Photon etc. LLTF Contrast Software Development Library

LLTF Contrast API Specification

Version 1.2.0 Revision 2016-02-15

Table of Contents

1. About This Manual 2

2. Customer Service 2

3. Introduction 3

4. Management Functions 4

4.1. PE_CREATE 4

4.2. PE_DESTROY 4

4.3. PE_GETSYSTEMCOUNT 5

4.4. PE_GETSYSTEMNAME 5

4.5. PE_GETLIBRARYVERSION 5

4.6. PE_GETSTATUSSTR 6

5. LLTF Contrast Control Functions 7

5.1. PE_CLOSE 7

5.2. PE_OPEN 7

5.3. PE_GETWAVELENGTH 8

5.4. PE_SETWAVELENGTH 8

5.5. PE_GETWAVELENGTHRANGE 8

5.6. PE_HASHARMONICFILTER 9

5.7. PE_GETHARMONICFILTERENABLED 9

5.8. PE_SETHARMONICFILTERENABLED 9

5.9. PE_GETGRATINGCOUNT 10

5.10. PE_GETGRATINGNAME 10

5.11. PE_GETGRATINGWAVELENGTHRANGE 11

5.12. PE_GETGRATINGWAVELENGTHEXTENDEDRANGE 11

5.13. PE_SETWAVELENGTHONGRATING 12

5.14. PE_GETGRATING 12

6. Status Codes 13

7. Appendix A – Code Example 14

8. Appendix B – Revision History 15

1. About This Manual

This manual describes how to use the software development kit (SDK) available for each Photon etc. LLTF Contrast system. It simplifies the integration of Photon etc. LLTF Contrast in a complex application by offering an intuitive and consistent interface.

2. Customer Service

Business hours:	Monday to Friday from 9:00 to 17:00 Eastern Standard Time (GMT -5)
Email:	info@photonetc.com
Phone:	(514) 385-9555
Address:	**Photon etc. inc.** 5795 De Gaspé Avenue, #222 Montréal, Québec H2S 2X3 Canada

To obtain more information regarding our products: sales@photonetc.com

To communicate with one of our experts regarding specific queries or applications: expert@photonetc.com

For technical support: support@photonetc.com

3. Introduction

LLTF Contrast SDK is intended for developers integrating a Photon etc. LLTF Contrast in their software. This SDK provides the necessary dynamic libraries (DLL) for both 32-bit and 64-bit architecture. It is important to note that all the drivers needed by the SDK must be installed before executing the software. All SDK functions are decorated with `'PE_'` prefix.

This manual is structured by grouping functions that have the same scope into sections. The **Management Functions** create and search for available LLTF Contrast. The **LLTF Contrast Control Functions** give access to every parameter of the system and its accessories. Finally, each **Status Code** is described and a code example written in `C` is given as a starting point to familiarize you with the SDK.

4. Management Functions

Almost all SDK functions require a valid PE_HANDLE structure to execute. To acquire a handle on this structure, you must have a configuration file describing all available Photon etc. LLTF Contrast systems. The configuration file *system.xml* is distributed with your system and is located in PHySpec installation directory. Be careful not to rename, move or delete this file otherwise PHySpec would not be able to charge any of your system. You can query the configuration file to have a list of available systems which do not have to be connected to be listed. Once you know the name of your system, you can connect to it using the PE_Open function. See **LLTF Contrast Control Functions** for more details about how to operate your system. All the management functions can be called without an actual system connected.

4.1. PE_CREATE

Acquire a handle on a LLTF Contrast which will be used for subsequent accesses. The configuration file *system.xml* is located in PHySpec installation directory.

This configuration file should not be modified but can be copied to another location for easier access. Take notice that the configuration file will change when PHySpec calibrates the LLTF Contrast. If you copied the configuration file to another location and recalibrate the LLTF Contrast, the new configuration will not be applied to your copied file. To avoid memory leaks and other system misbehaviours, you must call PE_Destroy when the handle is no longer needed.

```
PE_STATUS PE_Create(const char *conffile,
                    PE_HANDLE *peHandle)
```

Parameters

[in] conffile:	Path to the system configuration file including its file name
[out] peHandle:	Pointer to a PE_HANDLE structure where the handle is stored

Return value

Return PE_SUCCESS if successful, otherwise the return value is a PE_STATUS status code.

4.2. PE_DESTROY

Release a handle on LLTF Contrast previously acquired by PE_Create previously cleaned up before its destruction. If a system was opened, it will be properly closed and all other resources used will be freed. If the handle is NULL, this function has no effect.

```
PE_STATUS PE_Destroy(PE_HANDLE peHandle)
```

Parameters

[in] peHandle:	Handle to the system

Return value

Return PE_SUCCESS if successful, otherwise the return value is a PE_STATUS status code.

4.3. PE_GetSystemCount

Retrieve the number of systems listed in the configuration file previously selected on handle creation. This function retrieves the total number of systems, connected or not.

```
int PE_GetSystemCount(CPE_HANDLE peHandle)
```

Parameters

[in] peHandle:	Constant handle to the system

Return value

Return the number of systems available in the configuration file.

4.4. PE_GetSystemName

Retrieve the name of the system located at the specified `index` in the list of available systems. This name identifies uniquely a system and can be used to connect to the system.

```
PE_STATUS PE_GetSystemName(CPE_HANDLE peHandle,
                           int index,
                           char *name,
                           int size)
```

Parameters

[in] peHandle:	Constant handle to the system
[in] index:	Position of the system, must be lower than total system count
[out] name:	String defining uniquely a system
[in] size:	Maximum length of the given string buffer

Return value

Return `PE_SUCCESS` if successful, otherwise the return value is a `PE_STATUS` status code.

4.5. PE_GetLibraryVersion

Get the version number of the library currently in use. The version number is formatted following this formula: `(major << 16) + (minor << 8) + bugfix`. For example, if the library version is 1.8.12, this function will return an integer with the following hexadecimal value: `0x01080C`.

```
int PE_GetLibraryVersion()
```

Parameters

None.

Return value

Library version number encoded as an integer.

4.6. PE_GetStatusStr

Translate the value of `code` and return a textual description of the status. The description is a short explanation of the status code without contextual information.

```
const char* PE_GetStatusStr(PE_STATUS code)
```

Parameters

`[in] code:` Status code to translate

Return value

A pointer to the string describing the status.

5. LLTF Contrast Control Functions

PHySpec SDK provides functions that give a low-level control of your system. In order to open a connection to a system, you must have its exact name from the configuration file. See **PHySpec Management Functions** for more details on this topic. PHySpec SDK supports simultaneous connections to different systems, but you may not open a connection to the same system more than once. All functions are not thread-safe, which means that the system will have an undefined behaviour if accessed by multiple threads. If multi-threading is desired, each call to a function must be protected with an appropriate mechanism such as a mutex.

5.1. PE_CLOSE

Close the communication channel and reset the system to its default state.

```
PE_STATUS PE_Close(PE_HANDLE peHandle)
```

Parameters

`[in] peHandle:` Handle to the system

Return value

Return `PE_SUCCESS` if successful, otherwise the return value is a `PE_STATUS` status code.

5.2. PE_OPEN

Open a communication channel between the computer and the specified system. The system always opens in a default state.

```
PE_STATUS PE_Open(PE_HANDLE peHandle,
                  const char *name)
```

Parameters

`[in] peHandle:` Handle to the system

`[in] name:` String defining uniquely a system

Return value

Return `PE_SUCCESS` if successful, otherwise the return value is a `PE_STATUS` status code.

5.3. PE_GetWavelength

Retrieve the central wavelength filtered by the system in nanometres.

```
PE_STATUS PE_GetWavelength(CPE_HANDLE peHandle,
                           double *wavelength)
```

Parameters

[in] peHandle:	Constant handle to the system
[out] wavelength:	Central wavelength in nanometres

Return value

Return PE_SUCCESS if successful, otherwise the return value is a PE_STATUS status code.

5.4. PE_SetWavelength

Set the central wavelength filtered by the system in nanometres.

```
PE_STATUS PE_SetWavelength(PE_HANDLE peHandle,
                           double wavelength)
```

Parameters

[in] peHandle:	Handle to the system
[in] wavelength:	Central wavelength in nanometres

Return value

Return PE_SUCCESS if successful, otherwise the return value is a PE_STATUS status code.

5.5. PE_GetWavelengthRange

Retrieve the wavelength range of the system in nanometres.

```
PE_STATUS PE_GetWavelengthRange(CPE_HANDLE peHandle,
                                double *minimum
                                double *maximum)
```

Parameters

[in] peHandle:	Constant handle to the system
[out] minimum:	Minimum available wavelength in nanometres
[out] maximum:	Maximum available wavelength in nanometres

Return value

Return PE_SUCCESS if successful, otherwise the return value is a PE_STATUS status code.

5.6. PE_HasHarmonicFilter

Retrieve the availability of the harmonic filter accessory.

```
int PE_HasHarmonicFilter(CPE_HANDLE peHandle)
```

Parameters

[in]	peHandle:	Constant handle to the system

Return value

Return a non-zero value if the accessory is available, otherwise 0 is returned.

5.7. PE_GetHarmonicFilterEnabled

Retrieve the status of the harmonic filter accessory.

```
PE_STATUS PE_GetHarmonicFilterEnabled(CPE_HANDLE peHandle,
                                      int *enable)
```

Parameters

[in]	peHandle:	Constant handle to the system
[out]	enable:	State of the harmonic filter, 0 is disabled

Return value

Return `PE_SUCCESS` if successful, otherwise the return value is a `PE_STATUS` status code.

5.8. PE_SetHarmonicFilterEnabled

Set the status of the harmonic filter accessory.

```
PE_STATUS PE_SetHarmonicFilterEnabled(PE_HANDLE peHandle,
                                      int enable)
```

Parameters

[in]	peHandle:	Handle to the system
[in]	enable:	State of the harmonic filter, 0 is disabled

Return value

Return `PE_SUCCESS` if successful, otherwise the return value is a `PE_STATUS` status code.

5.9. PE_GETGRATINGCOUNT

Retrieve the system's grating number.

```
PE_STATUS PE_getGratingCount (CPE_HANDLE peHandle,
                              int *count)
```

Parameters

[in]	peHandle:	Handle to the system
[out]	count:	Grating count

Return value

Return PE_SUCCESS if successful, otherwise the return value is a PE_STATUS status code.

5.10. PE_GETGRATINGNAME

Retrieve the name of a grating.

```
PE_STATUS PE_GetGratingName (CPE_HANDLE peHandle,
                             int gratingIndex
                             char *name
                             int size)
```

Parameters

[in]	peHandle:	Constant handle to the system
[in]	gratingIndex:	Position of the grating, must be lower than total grating count (zero-based)
[out]	name:	String defining uniquely a grating
[in]	size:	Maximum length of the given string buffer

Return value

Return PE_SUCCESS if successful, otherwise the return value is a PE_STATUS status code.

5.11. PE_GetGratingWavelengthRange

Retrieve the wavelength range of a grating in nanometres.

```
PE_STATUS PE_GetGratingWavelengthRange (CPE_HANDLE peHandle,
                                        int gratingIndex
                                        double *minimum
                                        double *maximum)
```

Parameters

[in]	peHandle:	Constant handle to the system
[in]	gratingIndex:	Position of the grating, must be lower than total grating count (zero-based)
[out]	minimum:	Minimum available wavelength in nanometres
[out]	maximum:	Maximum available wavelength in nanometres

Return value

Return PE_SUCCESS if successful, otherwise the return value is a PE_STATUS status code.

5.12. PE_GetGratingWavelengthExtendedRange

Retrieve the extended wavelength range of a grating in nanometres.

```
PE_STATUS PE_GetGratingWavelengthExtendedRange (CPE_HANDLE peHandle,
                                                int gratingIndex
                                                double *extMinimum
                                                double *extMaximum)
```

Parameters

[in]	peHandle:	Constant handle to the system
[in]	gratingIndex:	Position of the grating, must be lower than total grating count (zero-based)
[out]	extMinimum:	Minimum available wavelength in nanometres
[in]	extMaximum:	Maximum available wavelength in nanometres

Return value

Return PE_SUCCESS if successful, otherwise the return value is a PE_STATUS status code.

5.13. PE_SetWavelengthOnGrating

Set the central wavelength filtered by the system with the specified grating in nanometres.

```
PE_STATUS PE_SetWavelengthOnGrating (PE_HANDLE peHandle,
                                     int gratingIndex
                                     double wavelength)
```

Parameters

[in]	peHandle:	Constant handle to the system
[in]	gratingIndex:	Position of the grating, must be lower than total grating count (zero-based)
[in]	wavelength:	Central wavelength in nanometres

Return value

Return PE_SUCCESS if successful, otherwise the return value is a PE_STATUS status code.

5.14. PE_GetGrating

Retrieve the grating currently used.

```
PE_STATUS PE_GetGrating (PE_HANDLE peHandle,
                         int *gratingIndex)
```

Parameters

[in]	peHandle:	Constant handle to the system
[out]	gratingIndex:	Position of the grating, must be lower than total grating count (zero-based)

Return value

Return PE_SUCCESS if successful, otherwise the return value is a PE_STATUS status code.

6. Status Codes

The table below describes all status that can be returned from a function. Checks for a specific status code should be done by its qualified name instead of its value, because status values may change in future version of the library.

Code	Name	Description
0	PE_SUCCESS	Function executed successfully.
1	PE_INVALID_HANDLE	Supplied handle is corrupted or has a NULL value.
2	PE_FAILURE	Communication with system failed.
3	PE_MISSING_CONFIGFILE	Configuration file is missing
4	PE_INVALID_CONFIGURATION	Configuration file is corrupted
5	PE_INVALID_WAVELENGTH	Requested wavelength is out of bound.
6	PE_MISSING_HARMONIC_FILTER	Nor harmonic filter present in the system configuration
7	PE_INVALID_FILTER	Requested filter does not match any available.
8	PE_UNKNOWN	An unknown status code has been returned by the system.
9	PE_INVALID_GRATING	Requested grating does not match any available.
10	PE_INVALID_BUFFER	Output buffer has a NULL value.
11	PE_INVALID_BUFFER_SIZE	Output buffer size is too small to receive the value.
12	PE_UNSUPPORTED_CONFIGURATION	The system configuration is not supported by this SDK.
13	PE_NO_FILTER_CONNECTED	No filter connected.

7. Appendix A – Code Example

The example below is intended to be a walkthrough demonstrating basic functionality of PHySpec SDK. This example only moves the first system available to an arbitrary wavelength and then immediately exits. Keep in mind that this example is not complete because it skips on error checking.

The first step is to create the environment to access all the library functionalities. To do so, call `PE_Create` with the appropriate configuration file for your system. This file is usually located in the PHySpec installation directory (`C:\Program Files (x86)\Photon etc\PHySpecV2`) and is named `system.xml`. All the information describing your system is contained in this file. Be aware that unwarranted modification to this file can cause the system to stop working, but cannot cause physical damage.

The next step is to connect to your system by calling `PE_Open`. The name argument needed can be found in the configuration file or by enumerating the systems available for your configuration. To enumerate the systems names, call `PE_GetSystemCount` to get the number of systems. Then iterate through those systems and retrieve their name with `PE_GetSystemName`.

After connecting to your system, you can control its central filtering wavelength with `PE_SetWavelength`. Other functions are also provided to describe or control the state of the system.

Before exiting the program, you should cleanup resources in use by calling `PE_Close` to disconnect the system and then `PE_Destroy` to release the PHySpec environment. Failure to call these functions may result in your system becoming unresponsive at your next attempt to connect.

```
#include <PE_Filter.h>

int main(int argc, char *argv[])
{
	PE_HANDLE handle = NULL;
	char systemName[256];

	/* Please note that error checks are omitted for clarity reasons */
	PE_Create("system.xml", &handle);

	/* Retrieve the first system name available */
	PE_GetSystemName(handle, 0, systemName, sizeof(systemName));

	/* Connect to the system found */
	PE_Open(handle, systemName);

	/* Select an arbitrary wavelength */
	PE_SetWavelength(handle, 650.0);

	/* Cleanup resource */
	PE_Close(handle);
	PE_Destroy(handle);
}
```

8. Appendix B – Revision History

The table below describes all the changes made to this document.

Version	Description	Date
1.0.0	Initial documentation release.	2014-02-21
1.2.0	Add grating's control functions.	2016-02-15