OMRON CORPORATION
ELECTRONIC AND MECHANICAL
COMPONENTS COMPANY

HVC-C2W

API Specification Document

■ Trademarks

“OKAO” and “OKAO Vision” are registered trademarks of OMRON Corporation in Japan.

■ Revision History

Rev	Contents	Prepared by	Reviewed by	Approved by
A	First release	AOB 2015.12.28 T.Inoue	AOB 2015.12.28 S.Manabe	AOB 2015.12.29 O.Matsutake

■ Additional Notes

You may not reproduce, duplicate or copy the contents of this specification document without proper written authorization from OMRON Corporation. The contents of this specification document may change without notice.

TABLE OF CONTENTS

1 Software Specifications

1.1 Data Type Definitions

Data type	Description
HVCW_CHAR	8-bit character
HVCW_UINT8	8-bit unsigned character type
HVCW_BYTE	8-bit unsigned character type
HVCW_BOOL	32-bit signed integer type
HVCW_INT32	32-bit signed integer type
HVCW_UINT32	32-bit unsigned integer type
HVCW_INT16	16-bit signed short integer type
HVCW_UINT16	16-bit unsigned short integer type
HVCW_VOID	void

1.2 Error Code Definitions

Error code	Description	Value
HVCW_SUCCESS	Process successful	1
HVCW_INVALID_PARAM	Invalid parameter	2
HVCW_NOT_READY	Process preparation not done	3
HVCW_BUSY	Process not available now	4
HVCW_NOT_SUPPORT	Requested process not supported	5
HVCW_TIMEOUT	Timeout error	6
HVCW_NOT_FOUND	Process target not found	7
HVCW_FAILURE	Unspecified error	8
HVCW_NOT_INITIALIZE	SDK not initialized	11
HVCW_DISCONNECTED	Camera disconnected	12
HVCW_NOHANDLE	Handle error	20
HVCW_NO_FACE	No face detected	30
HVCW_PLURAL_FACES	Multiple faces detected	31
HVCW_INVALID_RECEIVEDATA	Invalid data transmitted	40
HVCW_NOFILE	File does not exist	50
HVCW_SD_NOT_INSERT	SD card not inserted	61
HVCW_SD_READ	SD card error	62

1.3 Command Status Definitions

Command status	Description
0	Normal end
0xFF	Improper command
0xFE	Improper data length
0xFD	Improper data content
0xFC	Command not supported by selected OKAO mode
0xFB	Erase files and execute upload when the scheduler and event program are enabled
0xDF	File does not exist
0xDE	File could not be erased
0xCF	SD card not inserted
0xCE	SD card is full
0xCD	SD card read error
0xCC	SD card write error
0xBF	FLASH read error
0xBE	FLASH write error
0xAF	Driver error
0x9F	System failure error
1	No face detected
2	Multiple faces detected

1.4 Camera API List

1.5 Camera API Specifications

■ **Connect camera**

```
HVCW_INT32 HVCW_Connect(HHVC hHVC, HVCW_UINT8 *pucCameraId,
                                                HVCW_UINT8 *pucAccessToken)
```

Arguments	Input:	hHVC	HVC handle
		pucCameraId	Camera ID
		pucAccessToken	Access token
Return values	HVCW_SUCCESS		Normal end
	HVCW_NOHANDLE		Handle error
	HVCW_INVALID_PARAM		Parameter error
	HVCW_FAILURE		Unspecified error
Description	Connects to the camera. The access token is obtainable through login in to the system (CGI).		

■ **Disconnect camera**

```
HVCW_INT32 HVCW_Disconnect(HHVC hHVC)
```

Arguments	Input:	hHVC	HVC handle
Return values	HVCW_SUCCESS		Normal end
	HVCW_NOHANDLE		Handle error
	HVCW_INVALID_PARAM		Parameter error
	HVCW_NOT_INITIALIZE		Initialization error
	HVCW_FAILURE		Unspecified error
Description	Disconnects the camera.		

■ **Get camera firmware version**

```
HVCW_INT32 HVCW_GetCameraVersion(HHVC hHVC, HVCW_UINT8 aucVersion[128])
```

Arguments	Input:	hHVC	HVC handle
	Output:	aucVersion	Version
Return values	HVCW_SUCCESS		Normal end
	HVCW_NOHANDLE		Handle error
	HVCW_INVALID_PARAM		Parameter error
Description	Gets the camera firmware version info.		

■ Update camera firmware

```
HVCW_INT32 HVCW_UpdateFirmware(HHVC hHVC)
```

Arguments	Input: hHVC	HVC handle
Return values	HVCW_SUCCESS	Normal end
	HVCW_NOHANDLE	Handle error
	HVCW_INVALID_PARAM	Parameter error
	HVCW_TIMEOUT	Timeout error
	HVCW_DISCONNECTED	Disconnected
	HVCW_NOT_READY	Firmware up to date
	HVCW_NOT_INITIALIZE	Initialization error
	HVCW_FAILURE	Unspecified error
Description	Updates the camera firmware.	

■ Set camera night vision mode

```
HVCW_INT32 HVCW_SetNightVisionMode( HHVC hHVC, HVCW_NIGHT_VISION_MODE mode)
```

Arguments	Input: hHVC	HVC handle
	mode	Night vision mode
Return values	HVCW_SUCCESS	Normal end
	HVCW_NOHANDLE	Handle error
	HVCW_INVALID_PARAM	Parameter error
	HVCW_TIMEOUT	Timeout error
	HVCW_DISCONNECTED	Disconnected
	HVCW_NOT_INITIALIZE	Initialization error
	HVCW_FAILURE	Unspecified error
Description	Sets the camera night vision mode. Specify HVCW_NightVisionMode_Auto in mode to set to automatic mode. In automatic mode, the IR LED will turn on or off depending on the surrounding light. Specify HVCW_NightVisionMode_Manual in mode to set to manual mode. In manual mode, the IR LED can be set on or by calling HVCW_SetNightVisionStatus().	
Input specifications	mode: HVCW_NightVisionMode_Auto or HVCW_NightVisionMode_Manual	
Default value	mode: HVCW_NightVisionMode_Auto	

■ Get camera night vision mode

```
HVCW_INT32 HVCW_GetNightVisionMode(HHVC hHVC, HVCW_NIGHT_VISION_MODE *pMode)
```

Arguments	Input: hHVC	HVC handle
	Output: pMode	Night vision mode
Return values	HVCW_SUCCESS	Normal end
	HVCW_NOHANDLE	Handle error
	HVCW_INVALID_PARAM	Parameter error
	HVCW_TIMEOUT	Timeout error
	HVCW_DISCONNECTED	Disconnected
	HVCW_NOT_INITIALIZE	Initialization error
	HVCW_FAILURE	Unspecified error
Description	Gets the camera night vision mode set.	

■ **Set camera night vision status**

```
HVCW_INT32 HVCW_SetNightVisionStatus(HHVC hHVC,
                                     HVCW_NIGHT_VISION_STATUS status)
```

Arguments	Input: `hHVC`	HVC handle
	`status`	Night vision status
Return values	`HVCW_SUCCESS`	Normal end
	`HVCW_NOHANDLE`	Handle error
	`HVCW_INVALID_PARAM`	Parameter error
	`HVCW_TIMEOUT`	Timeout error
	`HVCW_DISCONNECTED`	Disconnected
	`HVCW_NOT_INITIALIZE`	Initialization error
	`HVCW_NOT_SUPPORT`	Mode error (night vision mode set to auto)
	`HVCW_FAILURE`	Unspecified error
Description	Sets the camera night vision status. The camera night vision status indicates whether the IR LED is turned on or off. This can only be specified when the camera night vision mode is set to manual mode.	
Input specifications	`status`: `HVCW_NightVisionStatus_Off` or `HVCW_NightVisionStatus_On`	

■ **Get camera night vision status**

```
HVCW_INT32 HVCW_GetNightVisionStatus(HHVC hHVC,
                                     HVCW_NIGHT_VISION_STATUS *pStatus)
```

Arguments	Input: `hHVC`	HVC handle
	Output: `pStatus`	Night vision status
Return values	`HVCW_SUCCESS`	Normal end
	`HVCW_NOHANDLE`	Handle error
	`HVCW_INVALID_PARAM`	Parameter error
	`HVCW_TIMEOUT`	Timeout error
	`HVCW_DISCONNECTED`	Disconnected
	`HVCW_NOT_INITIALIZE`	Initialization error
	`HVCW_FAILURE`	Unspecified error
Description	Gets the camera night vision status set.	

■ **Enable event monitor**

```
HVCW_INT32 HVCW_EnableEventMonitor(HHVC hHVC, HVCW_VOID *pUserParam,
                                   HVCW_EventCallback callbackFunc)
```

Arguments	Input: hHVC	HVC handle
	pUserParam	User parameter
	callbackFunc	Event callback function
Return values	HVCW_SUCCESS	Normal end
	HVCW_NOHANDLE	Handle error
	HVCW_INVALID_PARAM	Parameter error
	HVCW_NOT_INITIALIZE	Initialization error
	HVCW_FAILURE	Unspecified error
Description	Sets the event callback function to enable the event monitor. Refer to HVCW_Event for details on events. Refer to HVCW_EventCallback for details on the event callback function.	

■ **Disable event monitor**

```
HVCW_INT32 HVCW_DisableEventMonitor(HHVC hHVC)
```

Arguments	Input: hHVC	HVC handle
Return values	HVCW_SUCCESS	Normal end
	HVCW_NOHANDLE	Handle error
	HVCW_INVALID_PARAM	Parameter error
	HVCW_NOT_INITIALIZE	Initialization error
	HVCW_FAILURE	Unspecified error
Description	Disables the event monitor.	

■ **Set camera speaker volume**

```
HVCW_INT32 HVCW_SetSpeakerVolume(HHVC hHVC, HVCW_UINT32 unVolume)
```

Arguments	Input: `hHVC`	HVC handle
	`unVolume`	Speaker volume
Return values	`HVCW_SUCCESS`	Normal end
	`HVCW_NOHANDLE`	Handle error
	`HVCW_INVALID_PARAM`	Parameter error
	`HVCW_TIMEOUT`	Timeout error
	`HVCW_DISCONNECTED`	Disconnected
	`HVCW_NOT_INITIALIZE`	Initialization error
	`HVCW_FAILURE`	Unspecified error
Description	Sets the camera's speaker volume.	
Input specifications	`unVolume`: 0 to 100	
Default value	`unVolume`: 86	

■ **Get camera speaker volume**

```
HVCW_INT32 HVCW_GetSpeakerVolume(HHVC hHVC, HVCW_UINT32 *punVolume)
```

Arguments	Input: `hHVC`	HVC handle
	Output: `punVolume`	Speaker volume
Return values	`HVCW_SUCCESS`	Normal end
	`HVCW_NOHANDLE`	Handle error
	`HVCW_INVALID_PARAM`	Parameter error
	`HVCW_TIMEOUT`	Timeout error
	`HVCW_DISCONNECTED`	Disconnected
	`HVCW_NOT_INITIALIZE`	Initialization error
	`HVCW_FAILURE`	Unspecified error
Description	Gets the camera's speaker volume.	

■ Set camera mic sensitivity

```
HVCW_INT32 HVCW_SetMicSensitivity(HHVC hHVC,HVCW_UINT32 unSensitivity)
```

Arguments	Input: `hHVC`	HVC handle
	`unSensitivity`	Mic sensitivity
Return values	`HVCW_SUCCESS`	Normal end
	`HVCW_NOHANDLE`	Handle error
	`HVCW_INVALID_PARAM`	Parameter error
	`HVCW_TIMEOUT`	Timeout error
	`HVCW_DISCONNECTED`	Disconnected
	`HVCW_NOT_INITIALIZE`	Initialization error
	`HVCW_FAILURE`	Unspecified error
Description	Sets the camera mic sensitivity. A higher value indicates a higher sensitivity.	
Input specifications	`unSensitivity`: 0 to 9	
Default value	`unSensitivity`: 9	

■ Get camera mic sensitivity

```
HVCW_INT32 HVCW_GetMicSensitivity(HHVC hHVC,HVCW_UINT32 *punSensitivity)
```

Arguments	Input: `hHVC`	HVC handle
	`punSensitivity`	Mic sensitivity
Return values	`HVCW_SUCCESS`	Normal end
	`HVCW_NOHANDLE`	Handle error
	`HVCW_INVALID_PARAM`	Parameter error
	`HVCW_TIMEOUT`	Timeout error
	`HVCW_DISCONNECTED`	Disconnected
	`HVCW_NOT_INITIALIZE`	Initialization error
	`HVCW_FAILURE`	Unspecified error
Description	Gets the camera mic sensitivity set.	

■ Get Wi-Fi RSSI value

```
HVCW_INT32 HVCW_GetWiFiRSSI(HHVC hHVC, HVCW_INT32 *pnWifiRssi)
```

Arguments	Input: `hHVC`	HVC handle
	Output: `pnWifiRssi`	Camera Wi-Fi RSSI value
Return values	`HVCW_SUCCESS`	Normal end
	`HVCW_NOHANDLE`	Handle error
	`HVCW_INVALID_PARAM`	Parameter error
	`HVCW_TIMEOUT`	Timeout error
	`HVCW_DISCONNECTED`	Disconnected
	`HVCW_NOT_INITIALIZE`	Initialization error
	`HVCW_FAILURE`	Unspecified error
Description	Gets the camera Wi-Fi RSSI value. A larger RSSI value indicates a stronger receiving signal.	

■ Create data sound file

```
HVCW_INT32 HVCW_GenerateDataSoundFile(HVCW_UINT8 *pucTargetFile,
 HVCW_UINT8 *pucSSID, HVCW_UINT8 *pucPassword, HVCW_UINT8 *pusAccessToken)
```

Arguments	Input: `pucTargetFile`	File destination
	`pucSSID`	Network name (SSID)
	`pucPassword`	Password
	`pusAccessToken`	Access token
Return values	`HVCW_SUCCESS`	Normal end
	`HVCW_INVALID_PARAM`	Parameter error
	`HVCW_NOT_INITIALIZE`	Initialization error
	`HVCW_FAILURE`	Unspecified error
Description	Creates a data sound file for the network settings. Erase any created sound file on the application side. Information required for the network settings will include the network name (SSID), password and access token. The access token is obtainable through login in to the system (CGI). Data sound file details	

Item	Set value
Sampling rate	8000
Number of channels	1
Audio format	Signed 16 bit PCM
Byte order	Little Endian

Make sure to specify the full path when designating the file save destination.
Make sure to make the file save destination writeable.
Any already existing file will be overwritten by the newly created file when saved.

■ **Format storage**

```
HVCW_INT32 HVCW_RequestStorageFormat(HHVC hHVC,
                                     HVCW_STORAGE_FORMAT_RESULT_CODE *pResultCode)
```

Arguments	Input: `hHVC` Output: `pResultCode`	HVC handle Format results details

Return values	`HVCW_SUCCESS`
	`HVCW_NOHANDLE`
	`HVCW_INVALID_PARAM`
	`HVCW_TIMEOUT`
	`HVCW_DISCONNECTED`
	`HVCW_NOT_INITIALIZE`
	`HVCW_FAILURE`
		Normal end
		Handle error
		Parameter error
		Timeout error
		Disconnected
		Initialization error
		Unspecified error

Description	Formats the storage. Note that all the data stored will be erased. The settings stored on the Flash device will not be affected. Refer to `HVCW_StorageFormatResultCode()` for details on the format results.	

■ **Get connection type**

```
HVCW_INT32 HVCW_GetConnectionType(HHVC hHVC,
                                  HVCW_CONNECTION_TYPE *pConnType)
```

Arguments	Input: `hHVC` Output: `pConnType`	HVC handle Connection type

Return values	`HVCW_SUCCESS`
	`HVCW_NOHANDLE`
	`HVCW_INVALID_PARAM`
	`HVCW_TIMEOUT`
	`HVCW_DISCONNECTED`
	`HVCW_NOT_INITIALIZE`
	`HVCW_FAILURE`
		Normal end
		Handle error
		Parameter error
		Timeout error
		Disconnected
		Initialization error
		Unspecified error

Description	Gets the camera connection type. The connection type can be a P2P connection, relay server connection or local LAN connection.	

■ **Start live streaming**

```
HVCW_INT32 HVCW_StartLive(HHVC hHVC, HVCW_VOID *pUserParam,
                                    HVCW_VIDEO_RESOLUTION videoResolution,
                                HVCW_RequestRenderingCallback renderingCallback,
                                    HVCW_LiveEventCallback eventCallback)
```

Arguments	Input:	`hHVC`	HVC handle
		`pUserParam`	User parameter
		`videoResolution`	Resolution
		`renderingCallback`	Video frame
		`eventCallback`	Live event callback function
Return values	`HVCW_SUCCESS`		Normal end
	`HVCW_NOHANDLE`		Handle error
	`HVCW_INVALID_PARAM`		Parameter error
	`HVCW_TIMEOUT`		Timeout error
	`HVCW_DISCONNECTED`		Disconnected
	`HVCW_NOT_INITIALIZE`		Initialization error
	`HVCW_BUSY`		Process not available now
	`HVCW_FAILURE`		Unspecified error
Description	Starts live streaming. Live streaming with multiple cameras is possible but it may result in a reduction of the framerate due to the process load on the device. In such case, change the resolution on the application side. A notification will be sent through callback for the decoded video frame and sound data. The decoded data must be displayed or played on the application side. Refer to `HVCW_RequestRenderingCallback` for details on the video frame and sound data. The video frame and sound data will be synchronized at the notification time. The live streaming resolution may sometimes change automatically due to the network conditions or the device process speed. A notification will be sent through callback for a live event. Refer to `HVCW_EventCallback` for details. The API may not start with the selected resolution if other users are already live streaming on the low resolution mode. As such, confirm the contents of the received callback before rendering the video frame. Make sure to call `HVCW_FreeDecodedVideoBuffer()` and `HVCW_FreeDecodedAudioBuffer()` from the application side to free the notified video frame and the sound data. Make sure to call `HVCW_StopLive()` in order to stop the live streaming even if a live event error or disconnection notification was received.		
Input specifications	`videoResolution`: `HVCW_VideoResolution_High` (1280×720) or `HVCW_VideoResolution_Middle` (640×360) or `HVCW_VideoResolution_Low` (320×180)		

■ Stop live streaming

```
HVCW_INT32 HVCW_StopLive(HHVC hHVC)
```

Argument	Input: hHVC	HVC handle
Return values	HVCW_SUCCESS	Normal end
	HVCW_NOHANDLE	Handle error
	HVCW_INVALID_PARAM	Parameter error
	HVCW_TIMEOUT	Timeout error
	HVCW_DISCONNECTED	Disconnected
	HVCW_NOT_INITIALIZE	Initialization error
	HVCW_NOT_READY	Live streaming not started
	HVCW_FAILURE	Unspecified error
Description	Stops live streaming.	

■ Free video frame buffer

```
HVCW_INT32 HVCW_FreeDecodedVideoBuffer(HHVC hHVC, const HVCW_VOID *pBuffer)
```

Arguments	Input: hHVC	HVC handle
	pBuffer	Video frame buffer
Return values	HVCW_SUCCESS	Normal end
	HVCW_NOHANDLE	Handle error
	HVCW_INVALID_PARAM	Parameter error
Description	Frees the video frame buffer. Make sure to free the received video frame buffer by calling this function. Not freeing it may cause a memory leak.	

■ Free sound data buffer

```
HVCW_INT32 HVCW_FreeDecodedAudioBuffer(HHVC hHVC, const HVCW_VOID *pBuffer)
```

Arguments	Input: hHVC	HVC handle
	pBuffer	Sound data buffer
Return values	HVCW_SUCCESS	Normal end
	HVCW_NOHANDLE	Handle error
	HVCW_INVALID_PARAM	Parameter error
Description	Frees the sound data buffer. Make sure to free the received sound data buffer by calling this function. Not freeing it may cause a memory leak.	

■ Enter talk mode

```
HVCW_INT32 HVCW_EnterTalkMode(HHVC hHVC)
```

Argument	Input: hHVC	HVC handle
Return values	HVCW_SUCCESS	Normal end
	HVCW_NOHANDLE	Handle error
	HVCW_INVALID_PARAM	Parameter error
	HVCW_TIMEOUT	Timeout error
	HVCW_DISCONNECTED	Disconnected
	HVCW_NOT_INITIALIZE	Initialization error
	HVCW_NOT_READY	Live streaming not started
	HVCW_FAILURE	Unspecified error
Description	Enters talk mode. Talk mode is used during live streaming to play from the camera side sound data input from the device's mic. This function is used to prevent loopback of played sound on the camera side. Make sure to call it when entering talk mode. Entering talk mode can only be done while live streaming. Make sure to mute the live streaming sound output, i.e. not play the live streaming sound from the camera, before entering talk mode.	

■ Exit talk mode

```
HVCW_INT32 HVCW_ExitTalkMode(HHVC hHVC)
```

Argument	Input: hHVC	HVC handle
Return values	HVCW_SUCCESS	Normal end
	HVCW_NOHANDLE	Handle error
	HVCW_INVALID_PARAM	Parameter error
	HVCW_TIMEOUT	Timeout error
	HVCW_DISCONNECTED	Disconnected
	HVCW_NOT_INITIALIZE	Initialization error
	HVCW_NOT_READY	Live streaming not started
	HVCW_FAILURE	Unspecified error
Description	Exits talk mode.	

■ Transfer sound data

```
HVCW_INT32 HVCW_TransferSoundData(HHVC hHVC, HVCW_BYTE *pucSoundData,
                                  HVCW_UINT32 unSoundLen)
```

<table>
<tr><td>Arguments</td><td>Input: hHVC — HVC handle
pucSoundData — Sound data to transfer
unSoundLen — Sound data length</td></tr>
<tr><td>Return values</td><td>HVCW_SUCCESS — Normal end
HVCW_NOHANDLE — Handle error
HVCW_INVALID_PARAM — Parameter error
HVCW_TIMEOUT — Timeout error
HVCW_DISCONNECTED — Disconnected
HVCW_NOT_INITIALIZE — Initialization error
HVCW_NOT_READY — Live streaming not started
HVCW_FAILURE — Unspecified error</td></tr>
<tr><td>Description</td><td>Transfers the sound input from the device’s mic to the camera to play it.

Make sure to enter talk mode before calling this function. It is required to catch the mic sound data and transfer it quickly with this function on the application side in order to play the sound continuously. The sound data transfer is only possible during live streaming. Transferring the sound data requires units of the length of 1024 bytes. Use zero suppression for padding any data length shorter than 1024 bytes.

Transferable sound data format
<table>
<tr><th>Item</th><th>Value</th></tr>
<tr><td>Sampling rate</td><td>8000</td></tr>
<tr><td>Number of channels</td><td>1</td></tr>
<tr><td>Audio format</td><td>Signed 16 bit PCM</td></tr>
<tr><td>Byte order</td><td>Little Endian</td></tr>
</table>
</td></tr>
<tr><td>Input specifications</td><td>unSoundLen: 1024 (fixed)</td></tr>
</table>

■ **Set live streaming resolution**

```
HVCW_INT32 HVCW_SetVideoResolution(HHVC hHVC,
                                   HVCW_VIDEO_RESOLUTION videoResolution)
```

Arguments	Input: `hHVC` `videoResolution`	HVC handle Resolution

Return values	`HVCW_SUCCESS`
	`HVCW_NOHANDLE`
	`HVCW_INVALID_PARAM`
	`HVCW_TIMEOUT`
	`HVCW_DISCONNECTED`
	`HVCW_NOT_INITIALIZE`
	`HVCW_NOT_READY`
	`HVCW_NOT_SUPPORT`
	`HVCW_FAILURE`
		Normal end
		Handle error
		Parameter error
		Timeout error
		Disconnected
		Initialization error
		Live streaming not started
		Specified resolution not supported
		Unspecified error

Description	Sets the live streaming resolution. The resolution can only be set while live streaming. Specify the resolution at the start of the live streaming by calling `HVCW_StartLive()`. It will sometimes not be possible to change the resolution to the specified value if different live streaming users are using different resolution modes. The resolution can be changed without interrupting the live streaming.	
Input specifications	`videoResolution`: `HVCW_VideoResolution_High` (1280×720) or `HVCW_VideoResolution_Middle` (640×360) or `HVCW_VideoResolution_Low` (320×180)	

■ **Get live streaming resolution**

```
HVCW_INT32 HVCW_GetVideoResolution(HHVC hHVC,
                                   HVCW_VIDEO_RESOLUTION *pVideoResolution)
```

Arguments	Input: `hHVC` Output: `pVideoResolution`	HVC handle Resolution

Return values	`HVCW_SUCCESS`
	`HVCW_NOHANDLE`
	`HVCW_INVALID_PARAM`
	`HVCW_TIMEOUT`
	`HVCW_DISCONNECTED`
	`HVCW_NOT_INITIALIZE`
	`HVCW_NOT_READY`
	`HVCW_FAILURE`
		Normal end
		Handle error
		Parameter error
		Timeout error
		Disconnected
		Initialization error
		Live streaming not started
		Unspecified error

Description	Gets the live streaming resolution. The resolution can only be obtained while live streaming. The resolution info can also be obtained by referring to the width and height included in the rendering request callback function. Notifications will be sent when the resolution changes during live events.	

■ Enable sound detection

```
HVCW_INT32 HVCW_EnableSoundDetection(HHVC hHVC, HVCW_UINT32 unSensitivity)
```

Arguments	Input: hHVC	HVC handle
	unSensitivity	Sound sensitivity
Return values	HVCW_SUCCESS	Normal end
	HVCW_NOHANDLE	Handle error
	HVCW_INVALID_PARAM	Parameter error
	HVCW_TIMEOUT	Timeout error
	HVCW_DISCONNECTED	Disconnected
	HVCW_NOT_INITIALIZE	Initialization error
	HVCW_FAILURE	Unspecified error
Description	Enables sound detection. A higher value indicates a higher sound sensitivity.	
Input specifications	unSensitivity: 0 to 4	

■ Disable sound detection

```
HVCW_INT32 HVCW_DisableSoundDetection(HHVC hHVC)
```

Arguments	Input: hHVC	HVC handle
Return values	HVCW_SUCCESS	Normal end
	HVCW_NOHANDLE	Handle error
	HVCW_INVALID_PARAM	Parameter error
	HVCW_TIMEOUT	Timeout error
	HVCW_DISCONNECTED	Disconnected
	HVCW_NOT_INITIALIZE	Initialization error
	HVCW_FAILURE	Unspecified error
Description	Disables sound detection.	

■ Get sound detection status

```
HVCW_INT32 HVCW_GetSoundDetection(HHVC hHVC, HVCW_BOOL *pbOn,
                                  HVCW_UINT32 *punSensitivity)
```

Arguments	Input: hHVC	HVC handle
	Output: pbOn	Sound status
	punSensitivity	Sound sensitivity
Return values	HVCW_SUCCESS	Normal end
	HVCW_NOHANDLE	Handle error
	HVCW_INVALID_PARAM	Parameter error
	HVCW_TIMEOUT	Timeout error
	HVCW_DISCONNECTED	Disconnected
	HVCW_NOT_INITIALIZE	Initialization error
	HVCW_FAILURE	Unspecified error
Description	Gets the sound detection status (enabled or disabled) and sensitivity.	

■ Enable motion detection

```
HVCW_INT32 HVCW_EnableMotionDetection(HHVC hHVC,
                                      HVCW_UINT32 unDetectionParamsCount,
                                      const HVCW_DETECTIONPARAM aDetectionParams[10])
```

Arguments	Input: hHVC	HVC handle
	unDetectionParamsCount	Motion detection rectangle count
	aDetectionParams	Motion detection parameters
Return values	HVCW_SUCCESS	Normal end
	HVCW_NOHANDLE	Handle error
	HVCW_INVALID_PARAM	Parameter error
	HVCW_TIMEOUT	Timeout error
	HVCW_DISCONNECTED	Disconnected
	HVCW_NOT_INITIALIZE	Initialization error
	HVCW_FAILURE	Unspecified error
Description	Enables motion detection. Up to 10 rectangle areas for motion detection can be set. For example, setting unDetectionParamsCount to 2 will enable two rectangle areas, aDetectionParams[0] and aDetectionParams[1], while ignoring the rest. Rectangle areas outside of the image frame will be ignored. Coordinates are indicated in multiples of 16. Increasing the aDetectionParams.sensitivity value will increase the motion detection sensitivity.	
Input specifications	unDetectionParamsCount: 0 to 10 aDetectionParams.rect.nX: 0 to 1919 aDetectionParams.rect.nY: 0 to 1079 aDetectionParams.rect.nWidth: 1 to 1920 aDetectionParams.rect.nHeight: 1 to 1080 aDetectionParams.sensitivity: 0 to 4	

■ Disable motion detection

```
HVCW_INT32 HVCW_DisableMotionDetection(HHVC hHVC)
```

Arguments	Input: hHVC	HVC handle
Return values	HVCW_SUCCESS	Normal end
	HVCW_NOHANDLE	Handle error
	HVCW_INVALID_PARAM	Parameter error
	HVCW_TIMEOUT	Timeout error
	HVCW_DISCONNECTED	Disconnected
	HVCW_NOT_INITIALIZE	Initialization error
	HVCW_FAILURE	Unspecified error
Description	Disables motion detection.	

■ **Get motion detection status**

```
HVCW_INT32 HVCW_GetMotionDetection(HHVC hHVC, HVCW_BOOL *pbOn,
                                   HVCW_UINT32 *punDetectionParamCount,
                                   HVCW_DETECTIONPARAM aDetectionParams[10])
```

Arguments	Input: hHVC	HVC handle
	Output: pbOn	Status (enabled or disabled)
	Input: punDetectionParamCount	Motion detection rectangle count
	Output: aDetectionParams	Motion detection parameters
Return values	HVCW_SUCCESS	Normal end
	HVCW_NOHANDLE	Handle error
	HVCW_INVALID_PARAM	Parameter error
	HVCW_TIMEOUT	Timeout error
	HVCW_DISCONNECTED	Disconnected
	HVCW_NOT_INITIALIZE	Initialization error
	HVCW_FAILURE	Unspecified error
Description	Gets the motion detection status (enabled or disabled) and the sensitivity set.	

■ **Get camera MAC address**

```
HVCW_INT32 HVCW_GetCameraMacAddress(HHVC hHVC, HVCW_UINT8 aucMACAddress[32])
```

Arguments	Input: hHVC	HVC handle
	Output: aucMACAddress	MAC address
Return values	HVCW_SUCCESS	Normal end
	HVCW_NOHANDLE	Handle error
	HVCW_INVALID_PARAM	Parameter error
Description	Gets the camera's MAC address.	

■ **Check for new camera firmware**

```
HVCW_INT32 HVCW_CheckNewFirmware(HHVC hHVC, HVCW_UINT8 aucVersion[128])
```

Arguments	Input: hHVC	HVC handle
	Output: aucVersion	Latest version info
Return values	HVCW_SUCCESS	Normal end
	HVCW_NOHANDLE	Handle error
	HVCW_INVALID_PARAM	Parameter error
Description	Checks for firmware updates. The version info of the latest available firmware update will be output if the firmware is not up to date. The return value for aucVersion will be NULL if the firmware is up to date.	

■ Get storage info

```
HVCW_INT32 HVCW_GetStorageInfo(HHVC hHVC, HVCW_STORAGEINFO *pStorageInfo)
```

Arguments	Input: `hHVC` Output: `pStorageInfo`	HVC handle Storage info

Return values	`HVCW_SUCCESS`
	`HVCW_NOHANDLE`
	`HVCW_INVALID_PARAM`
	`HVCW_TIMEOUT`
	`HVCW_DISCONNECTED`
	`HVCW_NOT_INITIALIZE`
	`HVCW_FAILURE`
		Normal end
		Handle error
		Parameter error
		Timeout error
		Disconnected
		Initialization error
		Unspecified error

Description	Gets the storage info.	

■ Check for fast download support

```
HVCW_INT32 HVCW_IsSupportDownloadFileFast(HHVC hHVC,
                                          HVCW_BOOL *pbDownloadFileFast)
```

Arguments	Input: `hHVC` Output: `pbDownloadFileFast`	HVC handle Fast download support info

Return values	`HVCW_SUCCESS`
	`HVCW_NOHANDLE`
	`HVCW_INVALID_PARAM`
		Normal end
		Handle error
		Parameter error
		- `NULL` pointer argument

Description	Checks if fast download is supported. The return value will be `TRUE` if fast file download is supported and `FALSE` otherwise. The high speed file download function `HVCW_DownloadFileFast()` cannot be used if fast file download is not supported.	

■ **Download file (fast)**

```
HVCW_INT32 HVCW_DownloadFile_Fast(HHVC hHVC, HVCW_FILE_EXT fileExt,
                const HVCW_CHAR *pcFileName, HVCW_UINT32 unFileNameLength,
                                HVCW_INT32 *pnSize, HVCW_UINT8 **pucBuffer)
```

Arguments	Input:	hHVC	HVC handle
		fileExt	File extension
		pcFileName	File name
		unFileNameLength	File name length
	Output:	pnSize	File size
		pucBuffer	File data storing buffer
Return values	HVCW_SUCCESS		Normal end
	HVCW_NOHANDLE		Handle error - improper handle content
	HVCW_INVALID_PARAM		Parameter error - NULL pointer argument - set value out of specifications
	HVCW_TIMEOUT		Timeout error
	HVCW_DISCONNECTED		Camera disconnected
	HVCW_BUSY		Process not available now
	HVCW_NOT_INITIALIZE		Initialization error
	HVCW_NOFILE		Targeted file does not exist
	HVCW_SD_NOT_INSERT		SD card not inserted
	HVCW_SD_READ		SD card read error
	HVCW_NOT_SUPPORT		Fast download not supported
	HVCW_FAILURE		Unspecified error
Description	Downloads a file stored in the camera SD card. Specify the extension, name and length of the file to be downloaded. The downloaded file will be stored in pucBuffer. An HVCW_BUSY error will be output if the download is already under way or if the camera is unable to process (during a firmware update, etc.). Make sure to call HVCW_FreeFileBuffer() to free the downloaded data buffer. Not freeing it may cause a memory leak.		
Input specifications	HVCW_FILE_EXT: HVCW_FileExt_Log (log file) or HVCW_FileExt_MessageText (message text file) or HVCW_FileExt_Sound (sound file) or HVCW_FileExt_JpgImage (image file) or HVCW_FileExt_ThumbnailJpgImage (thumbnail image file) unFileNameLength: 5 to 60		

■ **Free file data buffer**

```
HVCW_INT32 HVCW_FreeFileBuffer(HHVC hHVC, const HVCW_UINT8 *pucBuffer)
```

Arguments	Input:	hHVC	HVC handle
		pucBuffer	File data storage buffer
Return values	HVCW_SUCCESS		Normal end
	HVCW_NOHANDLE		Handle error- improper handle content
	HVCW_INVALID_PARAM		Parameter error - NULL pointer argument
	HVCW_FAILURE		Unspecified error
Description	Frees the buffer storing the saved file data. Make sure to free the file data buffer obtained with HVCW_DownloadFileFast() by calling this function. Not freeing it may cause a memory leak.		

1.6 OKAO API list

1.7 OKAO API Specifications

■ **Get version**

```
HVCW_INT32 HVCW_GetVersion(HVCW_UINT8 *pucMajor, HVCW_UINT8 *pucMinor,
                                                 HVCW_UINT8 *pucRelease)
```

Arguments	Output: `pucMajor`	Major version
	`pucMinor`	Minor version
	`pucRelease`	Release version
Return values	`HVCW_SUCCESS`	Normal end
	`HVCW_INVALID_PARAM`	Parameter error
Description	Gets the SDK's version.	

■ **Create handle**

```
HHVC HVCW_CreateHandle(void)
```

Argument	None	
Return values	not `NULL`	HVC handle
	`NULL`	Failure
Description	Creates the HVC handle.	

■ **Delete handle**

```
HVCW_INT32 HVCW_DeleteHandle(HHVC hHVC)
```

Argument	Input: `hHVC`	HVC handle
Return values	`HVCW_SUCCESS`	Normal end
	`HVCW_NOHANDLE`	Handle error - improper handle content
	`HVCW_FAILURE`	Finalize failure
Description	Deletes the HVC handle.	

■ Set application ID

```
HVCW_INT32 HVCW_SetAppID(HHVC hHVC, HVCW_INT32 nAppID,
                                        HVCW_UINT8 *pucReturnStatus)
```

Arguments	Input: hHVC	HVC handle
	nAppID	Application ID
	Output: pucReturnStatus	Command status
Return values	HVCW_SUCCESS	Normal end
	HVCW_NOHANDLE	Handle error - improper handle content
	HVCW_INVALID_PARAM	Parameter error - NULL pointer argument - set value out of specification
	HVCW_DISCONNECTED	Camera disconnected
	HVCW_TIMEOUT	Timeout error
	HVCW_INVALID_RECEIVEDATA	Invalid data transmitted
	HVCW_NOT_INITIALIZE	Initialization error
	HVCW_FAILURE	Unspecified error
Description	Sets the application ID. The information stored on the camera can be used and managed for each application individually by setting application IDs. This information includes the log, images, event sound data, and notification messages. All other parameters, including the face recognition Album, OKAO setting parameters, event schedule, camera settings, etc. cannot be managed individually for each application Application IDs must be registered as unique ID numbers. Application IDs from 0 to 99 are reserved and not available.	
Input specifications	nAppID: -1 to INT32_MAX(0x7FFFFFFF)	
Default value	nAppID: -1 (not registered)	

■ Get application ID

```
HVCW_INT32 HVCW_GetAppID(HHVC hHVC, HVCW_INT32 *pnAppID,
                                        HVCW_UINT8 *pucReturnStatus)
```

Arguments	Input: hHVC	HVC handle
	Output: pnAppID	Application ID
	pucReturnStatus	Command status
Return values	HVCW_SUCCESS	Normal end
	HVCW_NOHANDLE	Handle error - improper handle content
	HVCW_INVALID_PARAM	Parameter error - NULL pointer argument
	HVCW_DISCONNECTED	Camera disconnected
	HVCW_TIMEOUT	Timeout error
	HVCW_INVALID_RECEIVEDATA	Invalid data transmitted
	HVCW_NOT_INITIALIZE	Initialization error
	HVCW_FAILURE	Unspecified error
Description	Gets the application ID.	

■ Set OKAO mode

```
HVCW_INT32 HVCW_SetOkaoMode(HHVC hHVC, HVCW_BOOL bOkaoMode,
                                        HVCW_UINT8 *pucReturnStatus)
```

Arguments	Input: hHVC	HVC handle
	bOkaoMode	OKAO mode
	Output: pucReturnStatus	Command status
Return values	HVCW_SUCCESS	Normal end
	HVCW_NOHANDLE	Handle error - improper handle content
	HVCW_INVALID_PARAM	Parameter error - NULL pointer argument - set value out of specifications
	HVCW_DISCONNECTED	Camera disconnected
	HVCW_TIMEOUT	Timeout error
	HVCW_INVALID_RECEIVEDATA	Invalid data transmitted
	HVCW_NOT_INITIALIZE	Initialization error
	HVCW_FAILURE	Unspecified error
Description	Sets the OKAO mode. Set bOkaoMode to TRUE to activate the OKAO mode. The live streaming, event programmer and scheduler will be disabled and only the OKAO process will be available. Set to FALSE to deactivate the OKAO mode and access all features.	
Input specifications	bOkaoMode: TRUE or FALSE	
Default value	bOkaoMode: FALSE	

■ Get OKAO mode

```
HVCW_INT32 HVCW_GetOkaoMode(HHVC hHVC, HVCW_BOOL *pbOkaoMode,
                                        HVCW_UINT8 *pucReturnStatus)
```

Arguments	Input: hHVC	HVC handle
	Output: pbOkaoMode	OKAO mode
	pucReturnStatus	Command status
Return values	HVCW_SUCCESS	Normal end
	HVCW_NOHANDLE	Handle error - improper handle content
	HVCW_INVALID_PARAM	Parameter error - NULL pointer argument - set value out of specifications
	HVCW_DISCONNECTED	Camera disconnected
	HVCW_TIMEOUT	Timeout error
	HVCW_INVALID_RECEIVEDATA	Invalid data transmitted
	HVCW_NOT_INITIALIZE	Initialization error
	HVCW_FAILURE	Unspecified error
Description	Gets the OKAO mode settings.	

■ **Get file count**

```
HVCW_INT32 HVCW_GetFileCount(HHVC hHVC, HVCW_FILE_EXT fileExt,
                    HVCW_UINT32 *punFileCount, HVCW_UINT8 *pucReturnStatus)
```

Arguments	Input:	`hHVC`	HVC handle
		`fileExt`	File extension
	Output:	`punFileCount`	File count
		`pucReturnStatus`	Command status
Return values	`HVCW_SUCCESS`		Normal end
	`HVCW_NOHANDLE`		Handle error - improper handle content
	`HVCW_INVALID_PARAM`		Parameter error - `NULL` pointer argument - set value out of specifications
	`HVCW_DISCONNECTED`		Camera disconnected
	`HVCW_TIMEOUT`		Timeout error
	`HVCW_INVALID_RECEIVEDATA`		Invalid data transmitted
	`HVCW_NOT_INITIALIZE`		Initialization error
	`HVCW_FAILURE`		Unspecified error
Description	Gets the number of files saved on the camera ID card. Specify in `fileExt` the file extension of the file count to be obtained and stored in `pnFileCount`.		
Input specifications	`HVCW_FILE_EXT`: `HVCW_FileExt_Log` (log file) or `HVCW_FileExt_MessageText` (message text file) or `HVCW_FileExt_Sound` (sound file) or `HVCW_FileExt_JpgImage` (image file) or `HVCW_FileExt_ThumbnailJpgImage` (thumbnail image file)		

■ **Get file info**

```
HVCW_INT32 HVCW_GetFileInfo(HHVC hHVC, HVCW_FILE_EXT fileExt,
                            HVCW_UINT32 unFileIndex, HVCW_FILEINFO *pFileInfo,
                                                     HVCW_UINT8 *pucReturnStatus)
```

Arguments	Input:	hHVC	HVC handle
		fileExt	File extension
		unFileIndex	File index
	Output:	pFileInfo	File info
		pucReturnStatus	Command status
Return values	HVCW_SUCCESS		Normal end
	HVCW_NOHANDLE		Handle error - improper handle content
	HVCW_INVALID_PARAM		Parameter error - NULL pointer argument - set value out of specifications
	HVCW_DISCONNECTED		Camera disconnected
	HVCW_TIMEOUT		Timeout error
	HVCW_INVALID_RECEIVEDATA		Invalid data transmitted
	HVCW_NOT_INITIALIZE		Initialization error
	HVCW_FAILURE		Unspecified error
Description	Gets the file info (file name and size) of the files saved on the camera SD card. Specify in nFileIndex the file index to the info to be obtained.		
Input specifications	HVCW_FILE_EXT: HVCW_FileExt_Log (log file) or HVCW_FileExt_MessageText (message text file) or HVCW_FileExt_Sound (sound file) or HVCW_FileExt_JpgImage (image file) or HVCW_FileExt_ThumbnailJpgImage (thumbnail image file)		

■ **Download file**

```
HVCW_INT32 HVCW_DownloadFile(HHVC hHVC, const HVCW_FILEINFO *pFileInfo,
                             HVCW_UINT8 *pucBuffer, HVCW_UINT8 *pucReturnStatus)
```

Arguments	Input:	hHVC	HVC handle
		pFileInfo	File info
	Output:	pucBuffer	File data storage buffer
		pucReturnStatus	Command status
Return values	HVCW_SUCCESS		Normal end
	HVCW_NOHANDLE		Handle error - improper handle content
	HVCW_INVALID_PARAM		Parameter error - NULL pointer argument
	HVCW_DISCONNECTED		Camera disconnected
	HVCW_TIMEOUT		Timeout error
	HVCW_INVALID_RECEIVEDATA		Invalid data transmitted
	HVCW_NOT_INITIALIZE		Initialization error
	HVCW_FAILURE		Unspecified error
Description	Downloads the files saved on the camera SD card. Specify the file info in pFileInfo after obtaining it with HVCW_GetFileinfo(). The downloaded files will be stored in pucBuffer.		

■ **Upload file**

```
HVCW_INT32 HVCW_UploadFile(HHVC hHVC, HVCW_CHAR acFileName[40],
                           HVCW_INT32 nBufferSize, const HVCW_UINT8 *pucBuffer,
                                                   HVCW_UINT8 *pucReturnStatus)
```

Arguments	Input: hHVC	HVC handle
	acFileName	File name
	nBufferSize	Buffer size
	pucBuffer	File data storage buffer
	Output: pucReturnStatus	Command status
Return values	HVCW_SUCCESS	Normal end
	HVCW_NOHANDLE	Handle error - improper handle content
	HVCW_INVALID_PARAM	Parameter error - NULL pointer argument
	HVCW_DISCONNECTED	Camera disconnected
	HVCW_TIMEOUT	Timeout error
	HVCW_INVALID_RECEIVEDATA	Invalid data transmitted
	HVCW_NOT_INITIALIZE	Initialization error
	HVCW_FAILURE	Unspecified error
Description	Uploads a file to the camera SD card. The available characters for acFileName include alpha-numerals, dots ".", hyphens "-" and underscores "_". Other characters cannot be used. The files available for upload are as follows: log files (log), message text files (txt), sound files (wav) and image files (jpg and yuv). The maximum message text file length available for upload is 127 bytes. File upload will fail when the scheduler or the event program is active. Make sure to make them inactive before calling this function.	

■ **Delete file**

```
HVCW_INT32 HVCW_DeleteFile(HHVC hHVC, const HVCW_FILEINFO *pFileInfo,
                                                   HVCW_UINT8 *pucReturnStatus)
```

Arguments	Input: hHVC	HVC handle
	pFileInfo	File info
	Output: pucReturnStatus	Command status
Return values	HVCW_SUCCESS	Normal end
	HVCW_NOHANDLE	Handle error - improper handle content
	HVCW_INVALID_PARAM	Parameter error - NULL pointer argument
	HVCW_DISCONNECTED	Camera disconnected
	HVCW_TIMEOUT	Timeout error
	HVCW_INVALID_RECEIVEDATA	Invalid data transmitted
	HVCW_NOT_INITIALIZE	Initialization error
	HVCW_FAILURE	Unspecified error
Description	Deletes a file stored in the camera SD card. File deletion will fail when the scheduler or the event program is active. Make sure to make them inactive before calling this function.	

■ **Get latest image size**

```
HVCW_INT32 HVCW_GetLastOkaoImageSize(HHVC hHVC, HVCW_INT32 *pnImgBufSize,
                                     HVCW_UINT8 *pucReturnStatus)
```

Arguments	Input:	hHVC	HVC handle
	Output:	pnImgBufSize	Image storage buffer size
		pucReturnStatus	Command status
Return values	HVCW_SUCCESS		Normal end
	HVCW_NOHANDLE		Handle error - improper handle content
	HVCW_INVALID_PARAM		Parameter error - NULL pointer argument
	HVCW_DISCONNECTED		Camera disconnected
	HVCW_TIMEOUT		Timeout error
	HVCW_INVALID_RECEIVEDATA		Invalid data transmitted
	HVCW_NOT_INITIALIZE		Initialization error
	HVCW_FAILURE		Unspecified error
Description	Gets the buffer size storing the latest image after running an OKAO process. The buffer size output will be the buffer size required to get the latest OKAO image through HVCW_GetLastOkaoImage(). Make sure to call this function in order to get the buffer size info before calling HVCW_GetLastOkaoImage() to get the latest image.		

■ **Get latest image**

```
HVCW_INT32 HVCW_GetLastOkaoImage(HHVC hHVC, HVCW_INT32 nImgBufSize,
                                 HVCW_UINT8 *pucImage, HVCW_UINT8 *pucReturnStatus)
```

Arguments	Input:	hHVC	HVC handle
		nImgBufSize	Image storage buffer size
	Output:	pucImage	Image storage buffer
		pucReturnStatus	Command status
Return values	HVCW_SUCCESS		Normal end
	HVCW_NOHANDLE		Handle error - improper handle content
	HVCW_INVALID_PARAM		Parameter error - NULL pointer argument - improper buffer size specified
	HVCW_DISCONNECTED		Camera disconnected
	HVCW_TIMEOUT		Timeout error
	HVCW_INVALID_RECEIVEDATA		Invalid data transmitted
	HVCW_NOT_INITIALIZE		Initialization error
	HVCW_FAILURE		Unspecified error
Description	Gets the latest image after running an OKAO process. The image format will be in JPEG. Make sure to secure the required amount of memory for the image file before calling this function. The required buffer size is obtained by calling HVCW_GetLastOkaoImageSize(). Make sure to call HVCW_GetLastOkaoImageSize() before calling this function.		
Input specifications	nImgBufSize: > pnImgBuffSize of HVCW_GetLastOkaoImageSize()		

■ **Take picture**

```
HVCW_INT32 HVCW_TakePicture(HHVC hHVC, HVCW_FILEINFO *pFileInfo,
                                        HVCW_UINT8 *pucReturnStatus)
```

Arguments	Input: `hHVC`	HVC handle
	Output: `pFileInfo`	Image file info
	`pucReturnStatus`	Command status
Return values	`HVCW_SUCCESS`	Normal end
	`HVCW_NOHANDLE`	Handle error - improper handle content
	`HVCW_INVALID_PARAM`	Parameter error - `NULL` pointer argument
	`HVCW_DISCONNECTED`	Camera disconnected
	`HVCW_TIMEOUT`	Timeout error
	`HVCW_INVALID_RECEIVEDATA`	Invalid data transmitted
	`HVCW_NOT_INITIALIZE`	Initialization error
	`HVCW_FAILURE`	Unspecified error
Description	Takes a picture. The picture taken will be stored on the camera SD card. Call `HVCW_DownloadFile()` to obtain the saved files.	

■ **Set threshold value**

```
HVCW_INT32 HVCW_OKAO_SetThreshold(HHVC hHVC,
        const HVCW_OKAO_THRESHOLD *pThreshold, HVCW_UINT8 *pucReturnStatus)
```

Arguments	Input:	hHVC	HVC handle
		pThreshold	Threshold value
	Output:	pucReturnStatus	Command status
Return values	HVCW_SUCCESS		Normal end
	HVCW_NOHANDLE		Handle error - improper handle content
	HVCW_INVALID_PARAM		Parameter error - NULL pointer argument - set value out of specifications
	HVCW_DISCONNECTED		Camera disconnected
	HVCW_TIMEOUT		Timeout error
	HVCW_INVALID_RECEIVEDATA		Invalid data transmitted
	HVCW_NOT_INITIALIZE		Initialization error
	HVCW_FAILURE		Unspecified error
Description	Sets the threshold value for human body detection, hand detection, pet detection, face detection and face recognition. For human body detection, hand detection, pet detection and face detection, a higher value will result in less false detections but also in less correct detections. For face recognition, a higher value will result in less false recognitions but also in less correct recognitions. Refer to HVCW_OKAO_THRESHOLD for details on the threshold value struct.		
Input specifications	pThreshold.nBody: 1 to 1000 pThreshold.nHand: 1 to 1000 pThreshold.nPet: 1 to 1000 pThreshold.nFace: 1 to 1000 pThreshold.nRecognition: 0 to 1000		
Default values	pThreshold.nBody: 500 pThreshold.nHand: 500 pThreshold.nPet: 500 pThreshold.nFace: 500 pThreshold.nRecognition: 500		

■ **Get threshold value**

```
HVCW_INT32 HVCW_OKAO_GetThreshold(HHVC hHVC,
            HVCW_OKAO_THRESHOLD *pThreshold, HVCW_UINT8 *pucReturnStatus)
```

Arguments	Input:	hHVC	HVC handle
	Output:	pThreshold	Threshold value
		pucReturnStatus	Command status
Return values	HVCW_SUCCESS		Normal end
	HVCW_NOHANDLE		Handle error- improper handle content
	HVCW_INVALID_PARAM		Parameter error - NULL pointer argument
	HVCW_DISCONNECTED		Camera disconnected
	HVCW_TIMEOUT		Timeout error
	HVCW_INVALID_RECEIVEDATA		Invalid data transmitted
	HVCW_NOT_INITIALIZE		Initialization error
	HVCW_FAILURE		Unspecified error
Description	Gets the threshold value set for human body detection, hand detection, pet detection, face detection and face recognition.		

■ **Set detection size range**

```
HVCW_INT32 HVCW_OKAO_SetSizeRange(HHVC hHVC,
      const HVCW_OKAO_SIZE_RANGE *pSizeRange, HVCW_UINT8 *pucReturnStatus)
```

Arguments	Input:	hHVC	HVC handle
		pSizeRange	Detection size range
	Output:	pucReturnStatus	Command status
Return values	HVCW_SUCCESS		Normal end
	HVCW_NOHANDLE		Handle error - improper handle content
	HVCW_INVALID_PARAM		Parameter error - NULL pointer argument - set value out of specifications
	HVCW_DISCONNECTED		Camera disconnected
	HVCW_TIMEOUT		Timeout error
	HVCW_INVALID_RECEIVEDATA		Invalid data transmitted
	HVCW_NOT_INITIALIZE		Initialization error
	HVCW_FAILURE		Unspecified error
Description	Sets the detection size range for human body detection, hand detection, pet detection and face detection. Increasing the detection range by setting a low minimum detection value (nMin) and a high maximum detection size value (nMax) will help detect objects of numerous sizes but will also increase the processing time. Refer to HVCW_OKAO_SIZE_RANGE for details on the detection size range struct.		
Input specifications	pSizeRange.***.nMin: 20 to 8192 pSizeRange.***.nMax: 20 to 8192 with pSizeRange.***.nMin ≤ pSizeRange.***.nMax		
Default value	pSizeRange.body.nMin: 30 pSizeRange.body.nMax: 8192 pSizeRange.hand.nMin: 40 pSizeRange.hand.nMax: 8192 pSizeRange.pet.nMin : 40 pSizeRange.pet.nMax : 8192 pSizeRange.face.nMin: 64 pSizeRange.face.nMax: 8192		

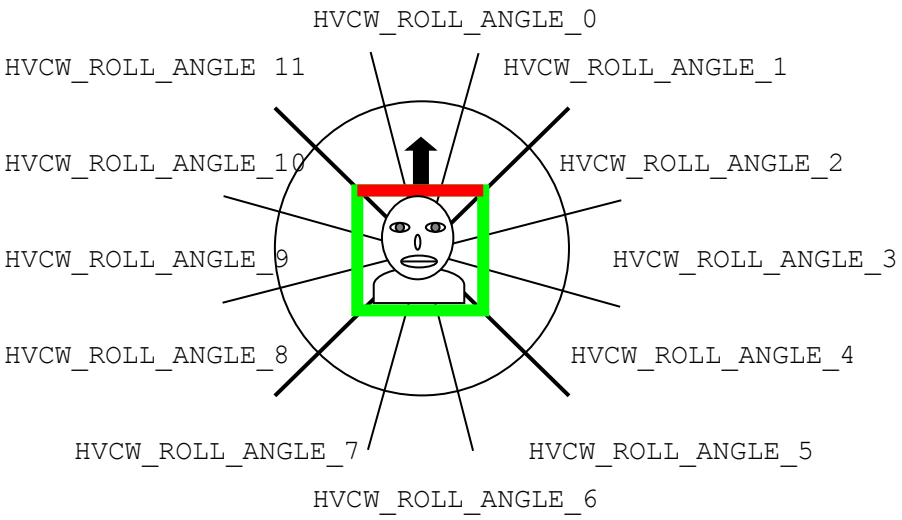
■ **Get detection size range**

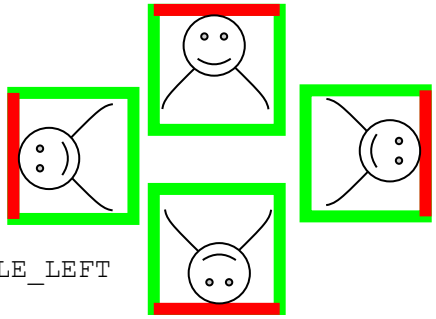
```
HVCW_INT32 HVCW_OKAO_GetSizeRange(HHVC hHVC,
            HVCW_OKAO_SIZE_RANGE *pSizeRange, HVCW_UINT8 *pucReturnStatus)
```

Arguments	Input:	hHVC	HVC handle
	Output:	pSizeRange	Detection size range
		pucReturnStatus	Command status
Return values	HVCW_SUCCESS		Normal end
	HVCW_NOHANDLE		Handle error - improper handle content
	HVCW_INVALID_PARAM		Parameter error - NULL pointer argument
	HVCW_DISCONNECTED		Camera disconnected
	HVCW_TIMEOUT		Timeout error
	HVCW_INVALID_RECEIVEDATA		Invalid data transmitted
	HVCW_NOT_INITIALIZE		Initialization error
	HVCW_FAILURE		Unspecified error
Description	Gets the detection size range set for human body detection, hand detection, pet detection and face detection.		

■ **Set detection angle**

```
HVCW_INT32 HVCW_OKAO_SetDetectionAngle(HHVC hHVC,
                        const HVCW_OKAO_DETECTION_ANGLE *pDetectionAngle,
                                                HVCW_UINT8 *pucReturnStatus)
```

Arguments	Input:	hHVC	HVC handle
		pDetectionAngle	Detection angle
	Output:	pucReturnStatus	Command status
Return values	HVCW_SUCCESS		Normal end
	HVCW_NOHANDLE		Handle error - improper handle content
	HVCW_INVALID_PARAM		Parameter error - NULL pointer argument - set value out of specifications
	HVCW_DISCONNECTED		Camera disconnected
	HVCW_TIMEOUT		Timeout error
	HVCW_INVALID_RECEIVEDATA		Invalid data transmitted
	HVCW_NOT_INITIALIZE		Initialization error
	HVCW_FAILURE		Unspecified error
Description	Sets the detection angle for human body detection, hand detection, pet detection and face detection. **Face Detection** Specify the yaw angle in nPose. Select one of the 3 symbols below. Frontal face: HVCW_POSE_ANGLE_FRONT(0) Frontal and half-profile face: HVCW_POSE_ANGLE_HALF_PROFILE(1) Frontal, half-profile and profile face: HVCW_POSE_ANGLE_PROFILE(2) Specify the roll angle in nAngle. Select one of the 12 symbols below. Multiple detection angles can be specified by using OR (|).  **Pet Detection** Only the roll angle can be set. Specify it in nPet. Select one of the 12 symbols above as for face detection. Multiple detection angles can be specified by using OR (|). **Human Body Detection and Hand Detection** Only the roll angle can be set. Specify it in nBody for human body detection and nHand for hand detection. Select the roll angle from one of the 4 symbols below. Multiple detection angles can be specified by using OR (|).		

	HVCW_ROLL_ANGLE_UP (ROLL_ANGLE_0)  HVCW_ROLL_ANGLE_RIGHT (ROLL_ANGLE_3) HVCW_ROLL_ANGLE_LEFT (ROLL_ANGLE_9) HVCW_ROLL_ANGLE_DOWN (ROLL_ANGLE_6) Refer to HVCW_OKAO_DETECTION_ANGLE for details on the detection angle struct.
Input specifications	**Human Body Detection, Hand Detection, Pet Detection** (pDetectionAngle.***) nBody: see above for details nHand: see above for details nPet: see above for details **Face Detection** (pDetectionAngle.nFace.***) nPose: HVCW_POSE_ANGLE_FRONT or HVCW_POSE_ANGLE_HALF_PROFILE or HVCW_POSE_ANGLE_PROFILE nAngle: see above for details
Default values	pDetectionAngle.nBody: HVCW_ROLL_ANGLE_UP pDetectionAngle.nHand: HVCW_ROLL_ANGLE_UP pDetectionAngle.nPet: HVCW_ROLL_ANGLE_0 pDetectionAngle.nFace.nPose: HVCW_POSE_ANGLE_FRONT pDetectionAngle.nFace.nAngle: HVCW_ROLL_ANGLE_0

■ **Get detection angle**

```
HVCW_INT32 HVCW_OKAO_GetDetectionAngle(HHVC hHVC,
  HVCW_OKAO_DETECTION_ANGLE *pDetectionAngle, HVCW_UINT8 *pucReturnStatus)
```

Arguments	Input: hHVC	HVC handle
	Output: pDetectionAngle	Detection angle
	pucReturnStatus	Command status
Return values	HVCW_SUCCESS	Normal end
	HVCW_NOHANDLE	Handle error - improper handle content
	HVCW_INVALID_PARAM	Parameter error - NULL pointer argument
	HVCW_DISCONNECTED	Camera disconnected
	HVCW_TIMEOUT	Timeout error
	HVCW_INVALID_RECEIVEDATA	Invalid data transmitted
	HVCW_NOT_INITIALIZE	Initialization error
	HVCW_FAILURE	Unspecified error
Description	Gets the detection angle set for human body detection, hand detection, pet detection and face detection.	

■ **Execute OKAO function**

```
HVCW_INT32 HVCW_OKAO_Execute(HHVC hHVC,
                             HVCW_BOOL abUseFunction[HVCW_OkaoFunction_Max],
                    HVCW_OKAO_RESULT *pResult, HVCW_UINT8 *pucReturnStatus)
```

Arguments	Input: `hHVC`	HVC handle
	`abUseFunction`	Function flag
	Output: `pResult`	Result info
	`pucReturnStatus`	Command status
Return values	`HVCW_SUCCESS`	Normal end
	`HVCW_NOHANDLE`	Handle error - improper handle content
	`HVCW_INVALID_PARAM`	Parameter error - `NULL` pointer argument
	`HVCW_DISCONNECTED`	Camera disconnected
	`HVCW_TIMEOUT`	Timeout error
	`HVCW_INVALID_RECEIVEDATA`	Invalid data transmitted
	`HVCW_NOT_INITIALIZE`	Initialization error
	`HVCW_FAILURE`	Unspecified error

Description	Executes the OKAO process.
	
	Specify the function flags of the OKAO functions to be executed by setting them to `TRUE` in `abUseFunction`.
	The function array is as follows.
	
	/* Function index */
	`typedef enum HVCW_OKAO_FUNCTION {`
	` HVCW_OkaoFunction_Body: 0,` /* Human body detection */
	` HVCW_OkaoFunction_Hand,` /* Hand detection */
	` HVCW_OkaoFunction_Pet,` /* Pet detection */
	` HVCW_OkaoFunction_Face,` /* Face detection */
	` HVCW_OkaoFunction_Direction,` /* Face direction estimation */
	` HVCW_OkaoFunction_Age,` /* Age estimation */
	` HVCW_OkaoFunction_Gender,` /* Gender estimation */
	` HVCW_OkaoFunction_Gaze,` /* Gaze estimation */
	` HVCW_OkaoFunction_Blink,` /* Blink estimation */
	` HVCW_OkaoFunction_Expression,` /* Expression estimation */
	` HVCW_OkaoFunction_Recognition,` /* Face recognition */
	` HVCW_OkaoFunction_Max`
	`};`
	
	Refer to `HVCW_OKAO_RESULT` for details on the OKAO process result struct.	

Input specifications	`abUseFunction`: `TRUE` or `FALSE`	
Default value	`abUseFunction`: `FALSE(0)` (for all)	

■ **Register Album**

```
HVCW_INT32 HVCW_ALBUM_Register(HHVC hHVC, HVCW_INT32 nUserID,
HVCW_INT32 nDataID, HVCW_OKAO_RESULT_DETECTION *pFaceResult,
                    HVCW_FILEINFO *pFileInfo, HVCW_UINT8 *pucReturnStatus)
```

Arguments	Input:	hHVC	HVC handle
		nUserID	User ID
		nDataID	Data ID
	Output:	pFaceResult	Face detection result info
		pFileInfo	Registered file info
		pucReturnStatus	Command status
Return values	HVCW_SUCCESS		Normal end
	HVCW_NO_FACE		No face detected
	HVCW_PLURAL_FACES		Multiple faces detected
	HVCW_NOHANDLE		Handle error- improper handle content
	HVCW_INVALID_PARAM		Parameter error - NULL pointer argument - set value out of specifications
	HVCW_DISCONNECTED		Camera disconnected
	HVCW_TIMEOUT		Timeout error
	HVCW_INVALID_RECEIVEDATA		Invalid data transmitted
	HVCW_NOT_INITIALIZE		Initialization error
	HVCW_FAILURE		Unspecified error
Description	Registers a face in the Album. Any face already registered will be overwritten. To register additional Album data for the same user, use the same user ID but a different data ID. Registration is not possible if no face was detected or if multiple faces were detected. Refer to HVCW_OKAO_RESULT_DETECTION for details on the face detection result struct.		
Input specifications	nUserID: 0 to 499 nDataID: 0 to 9		

■ **Change Album user name**

```
HVCW_INT32 HVCW_ALBUM_SetUserName(HHVC hHVC, HVCW_INT32 nUserID,
                    const HVCW_CHAR acName[44], HVCW_UINT8 *pucReturnStatus)
```

Arguments	Input: hHVC	HVC handle
	nUserID	User ID
	acName	User name
	Output: pucReturnStatus	Command status
Return values	HVCW_SUCCESS	Normal end
	HVCW_NOHANDLE	Handle error - improper handle content
	HVCW_INVALID_PARAM	Parameter error - NULL pointer argument - set value out of specifications
	HVCW_DISCONNECTED	Camera disconnected
	HVCW_TIMEOUT	Timeout error
	HVCW_INVALID_RECEIVEDATA	Invalid data transmitted
	HVCW_NOT_INITIALIZE	Initialization error
	HVCW_FAILURE	Unspecified error
Description	Changes a user's name in the Album. The user name of an ID with no data registered in the Album can also be changed.	
Input specifications	nUserID: 0 to 499	
Default value	acName: ""	

■ **Get Album user name**

```
HVCW_INT32 HVCW_ALBUM_GetUserName(HHVC hHVC, HVCW_INT32 nUserID,
                        HVCW_CHAR acName[44], HVCW_UINT8 *pucReturnStatus)
```

Arguments	Input: hHVC	HVC handle
	nUserID	User ID
	Output: acName	User name
	pucReturnStatus	Command status
Return values	HVCW_SUCCESS	Normal end
	HVCW_NOHANDLE	Handle error - improper handle content
	HVCW_INVALID_PARAM	Parameter error - NULL pointer argument - set value out of specifications
	HVCW_DISCONNECTED	Camera disconnected
	HVCW_TIMEOUT	Timeout error
	HVCW_INVALID_RECEIVEDATA	Invalid data transmitted
	HVCW_NOT_INITIALIZE	Initialization error
	HVCW_FAILURE	Unspecified error
Description	Gets the registered name of a specified user in the Album.	
Input specifications	nUserID: 0 to 499	

■ Delete Album data

```
HVCW_INT32 HVCW_ALBUM_DeleteData(HHVC hHVC, HVCW_INT32 nUserID,
                                 HVCW_INT32 nDataID, HVCW_UINT8 *pucReturnStatus)
```

Arguments	Input: hHVC	HVC handle
	nUserID	User ID
	nDataID	Data ID
	Output: pucReturnStatus	Command status
Return values	HVCW_SUCCESS	Normal end
	HVCW_NOHANDLE	Handle error - improper handle content
	HVCW_INVALID_PARAM	Parameter error - NULL pointer argument - set value out of specifications
	HVCW_DISCONNECTED	Camera disconnected
	HVCW_TIMEOUT	Timeout error
	HVCW_INVALID_RECEIVEDATA	Invalid data transmitted
	HVCW_NOT_INITIALIZE	Initialization error
	HVCW_FAILURE	Unspecified error
Description	Deletes specified data from a specified user from the Album. Set -1 in nDataID to erase all the data registered to a specified user.	
Input specifications	nUserID: 0 to 499 nDataID: 0 to 9 or -1	

■ Delete Album

```
HVCW_INT32 HVCW_ALBUM_DeleteAllData(HHVC hHVC, HVCW_UINT8 *pucReturnStatus)
```

Arguments	Input: hHVC	HVC handle
	Output: pucReturnStatus	Command status
Return values	HVCW_SUCCESS	Normal end
	HVCW_NOHANDLE	Handle error - improper handle content
	HVCW_INVALID_PARAM	Parameter error - NULL pointer argument
	HVCW_DISCONNECTED	Camera disconnected
	HVCW_TIMEOUT	Timeout error
	HVCW_INVALID_RECEIVEDATA	Invalid data transmitted
	HVCW_NOT_INITIALIZE	Initialization error
	HVCW_FAILURE	Unspecified error
Description	Deletes all the data for all the users in the Album.	

■ **Get user status**

```
HVCW_INT32 HVCW_ALBUM_GetRegistrationStatus(HHVC hHVC, HVCW_INT32 nUserID,
                        HVCW_BOOL abExist[10], HVCW_UINT8 *pucReturnStatus)
```

Arguments	Input:	hHVC	HVC handle
		nUserID	User ID
	Output:	abExist	Registration flag
		pucReturnStatus	Command status
Return values	HVCW_SUCCESS		Normal end
	HVCW_NOHANDLE		Handle error - improper handle content
	HVCW_INVALID_PARAM		Parameter error - NULL pointer argument - set value out of specifications
	HVCW_DISCONNECTED		Camera disconnected
	HVCW_TIMEOUT		Timeout error
	HVCW_INVALID_RECEIVEDATA		Invalid data transmitted
	HVCW_NOT_INITIALIZE		Initialization error
	HVCW_FAILURE		Unspecified error
Description	Gets the registration status of a designated user in the Album. Specify a user ID in nUserID. The value output in abExist will be TRUE if the specified user is registered in the Album and FALSE otherwise.		
Input specifications	nUserID: 0 to 499		

■ **Get Album size**

```
HVCW_INT32 HVCW_ALBUM_GetSize(HHVC hHVC, HVCW_INT32 *pnAlbumSize,
                                        HVCW_UINT8 *pucReturnStatus)
```

Arguments	Input:	hHVC	HVC handle
	Output:	pnAlbumSize	Album size
		pucReturnStatus	Command status
Return values	HVCW_SUCCESS		Normal end
	HVCW_NOHANDLE		Handle error - improper handle content
	HVCW_INVALID_PARAM		Parameter error - NULL pointer argument
	HVCW_DISCONNECTED		Camera disconnected
	HVCW_TIMEOUT		Timeout error
	HVCW_INVALID_RECEIVEDATA		Invalid data transmitted
	HVCW_NOT_INITIALIZE		Initialization error
	HVCW_FAILURE		Unspecified error
Description	Gets the Album size.		

■ Download Album

```
HVCW_INT32 HVCW_ALBUM_Download(HHVC hHVC, HVCW_UINT8 *pucAlbum,
                                HVCW_UINT8 *pucReturnStatus)
```

Arguments	Input:	hHVC	HVC handle
	Output:	pucAlbum	Album data
		pucReturnStatus	Command status
Return values	HVCW_SUCCESS		Normal end
	HVCW_NOHANDLE		Handle error - improper handle content
	HVCW_INVALID_PARAM		Parameter error - NULL pointer argument
	HVCW_DISCONNECTED		Camera disconnected
	HVCW_TIMEOUT		Timeout error
	HVCW_INVALID_RECEIVEDATA		Invalid data transmitted
	HVCW_NOT_INITIALIZE		Initialization error
	HVCW_FAILURE		Unspecified error
Description	Downloads and saves the Album data currently stored on the camera.		

■ Upload Album

```
HVCW_INT32 HVCW_ALBUM_Upload(HHVC hHVC, HVCW_INT32 nAlbumSize,
                  const HVCW_UINT8 *pucAlbum, HVCW_UINT8 *pucReturnStatus)
```

Arguments	Input:	hHVC	HVC handle
		nAlbumSize	Album size
		pucAlbum	Album
	Output:	pucReturnStatus	Command status
Return values	HVCW_SUCCESS		Normal end
	HVCW_NOHANDLE		Handle error - improper handle content
	HVCW_INVALID_PARAM		Parameter error - NULL pointer argument
	HVCW_DISCONNECTED		Camera disconnected
	HVCW_TIMEOUT		Timeout error
	HVCW_INVALID_RECEIVEDATA		Invalid data transmitted
	HVCW_NOT_INITIALIZE		Initialization error
	HVCW_FAILURE		Unspecified error
Description	Uploads and saves the Album data to the camera side.		

■ Save Album data

```
HVCW_INT32 HVCW_ALBUM_Save(HHVC hHVC, HVCW_UINT8 *pucReturnStatus)
```

Arguments	Input:	hHVC	HVC handle
	Output:	pucReturnStatus	Command status
Return values	HVCW_SUCCESS		Normal end
	HVCW_NOHANDLE		Handle error - improper handle content
	HVCW_INVALID_PARAM		Parameter error - NULL pointer argument
	HVCW_DISCONNECTED		Camera disconnected
	HVCW_TIMEOUT		Timeout error
	HVCW_INVALID_RECEIVEDATA		Invalid data transmitted
	HVCW_NOT_INITIALIZE		Initialization error
	HVCW_FAILURE		Unspecified error
Description	Saves the Album on the Flash ROM of the camera.		

■ **Set scheduler**

```
HVCW_INT32 HVCW_SetScheduler(HHVC hHVC, HVCW_SCHEDULER_TYPE schedulerType,
                    HVCW_BOOL bEnable, const HVCW_SCHEDULE_INFO *pSchedule,
                                                HVCW_UINT8 *pucReturnStatus)
```

Arguments	Input:	hHVC	HVC handle
		schedulerType	Schedule type
		bEnable	Schedule status
		pSchedule	Schedule info
	Output:	pucReturnStatus	Command status
Return values	HVCW_SUCCESS		Normal end
	HVCW_NOHANDLE		Handle error - improper handle content
	HVCW_INVALID_PARAM		Parameter error - NULL pointer argument - set value out of specifications
	HVCW_DISCONNECTED		Camera disconnected
	HVCW_TIMEOUT		Timeout error
	HVCW_INVALID_RECEIVEDATA		Invalid data transmitted
	HVCW_NOT_INITIALIZE		Initialization error
	HVCW_FAILURE		Unspecified error
Description	Sets a schedule in the scheduler. Set the schedule type in scheduleType. Specify 0 to set a one time schedule event. The scheduler will only be effective once for the specified time and date. Specify 1 to set a repeat schedule event. The scheduler will be effective between the specified start time and end time. Refer to HVCW_SCHEDULE_INFO for details on the scheduler struct.		
Input specifications	schedulerType: HVCW_ScheduleType_OneTime or HVCW_ScheduleType_Repeat bEnable: TRUE or FALSE pSchedule.nIndex: 0 to 34 (if schedulerType is 0) pSchedule.frequency: 0 (once), 1 (daily), 2 (weekday) pSchedule.bWeekday[7]: TRUE or FALSE (if pSchedule.frequency is 2) pSchedule.startTime.nMonth: 1 to 12 pSchedule.startTime.nDay: 1 to 31 pSchedule.startTime.nHour: 0 to 23 pSchedule.startTime.nMinute: 0 to 59 pSchedule.endTime.nMonth: 1 to 12 pSchedule.endTime.nDay: 1 to 31 pSchedule.endTime.nHour: 0 to 23 pSchedule.endTime.nMinute: 0 to 59 pSchedule.nInterval: 1 to 86400(60*60*24) pSchedule.abFunction[HVCW_OkaoFunction_Max]: TRUE or FALSE pSchedule.saveLog: 0 (none) or 1 (all) or 2 (detection) pSchedule.saveImage: 0 (none) or 1 (all) or 2 (detection)		
Default value	bEnable: FALSE pSchedule.frequency: 0 pSchedule.bWeekday[7]: FALSE pSchedule.startTime.nYear: 2000 pSchedule.startTime.nMonth: 1 pSchedule.startTime.nDay: 1 pSchedule.startTime.nHour: 0 pSchedule.startTime.nMinute: 0 pSchedule.endTime.nYear: 2000 pSchedule.endTime.nMonth: 1 pSchedule.endTime.nDay: 1 pSchedule.endTime.nHour: 1		

	pSchedule.endTime.nMinute: 1 pSchedule.nInterval: 1 pSchedule.abFunction[HVCW_OkaoFunction_Max]: FALSE pSchedule.saveLog: 0 pSchedule.saveImage: 0

■ **Get scheduler**

```
HVCW_INT32 HVCW_GetScheduler(HHVC hHVC, HVCW_SCHEDULER_TYPE schedulerType,
                             HVCW_BOOL *pbEnable, HVCW_SCHEDULE_INFO *pSchedule,
                                                  HVCW_UINT8 *pucReturnStatus)
```

Arguments	Input:	hHVC	HVC handle
		schedulerType	Schedule type
	Output:	pbEnable	Schedule status
		pSchedule	Schedule info
		pucReturnStatus	Command status
Return values	HVCW_SUCCESS		Normal end
	HVCW_NOHANDLE		Handle error - improper handle content
	HVCW_INVALID_PARAM		Parameter error - NULL pointer argument - set value out of specifications
	HVCW_DISCONNECTED		Camera disconnected
	HVCW_TIMEOUT		Timeout error
	HVCW_INVALID_RECEIVEDATA		Invalid data transmitted
	HVCW_NOT_INITIALIZE		Initialization error
	HVCW_FAILURE		Unspecified error
Description	Gets info on the specified scheduler. Specify a value in pSchedule > nIndex before calling this function in order to obtain the value set in OneTimeSchedule.		
Input specifications	schedulerType: HVCW_ScheduleType_OneTime or HVCW_ScheduleType_Repeat pSchedule.nIndex: 0 to 34		

■ **Set event program**

```
HVCW_INT32 HVCW_SetEventProgram(HHVC hHVC,
                HVCW_EVENT_PROGRAM_TYPE eventProgramType, HVCW_BOOL bEnable,
      const HVCW_EVENT_PROGRAM *pEventProgram, HVCW_UINT8 *pucReturnStatus)
```

Arguments	Input: hHVC	HVC handle
	eventProgramType	Event program type
	bEnable	Schedule status
	pEventProgram	Event program
	Output: pucReturnStatus	Command status
Return values	HVCW_SUCCESS	Normal end
	HVCW_NOHANDLE	Handle error - improper handle content
	HVCW_INVALID_PARAM	Parameter error - NULL pointer argument - set value out of specifications
	HVCW_DISCONNECTED	Camera disconnected
	HVCW_TIMEOUT	Timeout error
	HVCW_INVALID_RECEIVEDATA	Invalid data transmitted
	HVCW_NOT_INITIALIZE	Initialization error
	HVCW_FAILURE	Unspecified error
Description	Sets an event program. Select the event program type from sound detection events, motion detection events or timer events. Note that when selecting a sound detection or motion detection event the following parameters will not be set and this will require calling additional functions. bEnable pEventProgram > eventProgram > motionEvent.area.nX pEventProgram > eventProgram > motionEvent.area.nY pEventProgram > eventProgram > motionEvent.area.nWidth pEventProgram > eventProgram > motionEvent.area.nHeight pEventProgram > eventProgram > motionEvent.nSensitivity pEventProgram > eventProgram > soundEvent.nSensitivity Call HVCW_EnableMotionDetection() to start or change the motion detection process and HVCW_DisableMotionDetection() to end it. Call HVCW_EnableSoundDetection() to start the sound detection process and HVCW_DisableSoundDetection() to end it.	
Input specifications	eventProgramType: 0 (sound detection), 1 (motion detection), 2 (timer) bEnable: TRUE or FALSE pEventProgram > nIndex: 0 to 9 (for motion detection only) pEventProgram > eventProgram (HVCW_EVENT_PROGRAM_MOTION) > motionEvent.area.nX: 0 to 1919 motionEvent.area.nY: 0 to 1079 motionEvent.area.nWidth: 1 to 1920 motionEvent.area.nHeight: 1 to 1080 motionEvent.nSensitivity: 0 to 4 motionEvent.unDuration: 1 to 10 motionEvent.unRatio: 1 to 100 motionEvent.unOffPeriod: 0 to 86400 (60*60*24) pEventProgram > eventProgram (HVCW_EVENT_PROGRAM_SOUND) > soundEvent.nSensitivity: 0 to 4 soundEvent.unDuration: 1 to 10 soundEvent.unRatio: 1 to 100 soundEvent.unOffPeriod: 0 to 86400 (60*60*24) pEventProgram > eventProgram (HVCW_EVENT_PROGRAM_TIMER) >	

	timerEvent.unInterval: 1 to 86400 (60*60*24) pEventProgram > eventProgram (common) > okaoProcess.body.bEnable: TRUE or FALSE okaoProcess.body.nCount: 0 to 35 okaoProcess.pet.bEnable: TRUE or FALSE okaoProcess.pet.nCount: 0 to 10 okaoProcess.face.bEnable: TRUE or FALSE okaoProcess.face.nCount: 0 to 35 okaoProcess.age.bEnable: TRUE or FALSE okaoProcess.age.stAgeRange.nMin: 0 to 75 okaoProcess.age.stAgeRange.nMax: 0 to 75 okaoProcess.age.nConfidence: 0 to 1000 okaoProcess.gender.bEnable: TRUE or FALSE okaoProcess.gender.nGender: 0 (female), 1 (male), -1 (ignore) okaoProcess.gender.nConfidence: 0 to 1000 okaoProcess.expression.bEnable: TRUE or FALSE okaoProcess.expression.expression: 0 (neutral), 1 (happiness), 2 (surprise), 3 (anger), 4 (sadness), -1 (ignore) okaoProcess.expression.nScore: 0 to 100 okaoProcess.expression.degreeRange.nMin: -100 to 100 okaoProcess.expression.degreeRange.nMax: -100 to 100 okaoProcess.recognition.bEnable: TRUE or FALSE postProcess.saveLog: 0 (none), 1 (all), 2 (detection) postProcess.saveImage: 0 (none), 1 (all), 2 (detection) postProcess.notification.pushAlert.bEnable: TRUE or FALSE postProcess.notification.sound.bEnable: TRUE or FALSE frPostProcess.saveImage: 0 (none), 1 (all), 2 (user), 3 (unknown) frPostProcess.notification.pushAlert.knownUser.nUserID: 0 to 499, -1 (ignore) frPostProcess.notification.pushAlert.unknownUser.bEnable: TRUE or FALSE frPostProcess.notification.sound.knownUser.nUserID: 0 to 499, -1 (ignore) frPostProcess.notification.sound.unknownUser.bEnable: TRUE or FALSE
Default values	bEnable: FALSE pEventProgram > eventProgram(HVCW_EVENT_PROGRAM_MOTION) > motionEvent.area.nX: 0 motionEvent.area.nY: 0 motionEvent.area.nWidth: 1920 motionEvent.area.nHeight: 1080 motionEvent.nSensitivity: 3 motionEvent.unDuration: 1 motionEvent.unRatio: 100 motionEvent.unOffPeriod: 0 pEventProgram > eventProgram(HVCW_EVENT_PROGRAM_SOUND) > soundEvent.nSensitivity: 3 soundEvent.unDuration: 1 soundEvent.unRatio: 100 soundEvent.unOffPeriod: 0 pEventProgram > eventProgram(HVCW_EVENT_PROGRAM_TIMER) > timerEvent.unInterval: 1 pEventProgram > eventProgram (common) > okaoProcess.body.bEnable: FALSE

	okaoProcess.body.nCount: 1 okaoProcess.pet.bEnable: FALSE okaoProcess.pet.nCount: 1 okaoProcess.face.bEnable: FALSE okaoProcess.face.nCount: 1 okaoProcess.age.bEnable: FALSE okaoProcess.age.stAgeRange.nMin: 0 okaoProcess.age.stAgeRange.nMax: 75 okaoProcess.age.nConfidence: 0 okaoProcess.gender.bEnable: FALSE okaoProcess.gender.nGender: -1 okaoProcess.gender.nConfidence: 0 okaoProcess.expression.bEnable: FALSE okaoProcess.expression.expression: -1 okaoProcess.expression.nScore: 0 okaoProcess.expression.degreeRange.nMin: -100 okaoProcess.expression.degreeRange.nMax: 100 okaoProcess.recognition.bEnable: FALSE postProcess.saveLog: 0 postProcess.saveImage: 0 postProcess.notification.pushAlert.bEnable: FALSE postProcess.notification.sound.bEnable: FALSE frPostProcess.saveImage: 0 frPostProcess.notification.pushAlert.knownUser.nUserID: -1 frPostProcess.notification.pushAlert.unknownUser.bEnable: FALSE frPostProcess.notification.sound.knownUser.nUserID: -1 frPostProcess.notification.sound.unknownUser.bEnable: FALSE

■ **Get event program**

```
HVCW_INT32 HVCW_GetEventProgram(HHVC hHVC,
            HVCW_EVENT_PROGRAM_TYPE eventProgramType, HVCW_BOOL *pbEnable,
           HVCW_EVENT_PROGRAM *pEventProgram, HVCW_UINT8 *pucReturnStatus)
```

Arguments	Input:	hHVC	HVC handle
		eventProgramType	Event program type
	Output:	bEnable	Schedule status
		pEventProgram	Event program
		pucReturnStatus	Command status
Return values	HVCW_SUCCESS		Normal end
	HVCW_NOHANDLE		Handle error - improper handle content
	HVCW_INVALID_PARAM		Parameter error - NULL pointer argument - set value out of specifications
	HVCW_DISCONNECTED		Camera disconnected
	HVCW_TIMEOUT		Timeout error
	HVCW_INVALID_RECEIVEDATA		Invalid data transmitted
	HVCW_NOT_INITIALIZE		Initialization error
	HVCW_FAILURE		Unspecified error
Description	Gets the event program. Set the event index value in pEventProgram > nIndex before calling this function when getting detection motion event programs.		
Input specifications	eventProgramType: 0 (sound detection), 1 (motion detection), 2 (timer) pEventProgram > nIndex: 0 to 9		

1.8 Struct Definitions

■ **Video frame**

HVCW_VIDEOFRAME

Members	HVCW_BYTE *buffer[3]	Luminance and color difference info
	HVCW_UINT32 stride[3]	Stride info of each buffer
	HVCW_UINT32 width	Width
	HVCW_UINT32 height	Height
Description	This is the struct storing the video frame output in YUV420P.	

■ **Coordinate point**

HVCW_POINT

Members	HVCW_INT32 nX	X coordinates
	HVCW_INT32 nY	Y coordinates
Description	This is the struct storing the coordinates of a point.	

■ **Rectangle**

HVCW_RECT

Members	HVCW_INT32 nX	Point x coordinates
	HVCW_INT32 nY	Point y coordinates
	HVCW_INT32 nWidth	Width
	HVCW_INT32 nHeight	Height
Description	This is the struct storing the coordinates and size of a result rectangle.	

■ **Motion detection parameter**

HVCW_DETECTIONPARAM

Members	HVCW_RECT rect	Detection area range
	HVCW_UINT8 sensitivity	Sensitivity
Description	This is the struct storing the detection area range and the detection sensitivity.	

■ Schedule info

HVCW_SCHEDULE_INFO

Members	HVCW_INT32 nIndex	One time scheduler index *
	HVCW_SCHEDULE_FREQUENCY frequency	Scheduler frequency
	HVCW_BOOL bWeekday[7]	Day setting **
	HVCW_SCHEDULE_TIME startTime	Start time
	HVCW_SCHEDULE_TIME endTime	End time ***
	HVCW_INT32 nInterval	Process interval
	HVCW_BOOL abFunction[HVCW_OkaoFunction_Max]	OKAO function flag
	HVCW_SAVE_RESULT saveLog	Log save method
	HVCW_SAVE_RESULT saveImage	Image save method

Description

This is the struct storing the schedule info.

* nIndex is only used if the scheduler is set for a one-time event.
The scheduler frequency is specified in frequency. The possible values are HVCW_ScheduleFrequency_Once, HVCW_SchedulerFrequency_Daily and HVCW_ScheduleFrequency_Weekday.

	One time	Repeat
Once	Execute once at a set time	Execute for a set duration (from start time to end time)
Daily	Execute every day at set time	Execute every day for a set duration (from start time to end time)
Weekly	Execute on set day at set time	Execute on a set day for a set duration (from start to end time)

** bWeekday is used to specify the weekdays if the frequency is set for weekly events.

*** Set the end time in endtime only if repeating the event.

Specify the interval between the end of a process and the start of the next in nInterval.
Specify the OKAO functions to be executed in abFunction.
Refer to HVCW_OKAO_FUNCTION for details on the OKAO functions.

Set the save conditions for the log in saveLog and for the image in saveImage. The possible values are as follows.

0 (none) The log and/or image are not saved.
1 (all) The log and/or image are always saved.
2 (detection) The log and/or image will be saved if detection with the function specified with abFunction was successful.

■ Event program

HVCW_EVENT_PROGRAM

Members	HVCW_INT32 nIndex	Motion detection event index *
	HVCW_VOID* eventProgram	Event setting
Description	This is the struct storing all the event programs. * Only used for motion detection events.	

■ Motion detection event program

HVCW_EVENT_PROGRAM_MOTION

Members	HVCW_EP_MOTION_EVENT motionEvent	Motion detection event settings
	HVCW_EP_OKAO_PROCESS okaoProcess	OKAO process settings
	HVCW_EP_POST_PROCESS postProcess	Post processing settings
	HVCW_EP_FR_POST_PROCESS frPostProcess	Post processing settings for FR
Description	This is the struct storing the motion detection event programs.	

■ Sound detection event program

HVCW_EVENT_PROGRAM_SOUND

Members	HVCW_EP_SOUNDEVENT soundEvent	Sound detection event settings
	HVCW_EP_OKAO_PROCESS okaoProcess	OKAO process settings
	HVCW_EP_POST_PROCESS postProcess	Post processing settings
	HVCW_EP_FR_POST_PROCESS frPostProcess	Post processing settings for FR
Description	This is the struct storing the sound detection event programs.	

■ Timer event program

HVCW_EVENT_PROGRAM_TIMER

Members	HVCW_EP_TIMER_EVENT timerEvent	Timer event settings
	HVCW_EP_OKAO_PROCESS okaoProcess	OKAO process settings
	HVCW_EP_POST_PROCESS postProcess	Post processing settings
	HVCW_EP_FR_POST_PROCESS frPostProcess	Post processing settings for FR
Description	This is the struct storing the timer event programs.	

■ Motion detection event

HVCW_EP_MOTION_EVENT

Members	HVCW_RECT area	Motion detection area
	HVCW_INT32 nSensitivity	Motion detection sensitivity
	HVCW_UINT32 unDuration	Motion detection duration
	HVCW_UINT32 unRatio	Motion detection ratio
	HVCW_UINT32 unOffPeriod	Event interval
Description	This is the struct storing the motion detection info. The sensitivity of the motion detection is set in nSensitivity. The duration (in seconds) of the motion detection process is set in unDuration. The ratio of the motion detection set in unRatio indicates the required ratio in percentage of successful motion detections during the unDuration period to conclude in a motion detection event. The event interval set in unOffPeriod indicates the interval before starting a new motion detection process after motion has been detected. Increasing the interval will reduce the frequency of motion detection events happening.	

■ Sound detection event

HVCW_EP_SOUND_EVENT

Members	HVCW_INT32 nSensitivity	Sound detection sensitivity
	HVCW_UINT32 unDuration	Sound detection duration
	HVCW_UINT32 unRatio	Sound detection ratio
	HVCW_UINT32 unOffPeriod	Event interval
Description	This is the struct storing the sound detection info. The sensitivity of the sound detection is set in nSensitivity. The duration (in seconds) of the sound detection process is set in unDuration. The ratio of the sound detection set in unRatio indicates the required ratio in percentage of successful sound detections during the unDuration period to conclude in a sound detection event. The event interval set in unOffPeriod indicates the interval before starting a new sound detection process after sound has been detected. Increasing the interval will reduce the frequency of sound detection events happening.	

■ Timer event

HVCW_EP_TIMER_EVENT

Members	HVCW_UINT32 unInterval	Event interval
Description	This is the struct storing the event interval info for timer events. The interval period before the next event is stored in unInterval. Increasing the interval will reduce the frequency of timer events occurring.	

■ OKAO process

HVCW_EP_OKAO_PROCESS

Members	HVCW_EP_BODY body	Body value
	HVCW_EP_PET pet	Pet value
	HVCW_EP_FACE face	Face value
	HVCW_EP_AGE age	Age value
	HVCW_EP_GENDER gender	Gender value
	HVCW_EP_EXPRESSION expression	Expression value
	HVCW_EP_RECOGNITION recognition	Recognition value
Description	This is the struct storing the OKAO process conditions used for events.	

■ Human body detection

HVCW_EP_BODY

Members	HVCW_BOOL bEnable	Human body detection flag
	HVCW_INT32 nCount	Result count
Description	This is the struct storing the human body detection function flag. Set bEnable to TRUE to enable the condition requirements for human body detection. The conditions will be fulfilled if the number of human bodies detected is higher or equal to the value set in nCount.	

■ Pet detection

HVCW_EP_PET

Members	HVCW_BOOL bEnable	Pet detection flag
	HVCW_INT32 nCount	Result count
Description	This is the struct storing the pet detection function flag. Set bEnable to TRUE to enable the condition requirements for pet detection. The conditions will be fulfilled if the number of pets detected is higher or equal to the value set in nCount.	

■ Face detection

HVCW_EP_FACE

Members	HVCW_BOOL bEnable	Face detection flag
	HVCW_INT32 nCount	Result count
Description	This is the struct storing the face detection function flag. Set bEnable to TRUE to enable the condition requirements for face detection. The conditions will be fulfilled if the number of faces detected is higher or equal to the value set in nCount.	

■ Age estimation

HVCW_EP_AGE

Members	HVCW_BOOL bEnable	Age estimation flag
	HVCW_RANGE stAgeRange	Estimated age
	HVCW_INT32 nConfidence	Degree of confidence
Description	This is the struct storing the age estimation function flag. Set bEnable to TRUE to enable the condition requirements for age estimation. The conditions will be fulfilled if the age is within the age range set in stAgeRange and the degree of confidence is higher or equal to the value set in nConfidence.	

■ Gender estimation

HVCW_EP_GENDER

Members	HVCW_BOOL bEnable	Gender estimation flag
	HVCW_INT32 nGender	Estimated gender
	HVCW_INT32 nConfidence	Degree of confidence
Description	This is the struct storing the gender estimation function flag. Set bEnable to TRUE to enable the condition requirements for gender estimation. The conditions will be fulfilled if the gender is the gender set in nGender and the degree of confidence is higher or equal to the value set in nConfidence.	

■ Expression estimation

HVCW_EP_EXPRESSION

Members	HVCW_BOOL bEnable	Expression estimation flag
	HVCW_EXPRESSION expression	Estimated expression
	HVCW_INT32 nScore	Expression score
	HVCW_RANGE degreeRange	Expression degree
Description	This is the struct storing the expression estimation function flag. Set bEnable to TRUE to enable the condition requirements for age estimation. The conditions will be fulfilled if for the expression set in expression the expression score is equal to or higher than the value set in nScore or if the expression degree is higher or equal to the value set in degreeRange.	

■ Post processing

HVCW_EP_POST_PROCESS

Members	HVCW_SAVE_RESULT saveLog	Log save
	HVCW_SAVE_RESULT saveImage	Image save
	HVCW_EP_NOTIFICATION notification	Notification
Description	This is the struct storing the saved log and saved images conditions for an event.	

■ Notification settings

HVCW_EP_NOTIFICATION

Members	HVCW_EP_PUSH_ALERT pushAlert	Push alert
	HVCW_EP_SOUND sound	Sound output
Description	This is the struct storing the push notification and sound output conditions for an event.	

■ Push notification

HVCW_EP_PUSH_ALERT

Members	HVCW_BOOL bEnable	Push alert flag
	HVCW_FILEINFO fileInfo	Push alert message file
Description	This is the struct storing the push alert conditions for an event. Set bEnable to TRUE to enable push notifications and to FALSE otherwise. Specify the notification message file in fileInfo.	

■ Sound notification

HVCW_EP_SOUND

Members	HVCW_BOOL bEnable	Sound output flag
	HVCW_FILEINFO fileInfo	Sound output file
Description	This is the struct storing the sound output conditions for an event. Set bEnable to TRUE to enable sound notifications and to FALSE otherwise. Specify the sound file in fileInfo.	

■ Face recognition

HVCW_EP_RECOGNITION

Member	HVCW_BOOL bEnable	Face recognition flag
Description	This is the struct storing the face recognition flag. Set bEnable to TRUE to enable face recognition and to FALSE otherwise.	

■ Face recognition post processing

HVCW_EP_FR_POST_PROCESS

Members	HVCW_FR_SAVE_RESULT saveImage	Image save
	HVCW_EP_FR_NOTIFICATION notification	Face recognition notification
Description	This is the struct storing the face recognition saved image and notification.	

■ Face recognition notification setting

HVCW_EP_FR_NOTIFICATION

Members	HVCW_EP_FR_PUSH_ALERT pushAlert	Push alert setting
	HVCW_EP_FR_SOUND sound	Sound output setting
Description	This is the struct storing the push alert notification and sound output conditions for face recognition.	

■ **Face recognition push notification**

HVCW_EP_FR_PUSH_ALERT

Members	HVCW_EP_KNOWNUSER knownUser[20]	Process for known user
	HVCW_EP_PUSH_ALERT unknownUser	Process for unknown user
Description	This is the struct containing the push notification content conditions for face recognition. The conditions can be specified for both known and unknown users.	

■ **Face recognition sound notification**

HVCW_EP_FR_SOUND

Members	HVCW_EP_KNOWNUSER knownUser[20]	Known user
	HVCW_EP_SOUND unknownUser	Unknown user
Description	This is the struct containing the sound output content conditions for face recognition. The conditions can be specified for both known and unknown users.	

■ **Known user**

HVCW_EP_KNOWNUSER

Members	HVCW_INT32 nUserID	User ID
	HVCW_FILEINFO fileInfo	File info
Description	This is the struct containing the notification contents for known users. Specify a sound output file or push notification file in fileInfo for a user specified in nUserID.	

■ **Specification range**

HVCW_RANGE

Members	HVCW_INT32 nMin	Minimum value
	HVCW_INT32 nMax	Maximum value

■ **Detection threshold value**

HVCW_OKAO_THRESHOLD

Members	HVCW_INT32 nBody	Threshold value for body detection
	HVCW_INT32 nHand	Threshold value for hand detection
	HVCW_INT32 nPet	Threshold value for pet detection
	HVCW_INT32 nFace	Threshold value for face detection
	HVCW_INT32 nRecognition	Threshold value for face recognition

■ **Detection size**

HVCW_OKAO_SIZE_RANGE

Members	HVCW_RANGE body	Detection size for bodies
	HVCW_RANGE hand	Detection size for hands
	HVCW_RANGE pet	Detection size for pets
	HVCW_RANGE face	Detection size for faces

■ **Detection angle**

HVCW_OKAO_DETECTION_ANGLE

Members	HVCW_UINT32 nBody	Detection angle for body detection
	HVCW_UINT32 nHand	Detection angle for hand detection
	HVCW_UINT32 nPet	Detection angle for pet detection
	HVCW_OKAO_DETECTION_ANGLE_FACE face	Detection angle for face detection

■ **Face detection angle**

`HVCW_OKAO_DETECTION_ANGLE_FACE`

Members	`HVCW_UINT32 nPose`	Facial pose (yaw)
	`HVCW_UINT32 nAngle`	Face angle (roll)

■ **Detection result**

`HVCW_OKAO_RESULT`

Members	`HVCW_OKAO_RESULT_BODYS bodys`	Results for body detection
	`HVCW_OKAO_RESULT_HANDS hands`	Results of hand detection
	`HVCW_OKAO_RESULT_PETS pets`	Results for pet detection
	`HVCW_OKAO_RESULT_FACES faces`	Results for face detection

■ **Body detection result**

`HVCW_OKAO_RESULT_BODYS`

Members	`HVCW_INT32 nCount`	Result count for bodies
	`HVCW_OKAO_RESULT_DETECTION body[35]`	Result for bodies

■ **Hand detection result**

`HVCW_OKAO_RESULT_HANDS`

Members	`HVCW_INT32 nCount`	Result count for hands
	`HVCW_OKAO_RESULT_DETECTION hand[35]`	Result for hands

■ **Pet detection result**

`HVCW_OKAO_RESULT_PETS`

Members	`HVCW_INT32 nCount`	Result count for pets
	`HVCW_OKAO_RESULT_PET pet[10]`	Result for pets

■ **Face detection result**

`HVCW_OKAO_RESULT_FACES`

Members	`HVCW_INT32 nCount`	Result count for faces
	`HVCW_OKAO_RESULT_FACE face[35]`	Result for faces

■ **Detailed detection results (hand/body)**

`HVCW_OKAO_RESULT_DETECTION`

Members	`HVCW_POINT center`	Center point coordinates
	`HVCW_INT32 nSize`	Size
	`HVCW_INT32 nConfidence`	Degree of confidence

■ **Detailed detection results (pet)**

`HVCW_OKAO_RESULT_PET`

Members	`HVCW_POINT ptCenter`	Center point coordinates
	`HVCW_INT32 nSize`	Size
	`HVCW_INT32 nConfidence`	Degree of confidence
	`HVCW_INT32 nPetType`	Pet type

■ **Detailed detection results (face)**

`HVCW_OKAO_RESULT_FACE`

Members	`HVCW_POINT center`	Center point coordinates
	`HVCW_INT32 nSize`	Size
	`HVCW_INT32 nConfidence`	Degree of confidence
	`HVCW_OKAO_RESULT_DIRECTION direction`	Face direction
	`HVCW_OKAO_RESULT_AGE age`	Estimated age
	`HVCW_OKAO_RESULT_GENDER gender`	Estimated gender
	`HVCW_OKAO_RESULT_GAZE gaze`	Gaze direction
	`HVCW_OKAO_RESULT_BLINK blink`	Blink degree
	`HVCW_OKAO_RESULT_EXPRESSION expression`	Expression result
	`HVCW_OKAO_RESULT_RECOGNITION recognition`	Face recognition result

■ **Detection angles**

`HVCW_OKAO_RESULT_DIRECTION`

Members	`HVCW_INT32 nLR`	Yaw (left-right) angle
	`HVCW_INT32 nUD`	Pitch (up-down) angle
	`HVCW_INT32 nRoll`	Roll angle
	`HVCW_INT32 nConfidence`	Degree of confidence

■ **Estimated age**

`HVCW_OKAO_RESULT_AGE`

Members	`HVCW_INT32 nAge`	Age
	`HVCW_INT32 nConfidence`	Degree of confidence

■ **Estimated gender**

`HVCW_OKAO_RESULT_GENDER`

Members	`HVCW_INT32 nGender`	Gender
	`HVCW_INT32 nConfidence`	Degree of confidence

■ **Estimated gaze**

`HVCW_OKAO_RESULT_GAZE`

Members	`HVCW_INT32 nLR`	Yaw (left-right) angle
	`HVCW_INT32 nUD`	Pitch (up-down) angle

■ **Estimated blink**

`HVCW_OKAO_RESULT_BLINK`

Members	`HVCW_INT32 nLeftEye`	Left eye blink degree
	`HVCW_INT32 nRightEye`	Right eye blink degree

■ **Estimated expression**

`HVCW_OKAO_RESULT_EXPRESSION`

Members	`HVCW_INT32 anScore[HVCW_Expression_Max]`	Expression score
	`HVCW_INT32 nDegree`	Expression degree (negative/positive)

■ **Face recognition score**

`HVCW_OKAO_RESULT_RECOGNITION`

Members	`HVCW_INT32 nUID`	User ID
	`HVCW_INT32 nScore`	Score

■ File info

HVCW_FILEINFO

Members	HVCW_CHAR acName[40]	File name
	HVCW_INT32 nSize	File size
	HVCW_UINT32 Reserved	Reserved

■ Schedule time

HVCW_SCHEDULE_TIME

Members	HVCW_INT32 nYear	Year (YYYY)
	HVCW_INT32 nMonth	Month
	HVCW_INT32 nDay	Day
	HVCW_INT32 nHour	Hour
	HVCW_INT32 nMinute	Minutes

■ Storage info

HVCW_STORAGEINFO

Members	HVCW_UINT32 ucTotalSize	Total storage size (KB)
	HVCW_UINT32 ucUsedSize	Used storage size (KB)
	HVCW_UINT32 ucFreeSize	Free storage size (KB)
	HVCW_STORAGE_STATUS storageStatus	Storage status

1.9 Enumeration Type Definitions

- **Video resolution**

```
typedef enum HVCW_VideoResolution{
  HVCW_VideoResolution_High,
  HVCW_VideoResolution_Middle,
  HVCW_VideoResolution_Low
}HVCW_VIDEO_RESOLUTION;
```

Definition	Description
HVCW_VideoResolution_High	0 = 1280×720
HVCW_VideoResolution_Middle	1 = 640×360
HVCW_VideoResolution_Low	2 = 320×180

- **Night vision mode**

```
typedef enum HVCW_NightVisionMode{
  HVCW_NightVisionMode_Auto,
  HVCW_NightVisionMode_Manual
}HVCW_NIGHT_VISION_MODE;
```

Definition	Description
HVCW_NightVisionMode_Auto	0 = Automatic activation of night vision depending on the brightness of the surroundings
HVCW_NightVisionMode_Manual	1 = Manual activation of night vision

- **Night vision status**

```
typedef enum HVCW_NightVisionStatus{
  HVCW_NightVisionStatus_Off,
  HVCW_NightVisionStatus_On
}HVCW_NIGHT_VISION_STATUS;
```

Definition	Description
HVCW_NightVisionStatus_Off	0 = Night vision OFF
HVCW_NightVisionStatus_On	1 = Night vision ON

■ **Event**

```
typedef enum HVCW_Event{
  HVCW_Event_ConnectionNum,
  HVCW_Event_StreamingNum,
  HVCW_Event_NightVisionMode,
  HVCW_Event_NightVisionStatus,
  HVCW_Event_SpeakerVolume,
  HVCW_Event_Disconnected,
  HVCW_Event_Reconnected,
  HVCW_Event_StorageStatus
}HVCW_EVENT;
```

Definition	Description
HVCW_Event_ConnectionNum	0 = Camera connection number change
HVCW_Event_StreamingNum	1 = Live streaming number change
HVCW_Event_NightVisionMode	6 = Night vision mode change
HVCW_Event_NightVisionStatus	7 = Night vision status change
HVCW_Event_SpeakerVolume	9 = Camera speaker volume change
HVCW_Event_Disconnected	13 = Camera disconnected
HVCW_Event_Reconnected	14 = Camera reconnected
HVCW_Event_StorageStatus	18 = Camera storage status change

■ **Storage format result**

```
typedef enum HVCW_StorageFormatResultCode{
  HVCW_StorageFormatResultCode_Success,
  HVCW_StorageFormatResultCode_RemovedStorage,
  HVCW_StorageFormatResultCode_NotSupportStorage,
  HVCW_StorageFormatResultCode_Timeout,
  HVCW_StorageFormatResultCode_AlreadyRunning,
  HVCW_StorageFormatResultCode_CheckDisk,
  HVCW_StorageFormatResultCode_FatalError
}HVCW_STORAGE_FORMAT_RESULT_CODE;
```

Definition	Description
HVCW_StorageFormatResultCode_Success	0 = Formatting complete
HVCW_StorageFormatResultCode_RemovedStorage	1 = Storage removed while formatting
HVCW_StorageFormatResultCode_NotSupportStorage	2 = Storage not supported
HVCW_StorageFormatResultCode_Timeout	3 = Timeout during formatting
HVCW_StorageFormatResultCode_AlreadyRunning	4 = Already formatting
HVCW_StorageFormatResultCode_CheckDisk	5 = Now checking disk
HVCW_StorageFormatResultCode_FatalError	6 = Unspecified error

- **Connection type**

```
typedef enum HVCW_ConnectionType{
  HVCW_ConnectionType_Disconnect,
  HVCW_ConnectionType_P2P,
  HVCW_ConnectionType_Relay,
  HVCW_ConnectionType_Local
}HVCW_CONNECTION_TYPE;
```

Definition	Description
HVCW_ConnectionType_Disconnect	0 = Not connected
HVCW_ConnectionType_P2P	1 = P2P connection
HVCW_ConnectionType_Relay	2 = Relay connection
HVCW_ConnectionType_Local	3 = Local connection

- **Live event**

```
typedef enum HVCW_LiveEvent{
  HVCW_LiveEvent_Started,
  HVCW_LiveEvent_Stopped,
  HVCW_LiveEvent_Disconnected,
  HVCW_LiveEvent_Error,
  HVCW_LiveEvent_FPS,
  HVCW_LiveEvent_ChangeResolution
}HVCW_LIVE_EVENT;
```

Definition	Description	Attached info
HVCW_LiveEvent_Started	0 = Video decoding started after live streaming start.	NULL
HVCW_LiveEvent_Stopped	1 = Live streaming stop.	NULL
HVCW_LiveEvent_Disconnected	2= Camera disconnected. It is required to call HVCW_StopLive() to stop the live streaming.	NULL
HVCW_LiveEvent_Error	3 = Live streaming interrupted. It is required to call HVCW_StopLive() to stop the live streaming.	NULL
HVCW_LiveEvent_FPS	4 = FPS info.	FPS
HVCW_LiveEvent_ChangeResolution	5 = Live streaming resolution change.	HVCW_VideoResolution

- **Scheduler type**

```
typedef enum HVCW_ScheduleType{
  HVCW_ScheduleType_OneTime,
  HVCW_ScheduleType_Repeat,
  HVCW_ScheduleType_Max
}HVCW_SCHEDULER_TYPE;
```

Definition	Description
HVCW_ScheduleType_OneTime	0 = One time scheduler
HVCW_ScheduleType_Repeat	1 = Repeat scheduler

■ **Event type**

```
typedef enum HVCW_EventProgramType{
  HVCW_EventProgramType_Sound,
  HVCW_EventProgramType_Motion,
  HVCW_EventProgramType_Timer,
  HVCW_EventProgramType_Max
}HVCW_EVENT_PROGRAM_TYPE;
```

Definition	Description
HVCW_EventProgramType_Sound	0 = Sound detection event
HVCW_EventProgramType_Motion	1 = Motion detection event
HVCW_EventProgramType_Timer	2 = Timer event

■ **Schedule frequency**

```
typedef enum HVCW_ScheduleFrequency{
  HVCW_ScheduleFrequency_Once,
  HVCW_ScheduleFrequency_Daily,
  HVCW_ScheduleFrequency_Weekday,
  HVCW_ScheduleFrequency_Max
}HVCW_SCHEDULE_FREQUENCY;
```

Definition	Description
HVCW_ScheduleFrequency_Once	0 = Execute once
HVCW_ScheduleFrequency_Daily	1 = Execute every day
HVCW_ScheduleFrequency_Weekday	2 = Execute on designated day

■ **Results save preferences**

```
typedef enum HVCW_FRSaveResult{
  HVCW_FRSaveResult_None,
  HVCW_FRSaveResult_All,
  HVCW_FRSaveResult_Known,
  HVCW_FRSaveResult_Unknown,
  HVCW_FRSaveResult_Max
}HVCW_FR_SAVE_RESULT;
```

Definition	Description
HVCW_FRSaveResult_None	0 = No result saved
HVCW_FRSaveResult_All	1 = All results saved
HVCW_FRSaveResult_Known	2 = Saved if known user
HVCW_FRSaveResult_Unknown	3 = Saved if unknown user

■ Image save preferences

```
typedef enum HVCW_SaveResult{
  HVCW_SaveResult_None,
  HVCW_SaveResult_All,
  HVCW_SaveResult_Detection,
  HVCW_SaveResult_Max
}HVCW_SAVE_RESULT;
```

Definition	Description
HVCW_SaveResult_None	0 = Results not saved
HVCW_SaveResult_All	1 = All results saved
HVCW_SaveResult_Detection	2= Saved if OKAO conditions met

■ File extension

```
typedef enum HVCW_FileExt{
  HVCW_FileExt_Log,
  HVCW_FileExt_MessageText,
  HVCW_FileExt_Sound,
  HVCW_FileExt_JpgImage,
  HVCW_FileExt_ThumbnailJpgImage
}HVCW_FILE_EXT;
```

Definition	Description
HVCW_FileExt_Log	2 = Log file (log)
HVCW_FileExt_MessageText	3 = Message text file (txt)
HVCW_FileExt_Sound	4 = Sound file (wav)
HVCW_FileExt_JpgImage	6 = Image file (jpg)
HVCW_FileExt_ThumbnailJpgImage	8 = Thumbnail image file (jpg)

■ **OKAO function**

```
typedef enum HVCW_OkaoFunction{
  HVCW_OkaoFunction_Body,
  HVCW_OkaoFunction_Hand,
  HVCW_OkaoFunction_Pet,
  HVCW_OkaoFunction_Face,
  HVCW_OkaoFunction_Direction,
  HVCW_OkaoFunction_Age,
  HVCW_OkaoFunction_Gender,
  HVCW_OkaoFunction_Gaze,
  HVCW_OkaoFunction_Blink,
  HVCW_OkaoFunction_Expression,
  HVCW_OkaoFunction_Recognition,
  HVCW_OkaoFunction_Max
}HVCW_OKAO_FUNCTION;
```

Definition	Description
HVCW_OkaoFunction_Body	0 = Human body detection
HVCW_OkaoFunction_Hand	1 = Hand detection
HVCW_OkaoFunction_Pet	2 = Pet detection
HVCW_OkaoFunction_Face	3 = Face detection
HVCW_OkaoFunction_Direction	4 = Face direction estimation
HVCW_OkaoFunction_Age	5 = Age estimation
HVCW_OkaoFunction_Gender	6 = Gender estimation
HVCW_OkaoFunction_Gaze	7 = Gaze estimation
HVCW_OkaoFunction_Blink	8 = Blink estimation
HVCW_OkaoFunction_Expression	9 = Expression estimation
HVCW_OkaoFunction_Recognition	10 = Face recognition

■ **Expression**

```
typedef enum HVCW_Expression{
  HVCW_Expression_Ignore,
  HVCW_Expression_Neutral,
  HVCW_Expression_Happiness,
  HVCW_Expression_Surprise,
  HVCW_Expression_Anger,
  HVCW_Expression_Sadness,
  HVCW_Expression_Max
}HVCW_EXPRESSION;
```

Definition	Description
HVCW_Expression_Ignore	-1 = Ignore
HVCW_Expression_Neutral	0 = Neutral
HVCW_Expression_Happiness	1 = Happiness
HVCW_Expression_Surprise	2 = Surprise
HVCW_Expression_Anger	3 = Anger
HVCW_Expression_Sadness	4 = Sadness

■ **Storage status**

```
typedef enum HVCW_StorageStatus{
  HVCW_StorageStatus_NotInsert,
  HVCW_StorageStatus_NotReady,
  HVCW_StorageStatus_NeedFormat,
  HVCW_StorageStatus_Normal,
  HVCW_StorageStatus_Error,
  HVCW_StorageStatus_Formatting,
  HVCW_StorageStatus_NotSupport
}HVCW_STORAGE_STATUS;
```

Definition	Description
HVCW_StorageStatus_NotInsert	0 = Storage not found (SD card not inserted)
HVCW_StorageStatus_NotReady	1 = Storage found but not useable
HVCW_StorageStatus_NeedFormat	2 = Storage requires formatting
HVCW_StorageStatus_Normal	3 = Storage useable
HVCW_StorageStatus_Error	4 = Storage unusable due to error
HVCW_StorageStatus_Formatting	6 = Storage formatting
HVCW_StorageStatus_NotSupport	7 = Storage not supported

1.10 Constant Definitions

```
#define HVCW_PET_TYPE_DOG  0       // Pet type: dog
#define HVCW_PET_TYPE_CAT  1       // Pet type: cat

#define HVCW_GENDER_FEMALE 0       // Gender: female
#define HVCW_GENDER_MALE   1       // Gender: male
```

1.11 Callback Type Definitions

■ **Event callback**

```
HVCW_INT32 (*HVCW_EventCallback)(HVCW_INT32 nEventId, HVCW_VOID *pUserParam,
                                 HVCW_VOID *pEventInfo)
```

Arguments	Input:	nEventId	Event ID
		pUserParam	User parameter
		pEventInfo	Event info
Return value	HVCW_SUCCESS		Normal end
Description	Callback function used for event notification.		

■ **Rendering request**

```
HVCW_INT32 (*HVCW_RequestRenderingCallback)(HVCW_BOOL bVideo,
     HVCW_VOID *pUserParam, HVCW_VOID *pRenderInfo, HVCW_UINT32 unInfoLen,
                                 HVCW_UINT32 unTimeStamp)
```

Arguments	Input:	bVideo	Video request (if TRUE) or sound request (if FALSE)
		pUserParam	User parameter
		pRenderInfo	Video frame or sound data
		unInfoLen	Render info size
		unTimeStamp	Time stamp
Return value	HVCW_SUCCESS		Normal end
Description	Callback function used to request display of video frames or playing of sound data during live streaming. Video frame details Item: Video format / Set value: YUV420SP Sound data details Item: Sampling rate / Set value: 8000 Item: Channel number / Set value: 1 Item: Audio format / Set value: Signed 16 bit PCM Item: Byte order / Set value: Little Endian If the callback function process is delayed, the following video frames and sound data may be skipping. The notified video frames or sound data can be freed by calling HVCW_FreeDecodedVideoBuffer() for the video buffer and HVCW_FreeDecodedAudioBuffer() for the sound buffer.		

■ **Live event notification**

```
HVCW_INT32 (*HVCW_LiveEventCallback)(HVCW_INT32 nEventId,
                                 HVCW_VOID *pUserParam, HVCW_VOID *pEventInfo)
```

Arguments	Input:	nEventId	Event ID
		pUserParam	User parameters
		pEventInfo	Event info
Return values	HVCW_SUCCESS		Normal end
Description	Callback function used for live event notification.		