INTERNATIONAL STANDARD

ISO 17215-3

Second edition
2021-05

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 17215-3:2021(E)

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

Contents

Page

Foreword ..... **v**
Introduction ..... **vi**
1 Scope **1**
2 Normative references **1**
3 Terms and definitions **1**
4 Symbols and abbreviated terms **2**
5 Conventions **2**
6 Overview of ISO 17215 series **3**
6.1 General 3
6.2 Document overview and structure 3
6.3 Open Systems Interconnection (OSI) model 4
6.4 Document reference according to OSI model 4
7 Camera application interface (OSI layer 7) **6**
7.1 Specific properties 6
7.2 API principles 6
7.2.1 Image cropping and windowing 7
7.3 API data types 7
7.4 API Return codes 8
7.5 API enumerations 8
7.5.1 Enumeration eMethodID 8
7.5.2 Enumeration eEventGroupType 9
7.5.3 Enumeration eCamErrorCodes 10
7.5.4 Enumeration eCameraMode 10
7.5.5 Enumeration eControlIndex 11
7.5.6 Enumeration eControlSupportedModes 11
7.5.7 Enumeration eControlMode 11
7.5.8 Enumeration ePersistentStorageID 12
7.6 API structures 12
7.6.1 Structure sPixelPosition 12
7.6.2 Structure sPixelMap 13
7.6.3 Structure sRectangle 13
7.6.4 Structure sImageDimension 14
7.6.5 Structure sImagerRegister 14
7.6.6 Structure sImagerRegisterBlock 14
7.6.7 Structure sImagerCharacteristic 14
7.6.8 Structure sIntrinsicCamParam 15
7.6.9 Structure sExtrinsicCamParam 16
7.6.10 Structure sPersistentEntryList 17
7.6.11 Structure sPersistentStorageEntry 17
7.6.12 Structure sTimeStamp 18
7.6.13 Structure sDatasheet 18
7.6.14 Structure sRegionOfInterest 18
7.6.15 Structure sVideoFormat 20
7.6.16 Structure sHistogramFormat 21
7.6.17 Structure sHistogramContent 22
7.6.18 Structure sVideoContent 22
7.6.19 Structure sControlMode 23
7.6.20 Structure sUnsignedCtl 23
7.6.21 Structure sSignedCtl 23
7.6.22 Structure sCombinedCtl 24
7.6.23 Structure sCamControl 24
7.6.24 Structure sCamStatus 24

7.6.25 Temperature ... 25
7.7 API reference ... 28
7.7.1 getDataSheet (MethodID 0x0001) ... 28
7.7.2 getCamStatus (MethodID 0002_{16}) ... 28
7.7.3 setCamMode (MethodID 0003_{16}) ... 28
7.7.4 setCamExclusive (MethodID 0011_{16}) ... 29
7.7.5 eraseCamExclusive (MethodID 0019_{16}) ... 29
7.7.6 setHostParameters (MethodID 0022_{16}) ... 30
7.7.7 getHostParameters (MethodID 0024_{16}) ... 30
7.7.8 eraseHostParameters (MethodID 0029_{16}) ... 31
7.7.9 setRegionOfInterest (MethodID 0101_{16}) ... 31
7.7.10 setRegionsOfInterest (MethodID 0102_{16}) ... 32
7.7.11 getRegionOfInterest (MethodID 0103_{16}) ... 32
7.7.12 getRegionsOfInterest (MethodID 0104_{16}) ... 32
7.7.13 eraseRegionOfInterest (MethodID 0109_{16}) ... 33
7.7.14 setVideoFormat (MethodID 0111_{16}) ... 33
7.7.15 getVideoFormat (MethodID 0113_{16}) ... 34
7.7.16 eraseVideoFormat (MethodID 0119_{16}) ... 34
7.7.17 setHistogramFormat (MethodID 0121_{16}) ... 35
7.7.18 getHistogramFormat (MethodID 0123_{16}) ... 35
7.7.19 eraseHistogramFormat (MethodID 0129_{16}) ... 36
7.7.20 SubscribeROIVideo (MethodID 0131_{16}) ... 36
7.7.21 UnSubscribeROIVideo (MethodID 0132_{16}) ... 37
7.7.22 SubscribeROIHistogram (MethodID 0x0133) ... 37
7.7.23 UnSubscribeROIHistogram (MethodID 0x0134) ... 37
7.7.24 setCamControl (MethodID 0201_{16}) ... 38
7.7.25 setCamControls (MethodID 0202_{16}) ... 38
7.7.26 getCamControl (MethodID 0203_{16}) ... 38
7.7.27 getCamControls (MethodID 0204_{16}) ... 39
7.7.28 setCamRegister (MethodID 0301_{16}) ... 39
7.7.29 setCamRegisters (MethodID 0302_{16}) ... 40
7.7.30 getCamRegister (MethodID 0303_{16}) ... 40
7.7.31 getCamRegisters (MethodID 0304_{16}) ... 41
7.7.32 setUsedRegisterSet (MethodID 0305_{16}) ... 41
7.8 Programming model for SOME/IP ... 42
7.8.1 General ... 42
7.8.2 Startup behaviour ... 43
7.8.3 Service discovery ... 43
7.8.4 Event group handling ... 46
7.9 PDU examples for SOME/IP ... 47
7.9.1 Request and response sequence (SOME/IP) ... 47
Bibliography ... **49**

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 of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT), see www.iso.org/iso/foreword.html.

This document was prepared by Technical Committee ISO/TC 22, *Road vehicles*, Subcommittee SC 31, *Data communication*.

This second edition cancels and replaces the first edition (ISO 17215-3:2014), which has been technically revised.

The main changes compared to the previous edition are as follows:

— corrections of Formulae and scaling in 7.6.8;

— editorial adoptions and corrections.

A list of all parts in the ISO 17215 series can be found on the ISO website.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html.

Introduction

Driver assistance systems are increasingly common in road vehicles. From the beginning, cameras were part of this trend. Analogue cameras were used in the beginning because of the 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) started being used in vehicles.

Currently, the market demands such systems with multiple functions. Even different viewing directions are in use. It seems plausible that in the near future a single vehicle could have between 6 and 12 cameras. For this reason and others like limitations in size, power consumption, etc., designs have been made where the cameras are separated from the processing unit. Therefore, a high-performance digital interface between camera and processing unit is necessary.

This document 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-the-art video communication interface for cameras, usable for operating and diagnosis purposes,

— 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 document is related to the general information and use case definition. This is a general overview document 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 document, in accordance with Table 1 are broken into following layers:

— application (layer 7), specified in this document (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 document not related to the OSI model.

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

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

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 another future communication document.

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

Part 3:
Camera message dictionary

1 Scope

This document specifies the standardized camera messages and data types used by a VCIC camera (OSI layer 7).

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 is independent on the camera used. The services can be implemented by general libraries.

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

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO 17215-1, *Road vehicles — Video communication interface for cameras (VCIC) — Part 1: General information and use case definition*

ISO 17215-2, *Road vehicles — Video communication interface for cameras (VCIC) — Part 2: Service discovery and control*

ISO 17215-4, *Road vehicles — Video communication interface for cameras (VCIC) — Part 4: Implementation of communication requirements*

3 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO 17215-1 and the following apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp

— IEC Electropedia: available at http://www.electropedia.org/

3.1
extrinsic parameter
parameter that denotes the coordinate system transformations from 3D world (vehicle) coordinates (m,°) to 3D-camera coordinates (m,°)

3.2
frame rate
update rate per time of camera images

3.3
global shutter
exposure that exposes all pixels at the same time

3.4
histogram
type of chart that acts as a graphical representation of the tonal distribution in a digital image

3.5
intrinsic camera parameter
parameter that denotes the coordinate system transformations from 3D camera (m) to 2D pixel coordinates (pixel)

4 Symbols and abbreviated terms

Abbreviated term	Symbol	Full term
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
ID		Identifier
ISO		International Organization for Standardization
LDR		Low Dynamic Range
LSB		Least significant bit
MAC		Media Access Control
OSI		Open Systems Interconnection
PSE		Persistent storage entry
ROI	ρ	Region of Interest, i.e. sub-part of overall image
RPC		Remote Procedure Call

5 Conventions

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

6 Overview of ISO 17215 series

6.1 General

This document has been established to implement a standardized video communication interface for cameras on a communication data link.

The focus of this document is to use existing protocols.

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

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

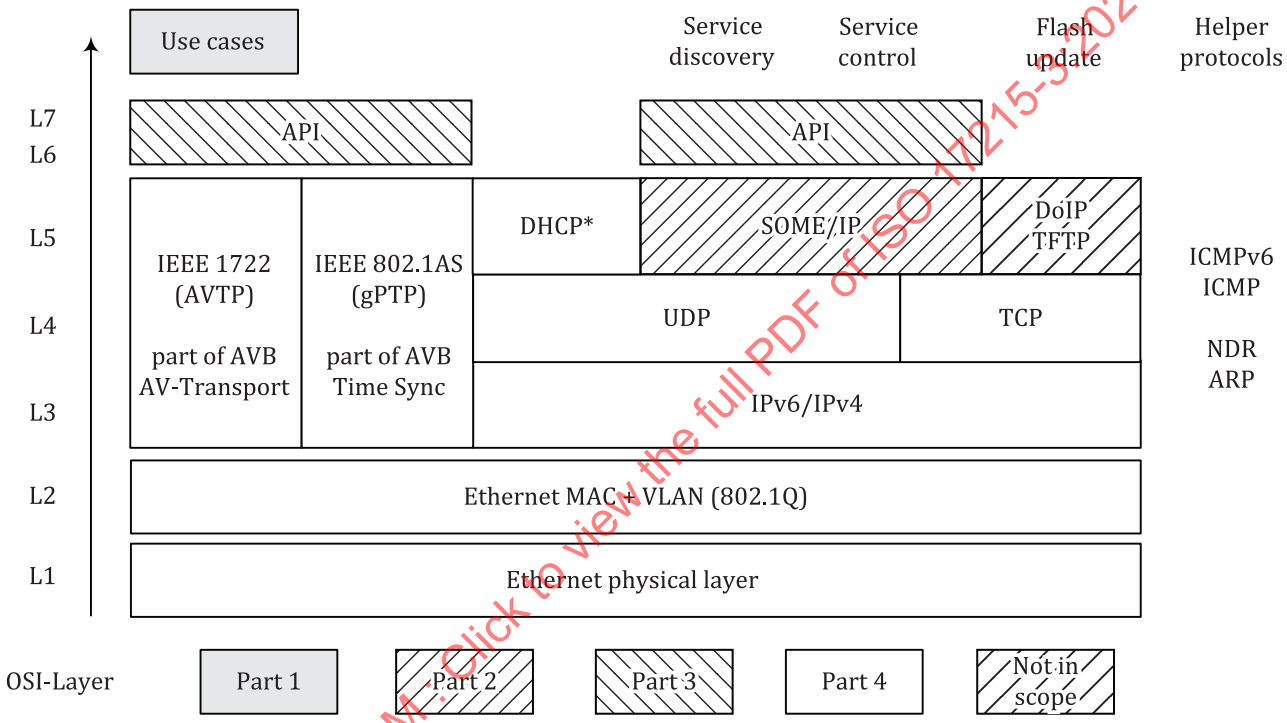


Figure 1 — Overview of the ISO 17215 series

6.2 Document overview and structure

The ISO 17215 series consists of a set of four parts, which provide all references and requirements to support the implementation of a standardized video communication interface for cameras according to the document 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.

— This document 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).

6.3 Open Systems Interconnection (OSI) model

This document 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 the ISO 17215 series 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 document. It also shows how the OSI service conventions have been applied to this document.

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 document.

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 document 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.

6.4 Document reference according to OSI model

Figure 2 illustrates the document references.

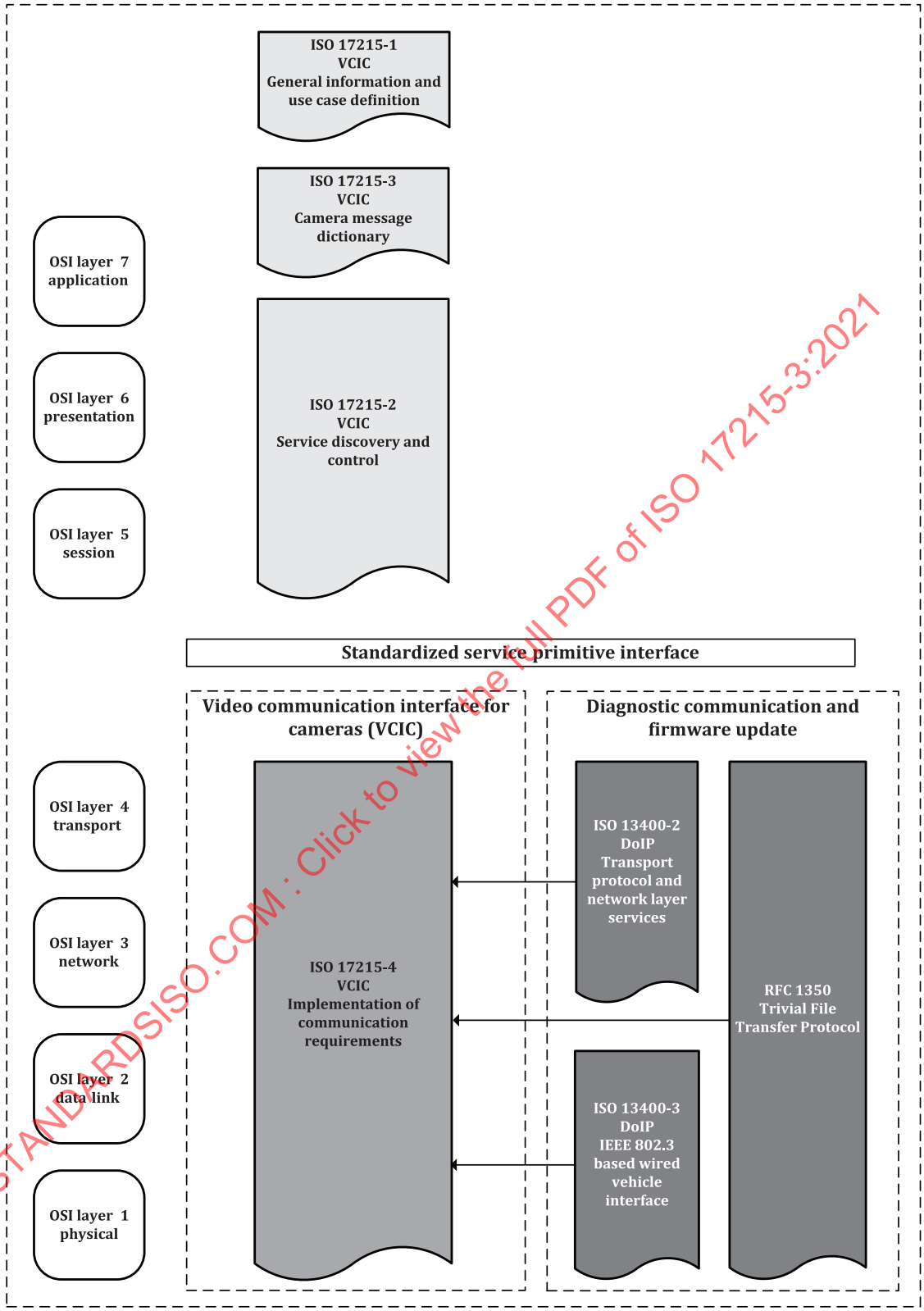


Figure 2 — Document reference according to OSI model

7 Camera application interface (OSI layer 7)

7.1 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 is detected during development, assembling, or repair of a car.

Consequently, the standard specifies the interface to an automotive camera, but does not 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.

7.2 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 Formula (1):

$$F_{\text{Camera}} = f\{C_{\text{Instance}}, F_{\text{API}}\{I_{\text{Method}}, a_1, \ldots, a_n\}\} \quad (1)$$

where

F_{Camera} is the camera function;

f is the mathematical sign for function;

C_{Instance} is the camera instance;

F_{API} is the API function;

I_{Method} is the Method ID;

a_1 is the argument 1;

a_n is the argument n;

The implementation depends highly on the programming language used. Therefore, this document 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 their Method ID.

The Method IDs are specified in 7.5.1.

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 follow.

Events are identified by its respective Eventgroup ID.

The Eventgroup IDs are specified in 7.5.2.

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

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

— general camera functions (MethodID 0001_{16} –$00FF_{16}$);

— video format functions (MethodID 0101_{16} –$01FF_{16}$);

— image control functions (MethodID 0201_{16} –$02FF_{16}$), and;

— imager functions (MethodID 0301_{16} –$03FF_{16}$).

7.2.1 Image cropping and windowing

The definition of the image windows is shown in Figure 3.

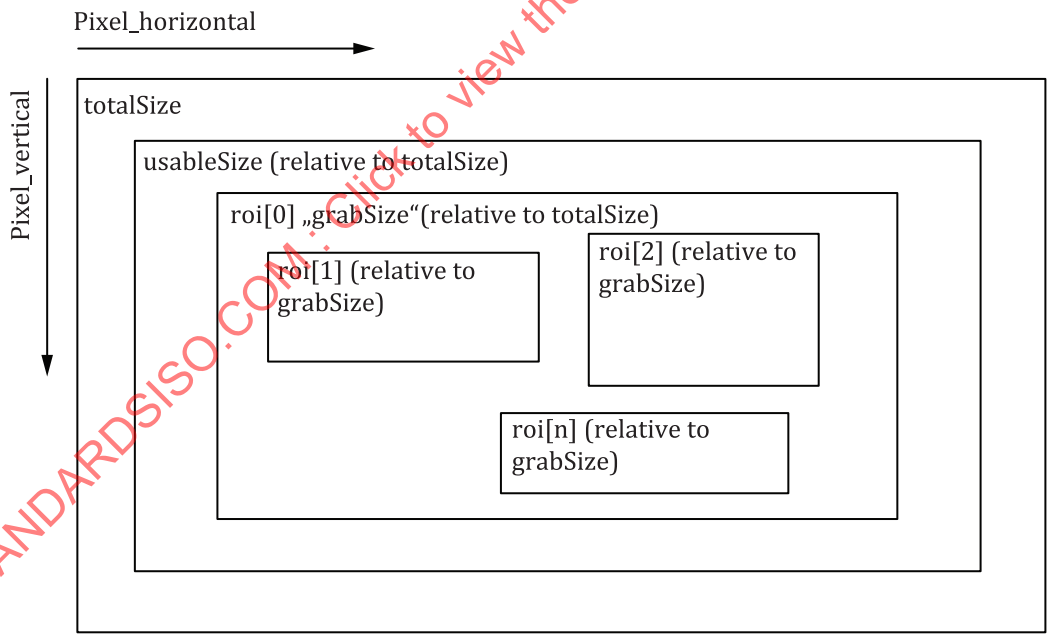


Figure 3 — Image cropping and windowing

7.3 API data types

All used data types are specified in ISO 17215-2:2014, 6.3.1.

7.4 API Return codes

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

— The return codes (00_{16} to $1F_{16}$) are defined in ISO 17215-2:2014, 6.1.1.7.

— The return codes (20_{16} to $3F_{16}$) are function-specific and are described in 7.5.3.

7.5 API enumerations

7.5.1 Enumeration eMethodID

Table 2 provides enumeration that 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	0001_{16}	Returns the datasheet of the camera.
getCamStatus	0002_{16}	Returns the current status of the camera.
setCamMode	0003_{16}	Start, stops, and restarts the camera application in the camera.
setCamExclusive	0011_{16}	Assigns the control of the camera exclusively to the requesting client.
eraseCamExclusive	0019_{16}	Removes the exclusive control lock for the requesting client.
setHostParameters	0022_{16}	Sets the host parameters using persistent storage entries.
getHostParameters	0024_{16}	Gets the requested host parameter by reading persistent storage entries.
eraseHostParameters	0029_{16}	Forces the camera to erase the requested host parameters addressed by the PSE ID.
setRegionOfInterest	0101_{16}	Sets the parameter for a region of interest addressed by index.
setRegionsOfInterest	0102_{16}	Sets the parameter for all supported regions of interest.
getRegionOfInterest	0103_{16}	Returns the parameter for region of interest addressed by ROI index.
getRegionsOfInterest	0104_{16}	Returns the parameter for all supported region of interest.
eraseRegionOfInterest	0109_{16}	Erases all parameter for the requested ROI.
setVideoFormat	0111	Sets the video format for a ROI addressed by ROI index.
getVideoFormat	0113_{16}	Reads the parameter of the current video format, addressed by the ROI index.
eraseVideoFormat	0119_{16}	Erases all video format parameter for the requested ROI.
setHistogramFormat	0121_{16}	It is the histogram format parameter for a ROI addressed by ROI index.
getHistogramFormat	0123_{16}	It is the parameter of the current histogram format, addressed by the ROI index.
eraseHistogramFormat	0129_{16}	Erases all histogram format parameter for the requested ROI.
SubscribeROIVideo	0131_{16}	Starts the transmission of a video stream for the requested ROI.
UnSubscribeROIVideo	0132_{16}	Stops the video streaming for the requested ROI.
SubscribeROIHistogram	0133_{16}	Starts the transmission of the histograms for the requested ROI.
UnSubscribeROIHistogram	0134_{16}	Signs off the transmission of the histograms for the requested ROI.
setCamControl	0201_{16}	Sets the parameter for a camera control addressed by the control index.
setCamControls	0202_{16}	Sets the parameters for all camera controls.

Table 2 *(continued)*

Name	Value	Description
getCamControl	0203_{16}	Returns the current parameter for a camera control addressed by the control index.
getCamControls	0204_{16}	Returns all current camera control parameters.
setCamRegister	0301_{16}	Writes the content of a register of the camera imager addressed by physical register address.
setCamRegisters	0302_{16}	Writes (atomic access) the content of a register block of the camera imager.
getCamRegister	0303_{16}	Reads the content of a register of the camera imager address by the physical register address.
getCamRegisters	0304_{16}	Reads the content of a register block of the camera imager.
setUsedRegisterSet	0305_{16}	Forces the camera to write the imager register set stored in the requested PSE to the imager.

7.5.2 Enumeration eEventGroupType

Table 3 provides enumeration that 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
E_LOCKED_BY_FOREIGN_INSTANCE	20_{16}	Camera service is already locked by another client.
E_LOCK_ EXPIRED	21_{16}	The camera lock has expired.
E_NOT_LOCKED	22_{16}	Camera is not locked.
E_INVALID_PS_ENTRY	24_{16}	The requested PSE ID is unknown.
E_INVALID_PS_OPERATION	25_{16}	The requested PSE operation is not allowed, e.g. store on a RO PSE.
E_INVALID_PS_DATA	26_{16}	The PSE contains a CRC16 error.
E_NO_MORE_SPACE	27_{16}	There is no more space available to store the PSE.
E_INVALID_ROI_INDEX	30_{16}	The requested ROI is out of range.
E_INVALID_ROI_NUMBER	31_{16}	The requested number of ROIs is out of range, defined by sDatasheet.numOfRegionOfInterest.
E_INVALID_VIDEO_FORMAT	32_{16}	It is an invalid value in video format.
E_INVALID_HISTOGRAM_FORMAT	33_{16}	It is an invalid value in histogram format.
E_INVALID_CONTROL_INDEX	35_{16}	The requested camControlIndex is out of range or not supported by the camera.
E_INVALID_CONTROL_MODE	36_{16}	The requested control mode is not supported by the camera.
E_INVALID_CONTROL_VALUE	37_{16}	The requested control value is out of the range, defined in sCamControl.
E_INVALID_REGISTER_ADDRESS	38_{16}	The register address is not supported by the imager.
E_INVALID_REGISTER_VALUE	39_{16}	The value for the given register address is not supported by the imager.
E_INVALID_REGISTER_OPERATION	$3A_{16}$	The requested operation (read or write) for the given register address is not supported by the imager.

7.5.3 Enumeration eCamErrorCodes

Table 4 provides enumeration that defines the camera API specific return codes. The values are in the range of 20_{16} to $3F_{16}$.

Table 4 — Enumeration eCamErrorCodes

Name	Value	Description
E_LOCKED_BY_FOREIGN_INSTANCE	20_{16}	Camera service is already locked by another client.
E_LOCK_EXPIRED	21_{16}	The camera lock has expired.
E_NOT_LOCKED	22_{16}	Camera is not locked.
E_INVALID_PS_ENTRY	24_{16}	The requested PSE ID is unknown.
E_INVALID_PS_OPERATION	25_{16}	The requested PSE operation is not allowed, e.g. store on a RO PSE.
E_INVALID_PS_DATA	26_{16}	The PSE contains a CRC16 error.
E_NO_MORE_SPACE	27_{16}	There is no more space available to store the PSE.
E_INVALID_ROI_INDEX	30_{16}	The requested ROI is out of range.
E_INVALID_ROI_NUMBER	31_{16}	The requested number of ROIs is out of range, defined by sDatasheet.numOfRegionOfInterest.
E_INVALID_VIDEO_FORMAT	32_{16}	It is an invalid value in video format.
E_INVALID_HISTOGRAM_FORMAT	33_{16}	It is an invalid value in histogram format.
E_INVALID_CONTROL_INDEX	35_{16}	The requested camControlIndex is out of range or not supported by the camera.
E_INVALID_CONTROL_MODE	36_{16}	The requested control mode is not supported by the camera.
E_INVALID_CONTROL_VALUE	37_{16}	The requested control value is out of the range, defined in sCamControl.
E_INVALID_REGISTER_ADDRESS	38_{16}	The register address is not supported by the imager.
E_INVALID_REGISTER_VALUE	39_{16}	The value for the given register address is not supported by the imager.
E_INVALID_REGISTER_OPERATION	$3A_{16}$	The requested operation (read or write) for the given register address is not supported by the imager.

7.5.4 Enumeration eCameraMode

Table 5 provides an enumeration type that 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 (CC OFF, Communication-Controller OFF) and stores the settings. This state can be only left by repowering or wake up on LAN.

7.5.5 Enumeration eControlIndex

Table 6 defines the enumeration eControlIndex.

Table 6 — Enumeration eControlIndex

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

7.5.6 Enumeration eControlSupportedModes

Table 7 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 is 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.

7.5.7 Enumeration eControlMode

Table 8 provides an enumeration type that 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

7.5.8 Enumeration ePersistentStorageID

The persistent storage entry identifiers as in Table 9 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 0000_{16} to $7FFF_{16}$. The manufacturer specific identifiers are in the range of 8000_{16} to $FFFF_{16}$.

Table 9 — Enumeration ePersistentStorageID

Name	Value	Description
UseAtBootTime	0000_{16}	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	0010_{16}	The payload consists of imager type specific information. The structure sImagerCharacteristic shall be used and be implemented as read only entry.
Defectpixelmap	0020_{16}	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	0030_{16}	The payload consists of the intrinsic calibration parameter. The structure sIntrinsicCamParam shall be used and might be implemented as read only entry.
ExtrinsicCamParam	0040_{16}	The payload consists of the extrinsic calibration parameter. The structure sExtrinsicCamParam shall be used.
ROISetting	$1yyx_{16}$	The payload consists of the parameter set for a ROI. The structure sRegionOfInterest shall be used, y = roiIndex, x = number of entries belonging to the roiIndex.
ImagerRegisterBlock	$20xx_{16}$	The payload consists of a block of imager registers. The structure sImagerRegisterBlock shall be used xx = number of entries.

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 1012_{16}, for example, identifies the second entry for the ROI with the index 1. Up to 256 PSEs for imager register parameter are allowed.

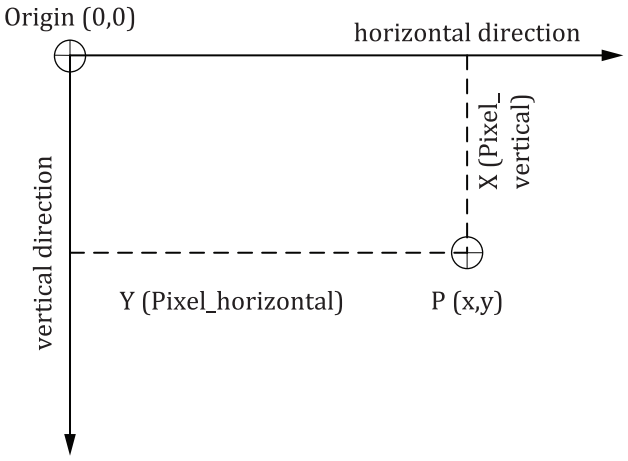
The boot parameter entry contains a list of PSEs. Only existing entries with the ID ROISetting and ImagerRegisterBlock are valid for this PSE ID. If this entry exists, the camera shall use content of the listed PSE as initial parameter set.

7.6 API structures

7.6.1 Structure sPixelPosition

Table 10 defines the position of a single pixel relative to a window as in 7.2.1. 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

7.6.2 Structure sPixelMap

The structure in Table 11 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

7.6.3 Structure sRectangle

Table 12 defines a rectangular area using two pixels coordinates (see also Figure 5).

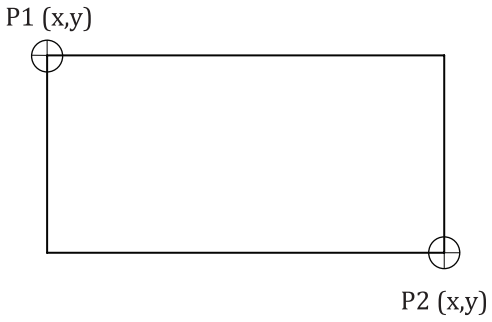


Figure 5 — sRectangle

Table 12 — Structure sRectangle

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

7.6.4 Structure sImageDimension

Table 13 defines the image dimension.

Table 13 — Structure sImageDimension

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

7.6.5 Structure sImagerRegister

Table 14 defines 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.

7.6.6 Structure sImagerRegisterBlock

Table 15 defines 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

7.6.7 Structure sImagerCharacteristic

Table 16 defines the characteristics of the camera.

Table 16 — Structure sImagerCharacteristic

Name	Type	Description
totalSize	sImageDimension	Maximum size of the imager's pixel array, P1 is always 0,0; see 7.2.1.
usableSize	sImageDimension	Usable size of the imager's pixel array, coordinates are relative to total size; see 7.2.1.
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

7.6.8 Structure sIntrinsicCamParam

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 the 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 — Structure sIntrinsicCamParam

Name	Type	Description
distortionModel	unit8	This parameter indicates in later editions of this document the distortion model used. It is fixed to 0 in this document.
pad1	unit8	Unused
pad2	unit8	Unused
pad3	unit8	Unused
c_x	float64	x-coordinate of the principal point [pixel]
c_y	float64	y-coordinate of the principal point [pixel]
f_x	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)
f_y	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 Formulae (2) to (8).

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

where

x is the x-axis value of the 3-D vector;

z is the z-axis value of the 3-D vector.

$$y' = \frac{y}{z} \tag{3}$$

where

y is the y-axis value of the 3-D vector.

$$r^2 = x'^2 + y'^2 \tag{4}$$

where

r is an auxiliary variable.

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

where

k_1, k_2, k_3, k_4, k_5, k_6, p_1 and p_2 are defined in Table 17.

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

$$u = f_x \times x'' + c_x \tag{7}$$

$$v = f_y \times y'' + c_y \tag{8}$$

where

u is the horizontal pixel position;

v is the vertical pixel position;

f_x, f_y, c_x and c_y are defined in Table 17.

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) with (u, v) = (0, 0) denoting the top left pixel of the image.

7.6.9 Structure sExtrinsicCamParam

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 the 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 18.

Table 18 — Structure sExtrinsicCamParam

Name	Type	Description
R	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 Formulae (9) to (11):

$$x = R[0] \times X + R[1] \times Y + R[2] \times Z + t_{vec}[0] \tag{9}$$

$$y = R[3] \times X + R[4] \times Y + R[5] \times Z + t_{vec}[1] \tag{10}$$

$$z = R[6] \times X + R[7] \times Y + R[8] \times Z + t_{vec}[2] \tag{11}$$

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 document.

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.

7.6.10 Structure sPersistentEntryList

The structure in Table 19 defines the content of a persistent storage entry (PSE) used in the set/get/restore host parameter functions.

Table 19 — Structure sPersistentEntryList

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

7.6.11 Structure sPersistentStorageEntry

The structure in Table 20 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 7.5.8.
PersistentData	uint8(Length)	Persistent data, the structure of the persistent data is 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.

7.6.12 Structure sTimeStamp

The structure in Table 21 defines 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	It is the 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 0002_{16} and nanosecondsField = 0000 0001_{16}.

7.6.13 Structure sDatasheet

The structure in Table 22 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 document
imagerCharacteristic	sImagerCharacteristic	Description of the camera's 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 7.6.14
maxLengthOfPersistentEntry	unit32	Maximum allowed length of a PSE

7.6.14 Structure sRegionOfInterest

The structure in Table 23 defines a rectangular region of interest. It is used to specify the parameter for an Event ID (see Table 24) 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 7.6.15.
histogramFormat	sHistogramFormat	For more details, see 7.6.16.

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 eventGroupID is derived from the roiIndex and the eventGroupType as defined by Formula (12).

$$E_{\text{groupID}} = E_{\text{groupType}} + (\rho_{\text{index}} + 1) \tag{12}$$

where

E_{groupID} is the eventGroupID;

$E_{\text{groupType}}$ is the eventGroupType;

ρ_{index} is the roiIndex.

Table 24 — Event ID

Name	Value	Description
VideoStream1	9001_{16}	Event ID for video events related to ROI index 0 (main video stream)
VideoStream2	9002_{16}	Event ID for video events related to ROI index 1
:		
VideoStreamN	$900n_{16}$	Event group ID for events related ROI index n, given by sDatasheet.numOfRegionOfInterest-1
Histogram1	$A001_{16}$	Event ID for histogram events related to ROI index 0
Histogram2	$A002_{16}$	Event ID for histogram events related to ROI index 1
:		
HistogramN	$A00n_{16}$	Event ID for histogram events related ROI index n, given by sDatasheet.numOfRegionOfInterest-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 7.6.15). 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 7.6.16 and 7.6.18

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

7.6.15 Structure sVideoFormat

The structure in Table 25 defines the parameter of the video format associated to certain ROI.

Table 25 — Structure sVideoFormat

Name	Type	Description
videoFormatEnabled	uint8	0: disabled 1: enabled
transmissionMethod	uint8	0: use AVTP packages 1: use notification packages
transmissionCycle	uint16	0: only once Every *n* image > 0
WidthAndHeight	sImageDimension	It is the 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}
Interlaced	uint8	0: progressive scan video 1: interlaced video
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.

7.6.16 Structure sHistogramFormat

The structure in Table 26 defines the histogram format used for a certain ROI.

Table 26 — Structure sHistogramFormat

Name	Type	Description
histogramEnabled	uint8	Disabled = 0 Enabled = 1
histogramUpdateCycle	uint8	Defines how often a histogram is calculated. Only once = 0 Every *n* image > 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
d_{type}	uint8	dataType 2: unit16 4: unit32
B_{size}	uint8	The size of the bin 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.
N_{bins}	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 (13):

$$N_{bins} = \frac{\left(2^{b_{depth}}\right)}{B_{size}} \tag{13}$$

where

b_{depth} is the depth of the used bits.

EXAMPLE 1 $B_{size} = 1$, $b_{dept} = 10$, $\rho_{width} = 128$, $\rho_{height} = 128$, and Formula (14)

$$N_{pixels} = \rho_{width} \times \rho_{height} \tag{14}$$

where

ρ_{width} is ROI width;

ρ_{height} is ROI height.

Using Formula (14) we obtain the following:

$$N_{pixels} = 128 \times 128 = 16\ 384$$

Therefore, the histogram data type of uint16 is chosen ($d_{type} = 2$).

The number of histogram bins is then given by:

$$N_{\text{bins}} = \left(\frac{2^{b_{\text{depth}}}}{B_{\text{size}}} \right) \times d_{\text{type}} \tag{15}$$

Using Formula (15) we obtain the following:

$$N_{\text{bins}} = \left(\frac{2^{10}}{1} \right) \times 2$$

EXAMPLE 2 $B_{\text{size}} = 2$, $b_{\text{depth}} = 10$, $\rho_{\text{width}} = 1\ 024$, $\rho_{\text{height}} = 480$, and Formula (14):

$$N_{\text{pixels}} = 1\ 024 \times 480 = 491\ 520$$

Therefore, the histogram data type of uint32 is chosen ($d_{\text{type}} = 4$).

Using Formula (15) we obtain the following:

$$N_{\text{bins}} = \left(\frac{2^{10}}{2} \right) \times 4 = \left(\frac{1\ 024}{2} \right) \times 4 = 2\ 048$$

7.6.17 Structure sHistogramContent

The structure in Table 27 defines the format of a histogram notification package.

Table 27 — Structure sHistogramContent

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 7.6.12.
numberOfROIValues	uint32	numberOfHistValues specified by the ROI definition in bytes, see 7.6.16.
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

7.6.18 Structure sVideoContent

The structure in Table 28 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 7.6.12.
NumberOfROIValues	uint32	Width × height × bit depth given by sVideoFormat. colourSpace, see 7.6.15.
ROIContent	uint8 [nBins]	Uncompressed content of the ROI

If the sizes of the video content exceed 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

7.6.19 Structure sControlMode

The structure in Table 29 defines a camera control mode register.

Table 29 — Structure sControlMode

Name	Type	Description
SupportedModes	unit8	Use 7.5.6.
ControlMode	uint8	Use 7.5.7.

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

On write, the mode of operation changes 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.

7.6.20 Structure sUnsignedCtl

The structure in Table 30 defines an unsigned camera control.

Table 30 — Structure sUnsignedCtl

Name	Type	Description
ControlStatus	sControlMode	See 7.6.19.
ControlValue	uint16	For information on valid values, consider sCamControl.

7.6.21 Structure sSignedCtl

The structure in Table 31 defines a signed camera control.

Table 31 — Structure sSignedCtl

Name	Type	Description
ControlStatus	sControlMode	See 7.6.19.
ControlValue	sint16	For information on valid values, consider sCamControl.

7.6.22 Structure sCombinedCtl

The structure in Table 32 defines a combined camera control.

Table 32 — Structure sCombinedCtl

Name	Type	Description
ControlStatus	sControlMode	See 7.6.19.
ControlValueA	uint8	For information on valid values, consider sCamControl.
ControlValueB	uint8	For information on valid values, consider sCamControl.

7.6.23 Structure sCamControl

The structure in Table 33 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 % to +20 %, resolution 0,1 %
Gain	sSignedCtl	Controls the gain circuits of the camera Unit db, range –20 to +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 %
Heater	sUnsignedCtl	Controls the heater of the camera Range on - off
Temperature	sUnsignedCtl	Controls the temperature of the camera See 7.6.25 for more information.

7.6.24 Structure sCamStatus

The structure in Table 34 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

7.6.25 Temperature

7.6.25.1 General

Application requests the internal temperature of the camera.

The camera always has to respond upon request with the corresponding 16-bit temperature information. The camera puts the temperature information with the defined resolutions (7 bits to 15 bits) on the 16-bit word in Table 35. 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.

For example, for less temperature accuracy, only 2 °C per LSB resolution is necessary, therefore, only 8 bits (including signed indicator) are evaluated. Or if a high accuracy is necessary, a resolution of 0,007 8 °C per LSB can be used by evaluating the 16-bit value (including signed indicator) information.

Table 35 — Temperature resolution

Parameter	Resolution range
Min. resolution	0.007 8 °C per LSB
Max. resolution	2 °C per 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 provides an example for temperature ranges.

Table 36 — Example for temperature ranges

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

7.6.25.2 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

Table 37 *(continued)*

Bits	Description	Values
14:8	Temperature value standard 7 bits standard resolution (°C per LSB)	7 bits = 2 °C per LSB
7:0	Temperature value extended 8 bits extended resolution (°C per LSB) Depends on the application, on how many bits are used/analysed. Application can scale up to maximum resolution of 15 bits, if a high resolution has to be used.	8 bits = 1 °C per LSB 9 bits = 0,5 °C per LSB 10 bits = 0,25 °C per LSB 11 bits = 0,125 °C per LSB 12 bits = 0,0625 °C per LSB 13 bits = 0,03125 °C per LSB 14 bits = 0,015625 °C per LSB 15 bits = 0,0078125 °C per LSB

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

7.6.25.3 Formula to calculate the temperature value

Formula (16) is used to calculate the temperature value:

$$T = \alpha_{\text{res}} \times T_{\text{raw\_value\_LSB}} \tag{16}$$

where

T is the temperature in °C;

α_{res} is the resolution value for °C per bit;

$T_{\text{raw\_value\_LSB}}$ is the raw least significant byte of the temperature value.

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

7.6.25.4 Examples

In the following example in Table 38, 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 per LSB).

Table 38 — Temperature value calculation (positive values)

Setting		Temperature value calculation (positive values)		
Used bits resolution	Resolution (°C per 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
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

Table 38 *(continued)*

Setting		Temperature value calculation (positive values)		
Used bits resolution	**Resolution (°C per LSB)**	**16-bit binary value (incl. signed indicator)**	**Dec value (LSB)**	**Temperature (°C)**
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 (16):

$$T = 2\left(\frac{°C}{LSB}\right) \times 51(LSB) = +102(°C)$$

In this example (Table 39), 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 per LSB):

Table 39 — Temperature value calculation (negative values)

Setting		Temperature value calculation (negative values)		
Used bits resolution	**Resolution (°C per 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 (16):

$$T = 2\left(\frac{°C}{LSB}\right) \times -13(LSB) = -26(°C)$$

Table 40 shows the corresponding calculated temperature output value for a fixed resolution of 9 bits, which is linked to 0,5 °C per LSB, for some specific 16-bit temperature values.

Table 40 — Temperature value with fixed resolution

Setting		Temperature value with fixed resolution 0,5 °C per LSB		
Used bits resolution	**Resolution (°C per 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
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

Table 40 *(continued)*

Setting		Temperature value with fixed resolution 0,5 °C per LSB		
Used bits resolution	**Resolution (°C per LSB)**	**16-bit binary value (incl. signed indicator)**	**Dec value (LSB)**	**Temperature (°C)**
9 bits of 15 bits	0,5	1110 0100 1000 0000	–110	–55

The temperature value is then calculated with Formula (16):

$$T = 0{,}5\left(\frac{°C}{LSB}\right) \times 250(LSB) = +125(°C)$$

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

7.7 API reference

7.7.1 getDataSheet (MethodID 0x0001)

Purpose: This method returns the datasheet of the camera.

Input parameters: None

Output parameters: See Table 41.

Table 41 — Output parameters getDataSheet (MethodID 0x0001)

Name	Type	Description
Datasheet	sDatasheet	Datasheet of the camera

Specific error codes: There are no specific error codes.

7.7.2 getCamStatus (MethodID 0002_{16})

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

Input parameters: None

Output parameters: See Table 42.

Table 42 — Output parameters getCamStatus (MethodID 0002_{16})

Name	Type	Description
camStatus	sCamStatus	Current status of the camera

Specific error codes: There are no specific error codes.

7.7.3 setCamMode (MethodID 0003_{16})

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

A client shall lock the camera using the setCamExclusive method before requesting this method.

Input parameters: See Table 43.

Table 43 — Input parameters setCamMode (MethodID 0003_{16})

Name	Type	Description
cameraMode	uint32	Use enumeration eCameraMode

Specific error codes: See Table 44.

Table 44 — Error codes setCamMode (MethodID 0003_{16})

Error code
E_LOCKED_BY_FOREIGN_INSTANCE
E_LOCK_ EXPIRED

7.7.4 setCamExclusive (MethodID 0011_{16})

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 issues 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 $FFFFFFFF_{16}$, the camera shall use its default timeout parameter instead.

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

Input parameters: See Table 45.

Table 45 — Input parameters setCamExclusive (MethodID 0011_{16})

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

Output parameters: None

Specific error codes: See Table 46.

Table 46 — Error code setCamExclusive (MethodID 0011_{16})

Error code
E_LOCKED_BY_FOREIGN_INSTANCE

7.7.5 eraseCamExclusive (MethodID 0019_{16})

Purpose: This method removes the exclusive control lock 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 00000000_{16}.

When the Instance ID does not match the lockedBy value, the camera shall report the E_LOCKED_BY_FOREIGN_INSTANCE error.

Input parameters: None

Output parameters: None

Specific error codes: See Table 47.

Table 47 — Error code eraseCamExclusive (MethodID 0x0019)

Error code
E_LOCKED_BY_FOREIGN_INSTANCE
E_NOT_LOCKED

7.7.6 setHostParameters (MethodID 0022_{16})

Purpose: Sets the host parameters using persistent storage entries.

A client shall lock the camera using the setCamExclusive method before requesting this method.

Input parameters: See Table 48.

Table 48 — Input parameters setHostParameters (MethodID 0022_{16})

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

Specific error codes: See Table 49.

Table 49 — Error codes setHostParameters (MethodID 0022_{16})

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

7.7.7 getHostParameters (MethodID 0024_{16})

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

Input parameters: See Table 50.

Table 50 — Input parameters getHostParameters (MethodID 0024_{16})

Name	Type	Description
persistentEntryList	sPersistentEntryList	List of requested PSE IDs

Output parameters: See Table 51.

Table 51 — Output parameters getHostParameters (MethodID 0024_{16})

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

Specific error codes: See Table 52.

Table 52 — Error codes getHostParameters (MethodID 0024_{16})

Error code
E_INVALID_PS_ENTRY

7.7.8 eraseHostParameters (MethodID 0029_{16})

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

A client shall lock the camera using the setCamExclusive method before requesting this method.

Input parameters: See Table 53.

Table 53 — Input parameters eraseHostParameters (MethodID 0029_{16})

Name	Type	Description
persistentEntryList	sPersistentEntryList	List of requested PSE IDs

Output parameters: None

Specific error codes: See Table 54.

Table 54 — Error codes eraseHostParameters (MethodID 0029_{16})

Error code
E_INVALID_PS_ENTRY
E_INVALID_PS_OPERATION
E_LOCKED_BY_FOREIGN_INSTANCE
E_LOCK_ EXPIRED

7.7.9 setRegionOfInterest (MethodID 0101_{16})

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

A client shall lock the camera using the setCamExclusive method before requesting this method.

Input parameters: See Table 55.

Table 55 — Input parameters setRegionOfInterest (MethodID 0101_{16})

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: See Table 56.

Table 56 — Error parameters setRegionOfInterest (MethodID 0101_{16})

Error code
E_INVALID_ROI_INDEX
E_INVALID_HISTOGRAM_FORMAT
E_INVALID_VIDEO_FORMAT
E_LOCKED_BY_FOREIGN_INSTANCE
E_LOCK_EXPIRED

7.7.10 setRegionsOfInterest (MethodID 0102_{16})

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

A client shall lock the camera using the setCamExclusive method before requesting this method.

Input parameters: See Table 57.

Table 57 — Input parameters setRegionsOfInterest (MethodID 0102_{16})

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: See Table 58.

Table 58 — Error codes setRegionsOfInterest (MethodID 0102_{16})

Error code
E_INVALID_ROI_INDEX
E_INVALID_ROI_NUMBER
E_LOCKED_BY_FOREIGN_INSTANCE
E_LOCK_EXPIRED

7.7.11 getRegionOfInterest (MethodID 0103_{16})

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

Input parameters: See Table 59.

Table 59 — Input parameters getRegionOfInterest (MethodID 0103_{16})

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

Output parameters: See Table 60.

Table 60 — Output parameters getRegionOfInterest (MethodID 0103_{16})

Name	Type	Description
regionOfInterest	sRegionOfInterest	Parameter of the requested ROI

Specific error codes: See Table 61.

Table 61 — Error codes getRegionOfInterest (MethodID 0103_{16})

Error code
E_INVALID_ROI_INDEX

7.7.12 getRegionsOfInterest (MethodID 0104_{16})

Purpose: This method returns the parameters for all supported ROIs.

Input parameters: None

Output parameters: See Table 62.

Table 62 — Output parameters getRegionsOfInterest (MethodID 0104_{16})

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

Specific error codes: See Table 63.

Table 63 — Error codes getRegionsOfInterest (MethodID 0104_{16})

Error code
E_INVALID_ROI_INDEX
E_INVALID_ROI_NUMBER

7.7.13 eraseRegionOfInterest (MethodID 0109_{16})

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

Input parameters: See Table 64.

Table 64 — Input parameters eraseRegionOfInterest (MethodID 0109_{16})

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: See Table 65.

Table 65 — Error codes eraseRegionOfInterest (MethodID 0109_{16})

Error code
E_INVALID_ROI_INDEX

7.7.14 setVideoFormat (MethodID 0111_{16})

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

A client shall lock the camera using the setCamExclusive method before requesting this method.

Input parameters: See Table 66.

Table 66 — Input parameters setVideoFormat (MethodID 0111_{16})

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: See Table 67.

Table 67 — Error codes setVideoFormat (MethodID 0111_{16})

Error code
E_INVALID_ROI_INDEX
E_INVALID_VIDEO_FORMAT
E_LOCKED_BY_FOREIGN_INSTANCE
E_LOCK_EXPIRED

7.7.15 getVideoFormat (MethodID 0113_{16})

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

Input parameters: See Table 68.

Table 68 — Input parameters getVideoFormat (MethodID 0113_{16})

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

Output parameters: See Table 69.

Table 69 — Output parameters getVideoFormat (MethodID 0113_{16})

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

Specific error codes: See Table 70.

Table 70 — Error codes getVideoFormat (MethodID 0113_{16})

Error code
E_INVALID_ROI_INDEX

7.7.16 eraseVideoFormat (MethodID 0119_{16})

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

A client shall lock the camera using the setCamExclusive method before requesting this method.

Input parameters: See Table 71.

Table 71 — Input parameters eraseVideoFormat (MethodID 0119_{16})

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: See Table 72.

Table 72 — Error codes eraseVideoFormat (MethodID 0119_{16})

Error code
E_INVALID_ROI_INDEX
E_LOCKED_BY_FOREIGN_INSTANCE

Table 72 *(continued)*

Error code
E_LOCK_EXPIRED

7.7.17 setHistogramFormat (MethodID 0121_{16})

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

A client shall lock the camera using the setCamExclusive method before requesting this method.

Input parameters: See Table 73.

Table 73 — Input parameters setHistogramFormat (MethodID 0121_{16})

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

Specific error codes: See Table 74.

Table 74 — Error codes setHistogramFormat (MethodID 0121_{16})

Error code
E_INVALID_ROI_INDEX
E_INVALID_HISTOGRAM_FORMAT
E_LOCKED_BY_FOREIGN_INSTANCE
E_LOCK_EXPIRED

7.7.18 getHistogramFormat (MethodID 0123_{16})

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

Input parameters: See Table 75.

Table 75 — Input parameters getHistogramFormat (MethodID 0123_{16})

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

Output parameters: See Table 76.

Table 76 — Output parameters getHistogramFormat (MethodID 0123_{16})

Name	Type	Description
Format	sHistogramFormat	Video format addressed by the ROI index

Specific error codes: See Table 77.

Table 77 — Error codes getHistogramFormat (MethodID 0123_{16})

Error code
E_INVALID_ROI_INDEX

7.7.19 eraseHistogramFormat (MethodID 0129_{16})

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

A client shall lock the camera using the setCamExclusive method before requesting this method.

Input parameters: See Table 78.

Table 78 — Input parameters eraseHistogramFormat (MethodID 0129_{16})

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: See Table 79.

Table 79 — Error codes eraseHistogramFormat (MethodID 0129_{16})

Error code
E_INVALID_ROI_INDEX
E_LOCKED_BY_FOREIGN_INSTANCE
E_LOCK_EXPIRED

7.7.20 SubscribeROIVideo (MethodID 0131_{16})

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

Input parameters: See Table 80.

Table 80 — Input parameters SubscribeROIVideo (MethodID 0131_{16})

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 receives a cyclic notification. The cycle is defined by the parameter transmissionCycle. See Table 81.

Table 81 — Output parameters SubscribeROIVideo (MethodID 0131_{16})

Name	Type	Description
ROIContent	sVideoContent	Content of the registered ROI See 7.6.18.

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

Specific error codes: See Table 82.

Table 82 — Error codes SubscribeROIVideo (MethodID 0131_{16})

Error code
E_INVALID_ROI_INDEX
E_INVALID_VIDEO_FORMAT

7.7.21 UnSubscribeROIVideo (MethodID 0132_{16})

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

Input parameters: See Table 83.

Table 83 — Input parameters UnSubscribeROIVideo (MethodID 0132_{16})

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

7.7.22 SubscribeROIHistogram (MethodID 0x0133)

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

Input parameters: See Table 84.

Table 84 — Input parameters SubscribeROIHistogram (MethodID 0133_{16})

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 receives cyclic a package. The cycle is defined by the parameter histogramUpdateCyle. See Table 85.

Table 85 — Output parameters SubscribeROIHistogram (MethodID 0133_{16})

Name	Type	Description
ROIhistogram	sHistogramContent	Histogram of the registered ROI See 7.6.16.

7.7.23 UnSubscribeROIHistogram (MethodID 0x0134)

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

Input parameters: See Table 86.

Table 86 — Input parameters UnSubscribeROIHistogram (MethodID 0134_{16})

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

7.7.24 setCamControl (MethodID 0201_{16})

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

A client shall lock the camera using the setCamExclusive method before requesting this method.

Input parameters: See Table 87.

Table 87 — Input parameters setCamControl (MethodID 0201_{16})

Name	Type	Description
camControlIndex	unit8	Use eControlIndex, see 7.5.5.
camControl	void	Use the type belonging to the camControlIndex, see 7.6.23.

Specific error codes: See Table 88.

Table 88 — Error codes setCamControl (MethodID 0201_{16})

Error code
E_INVALID_CONTROL_INDEX
E_INVALID_CONTROL_MODE
E_INVALID_CONTROL_VALUE
E_LOCKED_BY_FOREIGN_INSTANCE
E_LOCK_EXPIRED

7.7.25 setCamControls (MethodID 0202_{16})

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

A client must lock the camera using the setCamExclusive method before requesting this method.

Input parameters: See Table 89.

Table 89 — Input parameters setCamControls (MethodID 0202_{16})

Name	Type	Description
camControls	sCamControl	

Specific error codes: See Table 90.

Table 90 — Error codes setCamControls (MethodID 0202_{16})

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.

7.7.26 getCamControl (MethodID 0203_{16})

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

Input parameters: See Table 91.