Upgrade guide to SDK770 and above

Version 13.4.788.0 (September 2019)

Software release

Some new fields of the SETUP structure may be added in new software releases. This document is based on software release 788 (PhCon.dll, PhFile.dll, PhInt.dll version 13.4.788.0, PCC version 3.4.788.0).

VISION RESEARCH

AMETEK®
MATERIALS ANALYSIS DIVISION

100 Dey Road
Wayne, NJ 07470 USA
+1.973.696.4500
phantom@visionresearch.com

www.phantomhighspeed.com

Contents

1. Introduction

Phantom Dll's were designed to maintain a binary compatibility between versions so the access to new cameras models or software fixes is usually done just by replacing the ph*.dll files in the installation from the customer computer without any required change in the applications.

This new version of the Phantom SDK has the goal to introduce a new enumeration algorithm of the cameras that would improve the reliability of cameras detection on large and crowded networks with dozens of Phantom cameras. Disconnecting and reconnecting a camera became a simple process and you finally get live images from the camera in a few seconds after reconnection.

In the last years we added support for more than 63 partitions in the camera memory and for specifying fractional frame rates.

Using this new features involve some changing in your applications. Here you have a description of these changes. These changes were first introduced in our version 769.

2. Double frame rate

Some applications of the high speed cameras require specifying an exact fractional frame rate of the acquisition like the NTSC 29.97 fps. To provide support for double frame rate we added a new dFrameRate field at the end of ACQUIPARAMS and SETUP structures and renamed the old FrameRate field to FrameRateInt. This change will produce errors in your old code. What you have to do in your application is to replace FrameRate by dFrameRate and take into account the new field is of type double. If your customers are not interested in fractional frame rates you can stay on displaying integer values in your frame rate GUI (by converting dFrameRate).

3. More than 63 partitions

We used two constants in our previous phcon.h header:

```
#define MAXCINECNT 64
```

```
#define FIRST_FLASH_CINE 101
```

They were removed in this version and you have to replace them with calls to new functions:

```
int PhMaxCineCnt(UINT CN);
int PhFirstFlashCine(UINT CN);
```

This allows for simple increase of the cameras maximum number of partitions which is PhMaxCineCnt(CN) -1. Some of our current cameras support 511 partitions; most of them stay at 63.

4. Enumeration in the SDK

If you register your application for version 769 or newer, PhCheckCameraPool() will never notify any change. PhNotifyDeviceChange() will do nothing so you will remain with the cameras that were found at the start of the application.
To update the camera pool you will have to call immediately after PhRegisterClientEx() the function

```
HRESULT PhConfigPoolUpdate(UINT Period);
```

where Period is the interval in milliseconds when the software is looking (in a separate thread) for new cameras or for Offline cameras that were reconnected to the network. An example: Period = 3000; would provide a pretty fast update.

If an access to the camera produces a timeout error, the camera is switched to Offline mode and further accesses to the camera are immediately terminated with the same timeout error code. Requests for images will return dark images with "Offline" text stamped in the center.

You can call PhOffline to see if a camera is in this mode

```
BOOL PhOffline(UINT CN);
```

When the camera is reconnected, we will "revive" it, PhOffline will return FALSE and you can start to access it with commands or image requests.

The new enumeration guarantee that existing cameras remain in the same position in the pool; new cameras are added always at the end of the pool and PhGetCameraCount() will return the updated number of cameras. The absence of answer of an existing camera to broadcast will not switch the camera to Offline, only a timeout at a command sent to the camera will do that.

5. Visible list

If you want to connect only to the cameras that are specified in the list of IP addresses, you can build that list as in previous versions. The support for the Visible list was improved by adding new functions to enable the list (in previous

versions you had to remove all IP's from the list to look for all discovered cameras).

HRESULT PhDisableVisible(BOOL bDisabled);

BOOL PhIsVisibleDisabled(void);

You have a new function that will provide supplementary information about the camera that has the IP

```
HRESULT PhFillIDInfo(PCAMERAID pCam);
```

CAMERAID is the same structure used in PhGetAllIpCameras(). You have to fill the IP address of the camera in pCam->IP (it has to be net order i.e. big endian here) and the function will fill the Serial, Name, Model, CFA, CamVer. This information is provided even if the camera is absent now but was seen in the past at least once. It can be used to populate your list of cameras with more friendly details about the camera.

SDK will attempt to build the list of cameras exactly as it is in the active visible list. If a camera is absent at start it will be simulated, if it is connected later, it will be revived.

After changing the Visible list or switch between SeeAllDiscovered and SeeVisible please ask operator to restart the application because it is possible that some cameras have to be removed from the pool.

6. Compatibility with applications compiled for previous versions of the SDK

If the application was compiled for an older version, that is, PHCONHEADERVERSION < 769, the new dll's will remain compatible. You will still be able to give access to new Phantom camera in your old applications just by replacing the dll's with the new version.