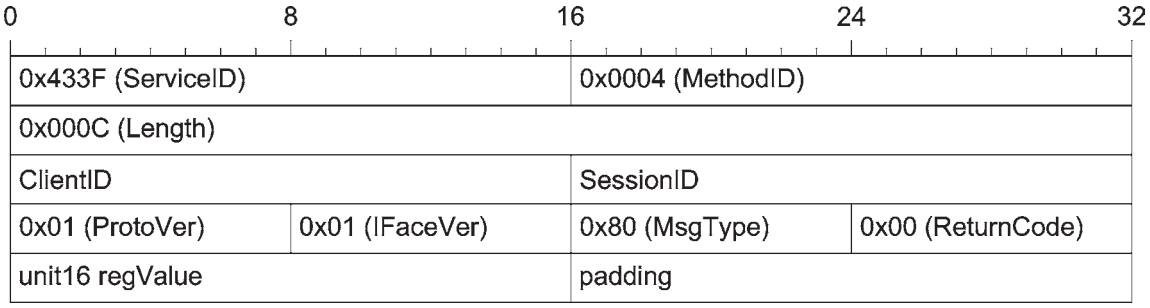


Figure 14 — PDU diagram for response (success)

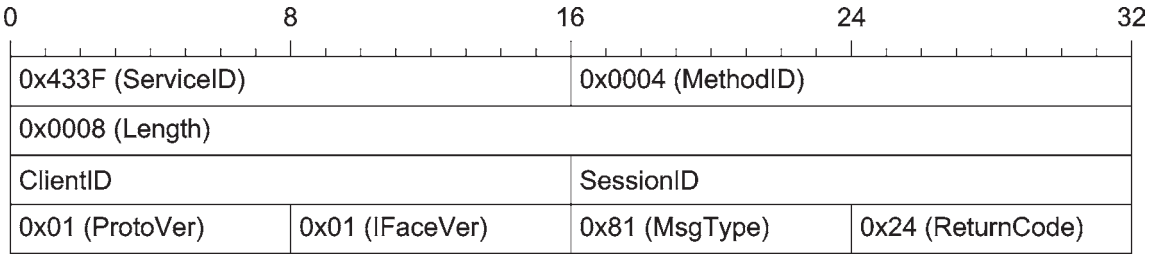


Figure 15 — PDU diagram for response (error 0x24 no such register)

ICS 43.040.15

Price based on 46 pages