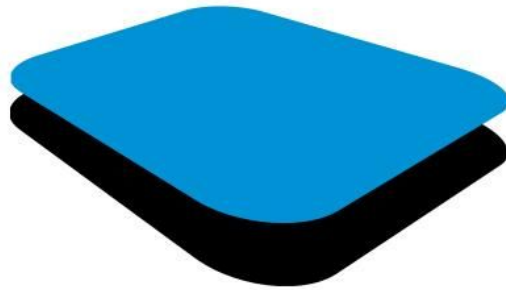


Lyncée tec^{DHM}



Lyncée tec

Koala Remote

Users Manual

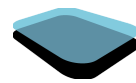
Version 8.0.x - February 2019

Versions and modifications of this manual

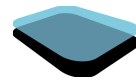
Version	Date	Main modifications
8.0.x	March 2019	First version of this manual

Table of contents

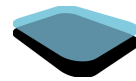
Versions and modifications of this manual	1
Table of contents	1
Introduction	4
Koala side (server side)	4
Client side	4
How to implement your own remote client application	5
References	5
Implementation	5
Error handling	6
The remote return codes	6
Troubleshooting your application with the logs	7
The Koala log	7
Adding a log4net log in your client application	7
How to update your client application to make it compatible with Koala 8.0	8
Update the reference libraries used by you application	8
Update the references to the Lyncée Tec library in your code	8
The remote functions	10
AccWDSearch	10
Acquisition2L	11
AddCorrSegment	12
AlgoResetPhaseMask	13
AxisInstalled	14
ComputePhaseCorrection	15
Connect	16
DigitizerAcquiring	17



ExtractPhaseProfile	18
ExtUIGoodBye	19
FastWDSearch	20
GetAxesPosMu	21
GetCameraShutterUs	22
GetDHMSerial	23
GetHoloContrast	24
GetHoloHeight	25
GetHoloImage	26
GetHoloWidth	27
GetIntensity32fImage	28
GetIntensityImage	29
GetLambdaNm	30
GetOPLPos	31
GetPhase32fImage	32
GetPhaseHeight	33
GetPhaseImage	34
GetPhaseProfile	35
GetPhaseProfileLength	36
GetPhaseWidth	37
GetPxSizeUm	38
GetRecDistCM	39
GetUnwrap2DState	40
InitXYZStage	41
KoalaShutDown	42
LoadHolo	43
Login	44
Logout	45
MoveAxes	46
MoveAxesArr	47
MoveAxis	48
MoveOPL	49
OnDistanceChange	50
OpenConfig	51
OpenFrmTopography	52
OpenHoloWin	53
OpenIntensityWin	54
OpenPhaseWin	55
RecordStroboFixFrequency	56



RecordStroboFrequencyScan	57
ResetCorrSegment	58
ResetGrab	59
SaveImageFloatToFile	60
SaveImageToFile	61
SelectDisplayWL	62
SelectTopoZone	63
SetCameraShutterUs	64
SetPhaseProfileState	65
SetRecDistCM	66
SetSourceState	67
SetUnwrap2DMethod	68
SetUnwrap2DState	69
SingleReconstruction	70
Former functions which were removed	71
CloseConfig	71
GetMaxHoloContrast	72
GetRaNm	73
SaveHolo2L	74
SetSurfLambdaCNm	75
SetSurfLambdaSNm	76
Appendix	77
Python remote client application example	77



Introduction

The *Remote Module* (also known as *Production Mode*) allows to send commands to the Koala interface from a separate user-defined client program, executing at the same time as Koala.

The remote module, is optional, and is usually sold separately.

To use the remote module, it must have been enabled in the factory settings of your system, and the remote option must have been selected at installation. If you did not purchase the module, it is always possible to buy and activate this option at a later point.

The remote commands communication takes place over a TCP-IP connection.

Koala side (server side)

The Koala software acts as a server for the remote commands.

For more details on the *Remote Module* from the Koala side, please see the *Controlling Koala via the Remote interface* section in the *Koala Manual*

Client side

There are two options for the client-side, either use the *Koala Remote Test App* (see the specific manual), or implement your own client. See section [How to implement your own remote client application](#) below.

How to implement your own remote client application

References

Lyncée Tec provides .NET libraries to help you implement your own client without the need to implement the details of the TCP-IP communication protocol.

The necessary libraries and configuration files are located in *C:\Program Files\LynceeTec\Koala\Remote\RemoteLibraires* (make sure you take the correct processor version, x64 or x86, according to your system and application).

File name	Description
<i>log4net.dll</i>	3 rd party library for logging. See the Troubleshooting your application with the logs section below.
<i>LynceeTec.KoalaRemote.Client.dll</i>	Utility for the implementation of a remote client application. This is the main library that you will have to reference in your code to implement your client application. See the Implementation section below.
<i>LynceeTec.KoalaRemote.dll</i>	Base definitions for the remote protocol
<i>LynceeTec.ToolBox.dll</i>	Utility library
<i>messages.xml</i>	Signed list of remote commands

The reference libraries and files must all be copied next to your application executable even if you do not directly refer to them in your code.

The Lyncée Tec reference libraries version number must always be the same as the version number of your Koala program.

Implementation

Your application needs to create an instance of the `LynceeTec.KoalaRemote.Client.KoalaRemoteClient` class which is defined in the *LynceeTec.KoalaRemote.Client.dll* library, using the parameterless constructor as follows:

```
KoalaRemoteClient m_client = new KoalaRemoteClient();
```

Through this instance you can call all the remote functions described in the [The remote functions](#) section of this manual.

Your application must first call the *Connect* function, followed by the *Login* function.

```
string userName = "";  
bool success = m_client.Connect("localhost", out userName, true);  
success = m_client.Login(password);
```

You can then call any remote function.

```
m_client.OpenConfig(137);
```

When closing, your application should call *Logout*, to notify Koala that it is disconnecting.

```
m_client.Logout();
```

For more details, please also refer to the provided examples for *Python*, *LabView* and *Matlab*. You can find a copy of the Python example in the [Python remote client application example](#) section of this manual.

Error handling

In case of an error due to a remote command, the command function throws an exception.

If the error is directly linked to the remote command, the exception will be of type `LynceeTec.KoalaRemote.Client.KoalaRemoteFunctionException` and will contain the following information:

- `ErrorCode (RemoteReturnCodes)` : The remote command return code.
- `ErrorMessage (String)`: The error message
- `FunctionName (String)`: The name of the remote command which failed

The remote return codes

The remote return codes are defined in the `LynceeTec.KoalaRemote.RemoteReturnCodes` enumeration in the *LynceeTec.KoalaRemote.dll*. The following table describes the possible values:

Error code	Error Name	Definition
-1000	<i>Unknown</i>	Unknown result. Should not happen. If this happens, please contact your Lyncee Tec support.
-6	<i>SystemBusy</i>	The command could not be executed because another one is executing
-5	<i>ExecutionFailed</i>	The command started execution, but an error occurred

-4	<i>MessageNotFound</i>	The command sent a message unknown in the remote protocol. Should not happen if you use the libraries corresponding to your current version of Koala
-3	<i>InvalidOperation</i>	The command was not executed because Koala is not in a state where it's execution was possible
-2	<i>TargetHandlerNotFound</i>	Deprecated. Handler object necessary for the command execution does not exist.
-1	<i>TargetObjectNotFound</i>	Deprecated. Object necessary for the command execution does not exist.
0	<i>OK</i>	The command executed successfully.
1	<i>LongAsyncOperation</i>	Deprecated. The command was not executed because a previous asynchronous command is still executing. In the current version, every remote command is synchronous and blocking until execution is finished.

Obviously, no exception is thrown when the result is *OK*.

In a standard situation, your code should only receive `KoalaRemoteFunctionException` with return codes of value *SystemBusy*, *ExecutionFailed* and *InvalidOperation*.

Troubleshooting your application with the logs

The Koala log

The Koala log can be very helpful to debug your remote client application and understand what happened in case of an error.

The Koala log is located in `C:\Users\{userName}\AppData\Local\LynceeTec\Koala`, where `{userName}` is your Windows login name. It can also directly be displayed in Koala, via the *Help* → *Show Log* menu.

Adding a log4net log in your client application

All Lyncée Tec libraries use *log4net* for logging. We recommend that you implement logging with *log4net* in your client application as well. By doing that, your client application log will automatically contain the log entries from the *LynceeTec.KoalaRemote.Client.dll* and *LynceeTec.KoalaRemote.dll* reference libraries.

The *log4net* library is included in the set of Lyncée Tec remote reference libraries (see the [References](#) section above). It is also available as a standard NuGet package on [nuget.org](https://www.nuget.org).

How to update your client application to make it compatible with Koala 8.0

Some libraries have been renamed in the newly released Koala version 8.0 and additional libraries are now mandatory.

To update your application, you need to update the reference libraries used by your application and rename their reference in your code.

Update the reference libraries used by you application

In your application reference folder, remove the *KoalaRemoteBase.dll*, *TCPClient.dll* and *message.xml* files.

Replace them by the new references located in *C:\Program Files\LynceeTec\Koala\Remote\RemoteLibraires* (make sure you take the correct processor version, x64 or x86, according to your system and application):

- *log4net.dll* (version 2.0.8.0)
- *LynceeTec.KoalaRemote.Client.dll* (same version as your current Koala)
- *LynceeTec.KoalaRemote.dll* (same version as your current Koala)
- *LynceeTec.ToolBox.dll* (same version as your current Koala)
- *messages.xml*

Update the references to the Lyncee Tec library in your code

Modify your code references according to the following table:

	Old	New (version 8.0)
Library name	TCPClient.dll	LynceeTec.KoalaRemote.Client.dll
namespace	KoalaClient	LynceeTec.KoalaRemote.Client
Class name	KoalaTCPClient	KoalaRemoteClient

Example in Python:

Old code	New code
<code>import clr</code>	<code>import clr</code>
<code>clr.AddReference("TCPClient")</code>	<code>clr.AddReference("LynceeTec.KoalaRemote.Client")</code>
<code>import KoalaClient</code>	<code>from LynceeTec.KoalaRemote.Client import KoalaRemoteClient</code>
<code>host = KoalaClient.KoalaTCPClient()</code>	<code>host = KoalaRemoteClient()</code>
<code>host.OpenConfig(configNumber)</code>	<code>host.OpenConfig(configNumber)</code>

The remote command interfaces have not changed, the rest of your code should remain compatible.

The remote functions

AccWDSearch

Performs an accurate working distance search.

Note that this function is blocking, like all remote functions, and might take several minutes, depending on the chosen parameters.

Koala UI equivalent:

Accurate working distance search in the *Working Distance window*. Open the working distance window, via the *Tools* → *Working Distance* menu.

Parameters:

- distUM (Int32): Full range of the search, in [um].
- stepUM (Double): Size of a step between two working distance measurements, in [um].

Return: void (nothing)

Requirements:

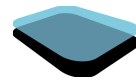
- The stage must be available and the axes must be initialized
- A configuration must be loaded
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation*

See also:

- [FastWDSearch](#)



Acquisition2L

Acquires an image (or several if temporal averaging is on) and reconstruct it.

Koala UI equivalent:

(No exact Koala UI equivalent)

(No parameters)

Return: void (nothing)

Requirements:

- A configuration must be loaded
- Live mode must be available
- The intern camera must be available
- Video mode must not be running
- A strobo measurement must not be running
- The sequence reconstruction to display window must not be opened
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation*

See also:


- [SingleReconstruction](#)

AddCorrSegment

Adds a phase correction segment (for 1D tilt correction)

Note: The start and end points can be outside the phase ROI, but it is recommended to avoid it.

Koala UI equivalent:

Tracing a phase correction segment in the *Phase Image window*, when the *Horizontal and Vertical profiles for phase mask adjustment* button  is activated.

Parameters:

- top (Int32): Y coordinate of the top left point of the segment, in pixel
- left (Int32): X coordinate of the top left point of the segment, in pixel
- length (Int32): Length of the segment, in pixel
- Orientation (Int32): Orientation of the segment: 0 = horizontal, 1 = vertical

Return: void (nothing)

Requirements:

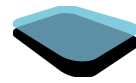
- A configuration must be loaded
- The phase window must be opened and must contain an image
- A reconstruction object must be created
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements \Rightarrow *Invalid Operation*

See also:

- [AlgoResetPhaseMask](#)
- [ComputePhaseCorrection](#)
- [ResetCorrSegment](#)



AlgoResetPhaseMask

Resets the current user masks for the phase tilt correction to the values saved in the database.

Koala UI equivalent:

Click on the *Reset mask* button in the *Basic settings* tab of the *Reconstruction settings window*.

(No parameters)

Return: void (nothing)

Requirements:

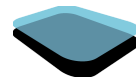
- A configuration must be loaded
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements \Rightarrow *Invalid Operation*

See also:

- [AddCorrSegment](#)
- [ComputePhaseCorrection](#)
- [ResetCorrSegment](#)



AxisInstalled

Gets a value indicating if an axis is installed for the current stage.

Koala UI equivalent:

(Not applicable)

Parameters:

- axisId (Int32): Id of the axis to check. Possible values are
 - 0: X axis
 - 1: Y axis
 - 2: Z axis
 - 3: Theta axis (rotation, system-dependent)
 - 4: Phi axis (rotation, system-dependent)
 - 5: Psi axis (rotation, system-dependent)

Return: Boolean

Requirements:

- A stage must be defined
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation*

See also:

- -

ComputePhaseCorrection

Computes the 1D phase correction using a specific method.

Note that if the *Reconstruction Settings* window is not opened, the function will open it automatically.

Koala UI equivalent:

Click on the *Perform fit* button in the *Basic settings* tab of the *Reconstruction settings window*.

Parameters:

- `fitMethod` (Int32): Method used to compute the phase correction. Possible values are:
 - 0: Tilt. In this case, *degree* must be equal to 1
 - 1: Polynomial.
- `degree` (Int32): Degree of the correction. Value must be a positive non-zero integer and can only be 1 *fitMethod* is *Tilt*.

Return: void (nothing)

Requirements:

- A configuration must be loaded
- A reconstruction object must be created
- The phase window must be opened and must contain an image
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation*
- Incoherent parameters ⇒ *Invalid Operation*

See also:

- [AddCorrSegment](#)
- [AlgoResetPhaseMask](#)
- [ResetCorrSegment](#)

Connect

Connects the client remote application to the Koala remote server.

Note that connection and login is handled automatically in the *Koala Remote Test App*. It is only needed if you implement your own client application.

Koala UI equivalent:

(not applicable)

Parameters:

- `hostName` (String): The IP of the computer where Koala is running. Use `localhost` if running on the same computer
- `userName` (String): Output parameter returning the name of the user currently logged in Koala
- `quiet` (Boolean): Deprecated parameter, will be removed in a later version. Set either to `true` or `false`

Return: `true` if the connection was successful.

Requirements:

- A user must be logged in

Possible errors:

- The *Remote Log Message window* is not opened or needs to be reopened because of a previous login failure

See also:

- [Login](#)
- [Logout](#)



DigitizerAcquiring

Gets a value indicating if the camera is currently in continuous grab mode or not.

Koala UI equivalent:

(not applicable)

(No parameters)

Return: Boolean: `true` if the camera is in continuous grab mode

Requirements:

- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements \Rightarrow *Invalid Operation*

See also:


- [ResetGrab](#)

ExtractPhaseProfile

Extracts a profile from the phase image and plots it in the phase profile window.

Note: The start and end points can be outside the phase ROI, but it is recommended to avoid it.

Koala UI equivalent:

Tracing a profile in the *Phase Image window*, when the *Draw a profile line* button  is activated.

Parameters:

- startX (Int32): X coordinate of the starting point of the profile in the phase ROI
- startY (Int32): Y coordinate of the starting point of the profile in the phase ROI
- endX (Int32): X coordinate of the ending point of the profile in the phase ROI
- endY (Int32): Y coordinate of the ending point of the profile in the phase ROI

Return: void (nothing)

Example: (See example for [GetPhaseProfile](#))

Requirements:

- A configuration must be loaded
- The phase window must be opened and must contain an image
- The phase profile window must be opened
- A reconstruction object must be created
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements \Rightarrow *Invalid Operation*

See also:

- [GetPhaseProfile](#)
- [GetPhaseProfileLength](#)
- [SetPhaseProfileState](#)

ExtUIGoodBye

This function is documented for backward compatibility, but it should be replaced by Logout. This function notifies the server side that a client application is closing, and it should close the connection.

This function is not blocking and will return before execution is finished without waiting for an answer.

After a call to *ExtUIGoodBye*, it is possible to call *Connect* and then *Login* again, without the need to restart the *Remote Message Log window* in Koala.

Note that disconnection is handled automatically in the *Koala Remote Test App*. It is only needed if you implement your own client application.

Koala UI equivalent:

(not applicable)

(No parameters)

Return: void (nothing)

Requirements:

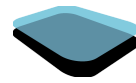
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation*

See also:

- [Connect](#)
- [Login](#)
- [Logout](#)



FastWDSearch

Performs a fast working distance search.

Note that this function is blocking, like all remote functions, and might take several seconds.

Koala UI equivalent:

Fast working distance search in the *Working Distance window*. Open the working distance window, via the *Tools* → *Working Distance* menu.

(no parameters)

Return: void (nothing)

Requirements:

- The stage must be available and the axes must be initialized
- A configuration must be loaded
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation*

See also:

- [AccWDSearch](#)

GetAxesPosMu

Copies the current positions of the stage axes, in [um], in the given memory location.

Koala UI equivalent:

(not applicable)

Parameters:

- `buffer (Double[])`: Array of doubles of size 4, where the data for the X, Y, Z and Theta axis respectively will be copied

Return: void (nothing)

Requirements:

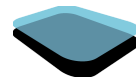
- A stage must be available and initialized
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements \Rightarrow *Invalid Operation* error
- The buffer array does not have the correct dimension \Rightarrow this will not result in an error on the server (Koala) side, but on the client application.

See also:


- [MoveAxis](#)
- [MoveAxes](#)
- [MoveAxesArr](#)



GetCameraShutterUs

Get the value of the camera shutter parameter, in [us].

Koala UI equivalent:

(Not applicable, but the current shutter value is displayed in the *Camera Settings window*, accessible via the *Settings / Camera* menu, or the *Camera configurations* button )

(No parameters)

Return: Int32: The shutter value, in [us]

Requirements:

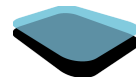
- The intern camera must be available
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

See also:

- [SetCameraShutterUs](#)



GetDHMSerial

Gets the numerical part of the serial number of the DHM device.

Koala UI equivalent:

(Not applicable, but the full serial number is displayed at the bottom right of the Koala main window, in the status bar.)

(No parameters)

Return: Int32

Requirements:

- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error


See also:

- -

GetHoloContrast

Gets a value representing the contrast of the last hologram (either last one grabbed or last loaded from disk, whichever occurred last) in an area half the size of the hologram, centered in the middle of the image.

Koala UI equivalent:

(Not applicable, but the hologram contrast is displayed in the *Hologram Histogram window*, which can be opened by clicking the *Draw a hologram*  button in the *Hologram Image window*.)

(No parameters)

Return: Double

Requirements:

- A configuration must be loaded
- An hologram must have been grabbed since program start
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements \Rightarrow *Invalid Operation* error

See also:

- -



GetHoloHeight

Gets the height of the hologram image, according to the current configuration.

Koala UI equivalent:

(not applicable)

(No parameters)

Return: Int32

Requirements:

- A configuration must be loaded
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

See also:

- [GetHoloWidth](#)

GetHoloImage

Copies the current hologram image in the given memory location

Koala UI equivalent:

(not applicable)

Parameters:

- `buffer (Byte[])`: Array of bytes of dimension `stride*height` where the data will be copied. (See the [GetPhaseImage](#) function for an example of how to compute the stride)
- `lambdald (Int32)`: Logical ID of the source from which the hologram must be saved (for alternate dual wavelength configurations). Optional, default value is 0.

Return: void (nothing)

Requirements:

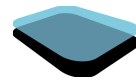
- A configuration must be loaded
- The hologram window must be opened and must contain an image
- A reconstruction object must be created
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements \Rightarrow *Invalid Operation* error

See also:

- [OpenHoloWin](#)



GetHoloWidth

Gets the width of the hologram image, according to the current configuration.

Koala UI equivalent:

(not applicable)

(No parameters)

Return: Int32

Requirements:

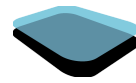
- A configuration must be loaded
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

See also:

- [GetHoloHeight](#)



GetIntensity32fImage

Copies the current intensity (amplitude) image in the given memory location, as a floating point image (32 bits)

Koala UI equivalent:

(not applicable)

Parameters:

- `buffer (Single[])`: Array of singles of dimension `stride*height` where the data will be copied

Return: void (nothing)

Requirements:

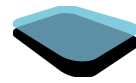
- A configuration must be loaded
- The intensity (amplitude) window must be opened and must contain an image
- A reconstruction object must be created
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements \Rightarrow *Invalid Operation* error

See also:

- [GetIntensityImage](#)
- [OpenIntensityWin](#)



GetIntensityImage

Copies the current intensity (amplitude) image in the given memory location, as a grayscale image (8 bits)

Koala UI equivalent:

(not applicable)

Parameters:

- `buffer (Byte[])`: Array of bytes of dimension `stride*height` where the data will be copied

Return: void (nothing)

Requirements:

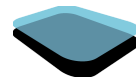
- A configuration must be loaded
- The intensity (amplitude) window must be opened and must contain an image
- A reconstruction object must be created
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements \Rightarrow *Invalid Operation* error

See also:

- [GetIntensity32fImage](#)
- [OpenIntensityWin](#)



GetLambdaNm

Gets the wavelength of a laser source, in [nm].

Koala UI equivalent:

(Not applicable, but the wavelength of the sources is displayed in the *Sources* tab of the *About your DHM* window, which can be opened via the *Help* → *About your DHM* menu.)

Parameters:

- `srcId` (Int32): The id of the source (logical or physical according to *useLogicalId*)
- `useLogicalId` (Boolean): Set to `true` to use logical id, to `false` to use physical id. Optional, default value is `true`.

Return: Single: The wavelength of the source, in [nm]

Requirements:

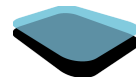
- If logical ids are used, a configuration must be loaded
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation*

See also:

- -



GetOPLPos

Gets the non-corrected position of the OPL motor in [qc].

Koala UI equivalent:

(not applicable)

(No parameters)

Return: Int32: The position of the OPL motor, in [qc]

Requirements:

- The OPL option must be enabled
- The OPL motor must be available and initialized
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation*

See also:

- [MoveOPL](#)

GetPhase32fImage

Copies the current phase image in the given memory location, as a floating point image (32 bits)

Koala UI equivalent:

(not applicable)

Parameters:

- `buffer (Single[])`: Array of singles of dimension `stride*height` where the data will be copied

Return: void (nothing)

Example: (See example for *GetPhaseImage*)

Requirements:

- A configuration must be loaded
- The phase window must be opened and must contain an image
- A reconstruction object must be created
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements \Rightarrow *Invalid Operation* error

See also:

- [GetPhaseImage](#)
- [OpenPhaseWin](#)



GetPhaseHeight

Gets the height of the phase image ROI, according to the current configuration.

Note that the intensity (amplitude) image has the same dimensions.

Koala UI equivalent:

(not applicable)

(No parameters)

Return: Int32

Requirements:

- A configuration must be loaded
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements \Rightarrow *Invalid Operation* error

See also:

- [GetPhaseWidth](#)
- [OpenConfig](#)



GetPhaseImage

Copies the current phase image in the given memory location, as a grayscale image (8 bits)

Koala UI equivalent:

(not applicable)

Parameters:

- **buffer (*Byte[]*)**: Array of bytes of dimension `stride*height` where the data will be copied

Return: void (nothing)

Example:

```
int width = host.GetPhaseWidth();
int height = host.GetPhaseHeight();
int stride = (width / 4)*4;
if (width % 4 != 0)
    stride += 4;
byte[] buffer = new byte[stride*height];
host.GetPhaseImage(buffer);
```

Requirements:

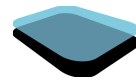
- A configuration must be loaded
- The phase window must be opened and must contain an image
- A reconstruction object must be created
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements \Rightarrow *Invalid Operation* error

See also:

- [GetPhase32fImage](#)
- [OpenPhaseWin](#)



GetPhaseProfile

Copies the current phase profile data in the given memory location

Note: the length of the array to receive the data is given by the function [GetPhaseProfileLength](#)

Koala UI equivalent:

(not applicable)

Parameters:

- buffer (Double[]): Array of doubles where the data will be copied

Return: void (nothing)

Example (C#):

```
host.SetPhaseProfileState(true);  
host.ExtractPhaseProfile(100, 100, 200, 200);  
int profileLength = host.GetPhaseProfileLength();  
double[] profileData = new double[profileLength];  
host.GetPhaseProfile(profileData);
```

Requirements:

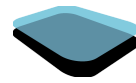
- A configuration must be loaded
- A reconstruction object must be created
- A phase profile must be selected
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

See also:

- [ExtractPhaseProfile](#)
- [GetPhaseProfileLength](#)
- [SetPhaseProfileState](#)



GetPhaseProfileLength

Gets the length of the current phase profile array.

Note: this function is needed to know the length of the array to give to [GetPhaseProfile](#).

Koala UI equivalent:

(not applicable)

(No parameters)

Return: Int32

Example: (See example for [GetPhaseProfile](#))

Requirements:

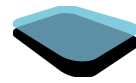
- A configuration must be loaded
- A reconstruction object must be created
- A phase profile must be selected
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

See also:

- [ExtractPhaseProfile](#)
- [GetPhaseProfile](#)
- [SetPhaseProfileState](#)



GetPhaseWidth

Gets the width of the phase image ROI, according to the current configuration.

Note that the intensity (amplitude) image has the same dimensions.

Koala UI equivalent:

(not applicable)

(No parameters)

Return: Int32

Requirements:

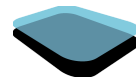
- A configuration must be loaded
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements \Rightarrow *Invalid Operation* error

See also:

- [GetPhaseHeight](#)



GetPxSizeUm

Returns the calibrated size, in [um], represented by a pixel in the image.

Note that this value is calibrated and configuration-dependent, because it depends on the objective used for this configuration.

Koala UI equivalent:

(not applicable)

(No parameters)

Return: Single: the calibrated size of a pixel, averaged between X and Y, in [um]

Requirements:

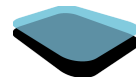
- A configuration must be loaded
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements \Rightarrow *Invalid Operation* error

See also:

- -



GetRecDistCM

Gets the current reconstruction distance, in [cm], of the active user processing configuration.

Koala UI equivalent:

(Not applicable, but the reconstruction distance is displayed in the *Focus* field of the *Reconstruction settings window*, which can be opened via the *Settings* → *Reconstruction* menu.)

(No parameters)

Return: Single: The reconstruction distance, in [cm]

Example: (See example for [OnDistanceChange](#))

Requirements:

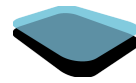
- A configuration must be loaded
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

See also:


- [OnDistanceChange](#)
- [SetRecDistCM](#)



GetUnwrap2DState

Gets a value indicating if the unwrapping of the phase image is enabled or not.

Koala UI equivalent:

(Not applicable, but the unwrapping is enabled if the *Unwrap*  button is activated in the *Phase Image window*.)

(No parameters)

Return: Boolean

Requirements:

- A configuration must be loaded
- A reconstruction object must be created
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements \Rightarrow *Invalid Operation* error

See also:

- [SetUnwrap2DState](#)
- [SetUnwrap2DMethod](#)



InitXYZStage

Moves the axes of the XYZ stage to their minimal position.

Please use with caution as moving to the minimal positions (X=0, Y=0, Z=0) may damage your stage if it is not correctly calibrated.

Koala UI equivalent:

(not applicable)

Parameters:

- `withProgressBar` (Boolean): If set to `true`, the progress bar will be displayed. (Note that progress tracking might not be implemented depending of your type of stage. In this case the progress bar will simply remain on 0.) Optional, default value is `false`.
- `moveToCenter` (Boolean): If set to `true`, will move the stage to the center of each axis range afterwards. Optional, default value is `false`.

Return: Boolean: returns `true` if the operation was successful.

Requirements:

- A stage must be available and initialized
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements \Rightarrow *Invalid Operation* error

See also:

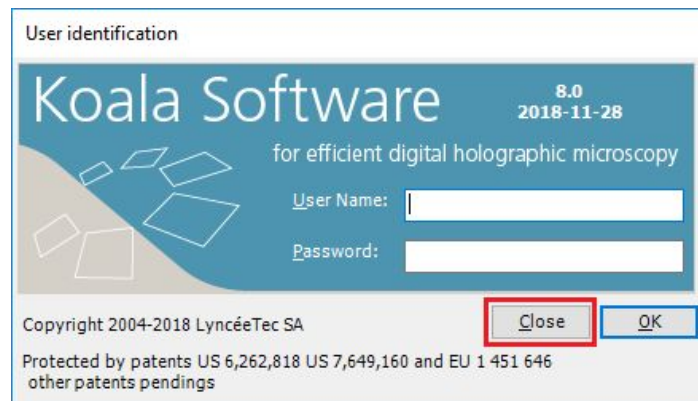
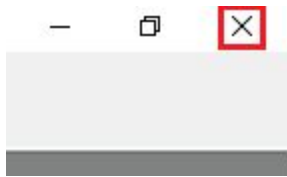
- -

KoalaShutDown

Closes Koala. This function is not blocking and will return before execution is finished.

Koala UI equivalent:

Click on the closing button at the top right of the Koala main window or clicking on the *Close* button in the login window of Koala.



Parameters:

- **confirm** (Boolean): `true` to display the dialog to ask for confirmation, `false` to close without the need for user input. Optional, default value is `false`.

Return: void (nothing)

Requirements: (nothing)

Possible errors: (nothing)

See also:



- -

LoadHolo

Loads an hologram from the disk. In order to get a correct phase and intensity reconstruction of the hologram, it should be loaded with the configuration that was used to record it. If the hologram window is not opened yet, it will be opened automatically.

Note that this function **does not support multi-holograms files anymore**. (Those files were created from alternate dual wavelengths configurations, which are deprecated.)

Koala UI equivalent:

Clicking on the *Open multi-hologram*  button or *Open hologram*  button (depending of your configuration) in the main toolbar and opening an hologram file. Or via the *Mode* → *Hologram 2-lambdas* or *Mode* → *Hologram* menu (depending of your configuration).

Parameters:

- Path (String): Full path of the hologram file. It is recommended to only work with files in .tif format.
- numLambda (Int32): The number of wavelengths of the configuration with which the hologram was taken.

Return: void (nothing)

Requirements:

- A user must be logged in
- A configuration must be loaded
- The hologram parameters such as height, width and number of bits per pixel must correspond to the parameters of the current configuration

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

See also:

- [SaveImageFloatToFile](#)
- [SaveImageToFile](#)

Login

Login for the remote client.

Note that connection and login is handled automatically in the *Koala Remote Test App*. It is only needed if you implement your own client application.

Koala UI equivalent:

(not applicable)

Parameters:

- password (String): The password for the current Koala user

Return: `true` if the login was successful.

Requirements:

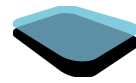
- A user must be logged in Koala
- The remote client application must be connected

Possible errors:

- The *Remote Log Message window* is not opened or needs to be reopened because of a previous login failure

See also:

- [Connect](#)
- [Logout](#)



Logout

Logout for the remote client and disconnects the TCP client without waiting for an answer before returning.

This function should be called instead of [ExtUIGoodBye](#), which is deprecated

After a *Logout*, it is possible to call *Connect* and then *Login* again, without the need to restart the *Remote Message Log window* in Koala.

Note that disconnection is handled automatically in the *Koala Remote Test App*. It is only needed if you implement your own client application.

Koala UI equivalent:

(not applicable)

(No parameters)

Return: void (nothing)

Requirements:

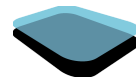
- A user must be logged in Koala
- The remote client application must be connected
- The remote client application must have logged in

Possible errors:

- The *Remote Log Message window* is not opened

See also:

- [Connect](#)
- [ExtUIGoodBye](#)
- [Login](#)



MoveAxes

Moves several axes of the stage simultaneously.

Koala UI equivalent:

Moving the axes via the *Stage XYZ window*, or moving the XYZ stage axes via the joystick.

Parameters:

- **absMove** (Boolean): Set to `true` for an absolute move, set to `false` for a relative move.
- **mvX** (Boolean): Set to `true` to move the X axis.
- **mvY** (Boolean): Set to `true` to move the Y axis.
- **mvZ** (Boolean): Set to `true` to move the Z axis.
- **mvTh** (Boolean): Set to `true` to move the Theta axis.
- **distX** (Double): Absolute position or distance to move for the X axis. In [um].
- **distY** (Double): Absolute position or distance to move for the Y axis. In [um].
- **distZ** (Double): Absolute position or distance to move for the Z axis. In [um].
- **distTh** (Double): Absolute position or distance to move for the Theta axis. In [um].
- **accX** (Double): Accuracy of the move for the X axis, in [um].
- **accY** (Double): Accuracy of the move for the Y axis, in [um].
- **accZ** (Double): Accuracy of the move for the Z axis, in [um].
- **accTh** (Double): Accuracy of the move for the Theta axis, in [um].
- **waitEnd** (Boolean): If set to `true`, the function will only return after the move is completed. Optional, default value is `true`.

Return: Boolean: `true` if the operation completed successfully.

Requirements:

- A stage must be available and initialized
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements \Rightarrow *Invalid Operation* error

See also:

- [GetAxesPosMu](#)
- [MoveAxis](#)
- [MoveAxesArr](#)

MoveAxesArr

Moves several axes of the stage simultaneously.

Koala UI equivalent:

Moving the axes via the *Stage XYZ window*, or moving the XYZ stage axes via the joystick.

Parameters:

- **axes** (Boolean[]): Array of 4 booleans for the X, Y, Z and Theta axes respectively, to indicate if the axis must be moved or not (set to `true` to move).
- **absMove** (Boolean): Set to `true` for an absolute move, set to `false` for a relative move.
- **dist** (Double[]): Array of 4 double for the X, Y, Z and Theta axes respectively, for the absolute position or distance to move for each axis. In [um].
- **acc** (Double[]): Array of 4 double for the X, Y, Z and Theta axes respectively, for the accuracy of the move for each axis. In [um].
- **waitEnd** (Boolean): If set to `true`, the function will only return after the move is completed. Optional, default value is `true`.

Return: Boolean: `true` if the operation completed successfully.

Requirements:

- A stage must be available and initialized
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error
- One or several of the argument array does not have the correct dimension ⇒ *Execution Failed* error.

See also:

- [GetAxesPosMu](#)
- [MoveAxis](#)
- [MoveAxes](#)

MoveAxis

Moves a single axis of the stage.

Koala UI equivalent:

Moving a single axis via the *Stage XYZ window*, or moving a XYZ stage axis via the joystick.

Parameters:

- `axisId` (Int32): Id of the axis to move. Possible values are
 - 0: X axis
 - 1: Y axis
 - 2: Z axis
 - 3: Theta axis (rotation, system-dependent)
- `absMove` (Boolean): Set to `true` for an absolute move, set to `false` for a relative move.
- `distUM` (Double): Absolute position or distance to move. In [um].
- `accuracyUM` (Double): Accuracy of the move, in [um]. This parameter is ignored and will be removed in a future version.
- `waitEnd` (Boolean): If set to `true`, the function will only return after the move is completed. Optional, default value is `true`.

Return: Boolean: `true` if the operation completed successfully.

Requirements:

- A stage must be available and initialized
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements \Rightarrow *Invalid Operation* error

See also:

- [GetAxesPosMu](#)
- [MoveAxes](#)
- [MoveAxesArr](#)



MoveOPL

Moves the OPL motor to a specific position, in [qc].

Note that this function does not require a configuration to be loaded, but it doesn't make much sense to move the OPL before loading a configuration.

Koala UI equivalent:

(not applicable for normal users)

Parameters:

- position (Int32): The position in qc where to move the OPL. This position corresponds to the position returned by the [GetOPLPos](#) function.

Return: void (nothing)

Requirements:

- The OPL option must be enabled
- The OPL motor must be available and initialized
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error
- The target OPL position is outside the range of possible values ⇒ *Execution Failed* error

See also:

- [GetOPLPos](#)



OnDistanceChange

Applies the last modification of the reconstruction distance, which will result in the recomputation of the intensity and phase images if they are available.

Koala UI equivalent:

(not applicable, done automatically)

(No parameters)

Return: void (nothing)

Example:

```
host.SetRecDistCM(reconstructionDistance) ;  
host.OnDistanceChange() ;
```

Requirements:

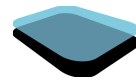
- A configuration must be loaded
- A reconstruction object must be created
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements \Rightarrow *Invalid Operation* error

See also:


- [GetRecDistCM](#)
- [SetRecDistCM](#)



OpenConfig

Opens the selected configuration.

Koala UI equivalent:

Corresponds to *File* → *Open Configuration* and then selecting a configuration and clicking on *OK*, or to clicking on the *Open configuration* button  on the toolbar and then selecting a configuration and clicking on *OK*.

Parameters:

- configNumber (*Int32*): ID of the Measurement Configuration to open

Return: void (nothing)

Requirements:

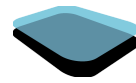
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error
- The configuration does not exist ⇒ *Execution Failed* error

See also:


- -



OpenFrmTopography

Opens the topography (roughness) window

Koala UI equivalent:

(No exact equivalent, but the *Roughness phase window (Topography window)* is opened when a topography zone is drawn on the *Phase Image window* when the *Draw a roughness area*  button is activated.)

(No parameters)

Return: void (nothing)

Requirements:

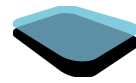
- The phase window must be opened and must contain an image
- The topography option must be enabled in the phase window
- The last selected item in the phase window must be a topography zone or topography profile
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

See also:

- [SelectTopoZone](#)



OpenHoloWin

Opens the hologram window

Koala UI equivalent:

Corresponds to clicking on the *Open Holo window* button  on the toolbar.

(No parameters)

Return: void (nothing)

Requirements:

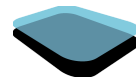
- A configuration must be loaded
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

See also:

- [OpenIntensityWin](#)
- [OpenPhaseWin](#)



OpenIntensityWin

Opens the intensity (amplitude) window

Koala UI equivalent:

Corresponds to clicking on the *Open Intensity window* button  in the toolbar.

Parameters:

- `updateXYScale` (*Boolean*): If set to `true`, the scale displayed in the *Intensity Image window* will be updated. Optional, default value is `true`.

Return: void (nothing)

Requirements:

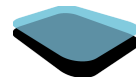
- A configuration must be loaded
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements \Rightarrow *Invalid Operation* error

See also:

- [OpenHoloWin](#)
- [OpenPhaseWin](#)



OpenPhaseWin

Opens the phase window

Koala UI equivalent:

Corresponds to clicking on the *Open Phase window* button  in the toolbar

Parameters:

- `withoutColorbar` (*Boolean*): If set to `true`, the phase window will not have a color bar. Optional, default value is `false`.
- `doReconstruction` (*Boolean*): If set to `true`, a full reconstruction will be performed after the phase image is opened, which will update its content. Optional, default value is `true` (recommended).
- `updateXYScale` (*Boolean*): If set to `true`, the scale displayed in the *Phase Image window* will be updated. Optional, default value is `true`.

Return: void (nothing)

Requirements:

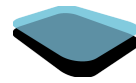
- A configuration must be loaded
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements \Rightarrow *Invalid Operation* error

See also:


- [OpenHoloWin](#)
- [OpenIntensityWin](#)



RecordStroboFixFrequency

Starts a fix frequency recording with the stroboscopic tool, for a given number of periods. The stroboscopic tool must have been previously configured by hand in Koala: the stroboscopic window must be opened on the *Main* tab, and all options except the number of periods must have been set manually.

Koala UI equivalent:

Corresponds to enabling *Record at start* and then clicking on the *Start stroboscopic mode* button  in the stroboscopic window.

Parameters:

- `numberOfPeriods (Int32)`: The number of periods of the signal generated by the stroboscopic tool to record

Return: String: the path where the result has been saved

Requirements:

- The stroboscopic module must be enabled
- A configuration must be loaded
- The camera must be available
- The stroboscopic tool must be available
- The stroboscopic window must be opened
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error
- The stroboscopic window is not configured correctly ⇒ *Execution failed* error


See also:

- [RecordStroboFrequencyScan](#)

RecordStroboFrequencyScan

Starts a frequency scan recording with the stroboscopic tool. The stroboscopic tool must have been previously configured by hand in Koala: the stroboscopic window must be opened on the *Scanning* tab, the *Frequency scan* option must have been selected and the desired options (minimal and maximal frequency, step size, number of periods per frequency, etc...) must have been set manually).

Koala UI equivalent:

Corresponds to enabling *Record at start* and then clicking on the *Start stroboscopic* mode button  in the stroboscopic window.

(No parameters)

Return: String: the path where the result has been saved

Requirements:

- The stroboscopic module must be enabled
- A configuration must be loaded
- The camera must be available
- The stroboscopic tool must be available
- The stroboscopic window must be opened
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error
- The stroboscopic window is not configured correctly ⇒ *Execution failed* error

See also:


- [RecordStroboFixFrequency](#)



ResetCorrSegment

Reset the phase correction segments or zone.

Koala UI equivalent:

In the *Phase Image* window, clicking on the *Reset phase correction segments* or *Reset phase correction zones* context menu (which appears when the *Horizontal and Vertical profile for phase mask adjustment*  button is activated, one or more phase correction segments, respectively zones, have been drawn and the button was the last activated button).

Parameters:

- dimension (Int32): 1 for resetting the phase correction segments, 2 for resetting the phase correction zones. Optional, default value is 1.

Return: void (nothing)

Requirements:

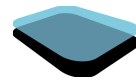
- A configuration must be loaded
- The phase window must be opened and must contain an image
- A reconstruction object must be created
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements \Rightarrow *Invalid Operation* error

See also:

- [AddCorrSegment](#)
- [AlgoResetPhaseMask](#)
- [ComputePhaseCorrection](#)



ResetGrab

Stops the continuous acquisition of the camera if it was ON.

Koala UI equivalent:

(Not applicable)

(No parameters)

Return: void (nothing)

Requirements:

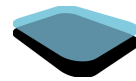
- The intern camera must be available
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements \Rightarrow *Invalid Operation* error

See also:

- [DigitizerAcquiring](#)



SaveImageFloatToFile

Saves a 32 bits image on the disk.

Note that the function does not return an error if the low-level saving operation failed.

Koala UI equivalent:

Clicking on the *Save* → *Save as float* context menu of an image

Parameters:

- `winId` (Int32): Id of the image to save. Possible values are:
 - 1: hologram
 - 2: amplitude image
 - 4: phase image
 - 8: Fourier imageFor amplitude, phase and Fourier images, what exact type of image is saved (Lambda 1 only, Lambda 2 only, short or long synthetic wavelength or wavelength mapping) depends on the selected image in the corresponding window. Values cannot be combined. To save several images, call the function several times.
- `fileName` (String): Full path of the destination file. The extension has no influence on the file format, which is determined by the *useBinFormat* parameter. However it is recommended to use the correct extension to avoid confusion when working with the file later on.
- `useBinFormat` (Boolean): Set to `true` to save as `.bin` file, `false` to save as `.txt`. Optional, default value is `false`.

Return: void (nothing)

Requirements:

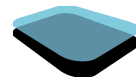
- A configuration must be loaded
- The corresponding window must be opened and must contain an image
- A reconstruction object must be created
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error
- Invalid `winId` ⇒ *Execution Failed* error

See also:

- [SaveImageToFile](#)
- [SelectDisplayWL](#)



SaveImageToFile

Saves an 8 bits image on the disk, as tiff, png or jpeg. If the directory does not exist, it will be created.

Note that the function does not return an error if the low-level saving operation failed.

Koala UI equivalent:

Clicking on the *Save* → *Save as TIF* context menu of an image

Parameters:

- winId (Int32): Id of the image to save. Possible values are:
 - 1: hologram
 - 2 : amplitude image
 - 4 : phase image
 - 8 : Fourier imageFor amplitude, phase and Fourier images, what exact type of image is saved (Lambda 1 only, Lambda 2 only, short or long synthetic wavelength or wavelength mapping) depends on the selected image in the corresponding window.
Values cannot be combined. To save several images, call the function several times.
- fileName (String): Full path of the destination file. The file format will be defined according to the extension. If no extension is given, the file will be recorded in tiff format. It is recommended to save holograms as .tif if you intend to load them again for further processing.
Possible extension values are
 - .png
 - .jpg
 - .tiff or .tif

Return: void (nothing)

Requirements:

- A configuration must be loaded
- The corresponding window must be opened and must contain an image
- A reconstruction object must be created
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation*
- Invalid winId ⇒ *Execution Failed*

See also:





- [SaveImageFloatToFile](#)
- [SelectDisplayWL](#)

SelectDisplayWL

Select the specific type of wavelength (WL) to display in the phase, amplitude or Fourier window.

Koala UI equivalent:

Clicking on one of the following buttons in an image:

- Long synthetic WL  button (in the *Phase Image window*)
- Short synthetic WL  button (in the *Phase Image window*)
- 1st WL  button (in the *Phase / Intensity / Fourier Image windows*)
- 2nd WL  button (in the *Phase / Intensity / Fourier Image windows*)

Parameters:

- winId (Int32): Id of the image to display. Possible values are:
 - 8192: phase lambda 1 image
 - 16384: phase lambda 2 image
 - 32768: phase long synthetic wavelength image
 - 65536: phase short synthetic wavelength image
 - 2048: amplitude (intensity) lambda 1 image
 - 4096: amplitude (intensity) lambda 2 image
 - 512: Fourier lambda 1 image
 - 1024: Fourier lambda 2 image

Values cannot be combined. To display several images, call the function several times.

Return: void (nothing)

Requirements:

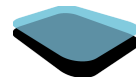
- The corresponding window must be opened and the corresponding image type must be available
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements \Rightarrow *Invalid Operation*
- Invalid winId \Rightarrow *Execution Failed*

See also:


- [OpenIntensityWin](#)
- [OpenPhaseWin](#)



SelectTopoZone

Selects an area for topographic (roughness) measurement.

Koala UI equivalent:


Tracing a zone in the *Phase Image window*, when the *Draw a roughness area* button  is activated.

Parameters:

- top (Int32): Y coordinate of the top left point of the zone, in pixel
- left (Int32): X coordinate of the top left point of the zone, in pixel
- width (Int32): width of the zone, in pixel
- height (Int32): height of the zone, in pixel

Return: void (nothing)

Requirements:

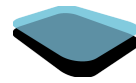
- A configuration must be loaded
- The phase window must be opened and must contain an image
- The topography option  must be selected in the phase window
- A reconstruction object must be created
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation*
- The given zone is not fully inside the phase ROI ⇒ *Execution Failed*

See also:

- [OpenFrmTopography](#)



SetCameraShutterUs

Sets the shutter value of the camera, in [us].

Note that this function does not require a configuration to be loaded, but it doesn't make much sense to set the shutter value before loading a configuration.

Koala UI equivalent:

Setting the *Shutter* value in the *Camera settings window*.

Parameters:

- shutterUs (Int32): The shutter value for the camera, in [us]

Return: void (nothing)

Requirements:

- The intern camera must be available
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

See also:

- [GetCameraShutterUs](#)



SetPhaseProfileState

Opens or closes the phase profile window

Koala UI equivalent:

Opening or closing the *Phase Profile window*

Parameters:

- `state` (Boolean): `true` to open the phase profile window, `false` to close it. Optional, default value is `false`.

Return: void (nothing)

Example: (See example for [GetPhaseProfile](#))

Requirements:

- A configuration must be loaded
- The phase window must be opened and must contain an image
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements \Rightarrow *Invalid Operation*

See also:

- [ExtractPhaseProfile](#)
- [GetPhaseProfile](#)
- [GetPhaseProfileLength](#)



SetRecDistCM

Sets the reconstruction distance, in [cm], for the active user processing configuration.

Note that the value is not saved in the database, the database value will be loaded again the next time the configuration is loaded.

Koala UI equivalent:

Setting the *Focus* value in the *Reconstruction settings window*

Parameters:

- distCM (Single): The reconstruction distance, in [cm]

Return: void (nothing)

Example: (See example for [OnDistanceChange](#))

Requirements:

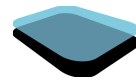
- A configuration must be loaded
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements \Rightarrow *Invalid Operation* error

See also:

- [GetRecDistCM](#)
- [OnDistanceChange](#)



SetSourceState

Sets a source ON or OFF.

Koala UI equivalent:

Clicking on one of the *Sources* buttons in the main menu, or on one of the *ON* or *OFF* buttons in the *Laser Sources window*.

Parameters:

- `srcId` (Int32): The id of the source (logical or physical according to *useLogicalId*)
- `state` (Boolean): Set to `true` to switch the source ON, `false` to switch it OFF
- `useLogicalId` (Boolean): Set to `true` to use logical id, to `false` to use physical id. Optional, default value is `true`.

Return: void (nothing)

Requirements:

- The laser source controller must be available
- If logical ids are used, a configuration must be loaded
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements \Rightarrow *Invalid Operation*

See also:

- -

SetUnwrap2DMethod

Sets the method used for unwrapping.

Note that unwrapping will not take place until it has been enabled with [SetUnwrap2DState](#).

Koala UI equivalent:

Selecting one of the menu option in the *Settings* → *Phase unwrapping* menu

Parameters:

- method (Int32): Id of the method to use for unwrapping. Possible values:
 - 0: Discrete Cosine Transform (DCT)
 - 1: Path-following (also known as Quality Path)
 - 2: Preconditioned Conjugate Gradient (PCG). Do not use, deprecated.

Return: void (nothing)

Requirements:

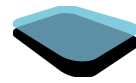
- A configuration must be loaded
- A reconstruction object must be created
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

See also:

- [GetUnwrap2DState](#)
- [SetUnwrap2DState](#)



SetUnwrap2DState

Enables or disables the unwrapping of the phase.

Koala UI equivalent:

Corresponds to clicking on the Unwrap  button of the *Phase Image window*.

Parameters:

- `state (Boolean)`: `true` to enable the unwrapping, `false` to disable it. Optional, default value is `false`.

Return: void (nothing)

Requirements:

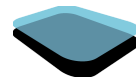
- A configuration must be loaded
- The phase window must be opened and must contain an image
- A reconstruction object must be created
- A sequence must not be playing
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements \Rightarrow *Invalid Operation* error

See also:

- [GetUnwrap2DState](#)
- [SetUnwrap2DMethod](#)



SingleReconstruction

Acquires an image (or several if temporal averaging is on) and reconstruct it.

Koala UI equivalent:

(not applicable)

(No parameters)

Return: void (nothing)

Requirements:

- A configuration must be loaded
- Live mode must be available
- The intern camera must be available
- Video mode must not be running
- A strobo measurement must not be running
- The sequence reconstruction to display window must not be opened
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation*

See also:

- [Acquisition2L](#)

Former functions which were removed

CloseConfig

Closes the current measurement configuration

Koala UI equivalent:

Corresponds to *File* → *Close Configuration*

Parameters:

- `confirm` (*Boolean*): If set to `true`, the confirmation window will be displayed to the user

Return: void (nothing)

Requirements:

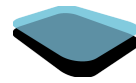
- A configuration must be loaded
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

See also:

- [OpenConfig](#)



GetMaxHoloContrast

Gets the maximum value of the contrast in the hologram image.

Koala UI equivalent:

(not applicable)

(No parameters)

Return: Double

Requirements:

- A configuration must be loaded
- An hologram must have been grabed since program start
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements \Rightarrow *Invalid Operation* error

See also:

- [GetHoloContrast](#)



GetRaNm

Gets the Ra (arithmetic mean roughness) roughness measurement in the selected topography zone.

Koala UI equivalent:

(Not applicable, but roughness measurements are displayed in the *Surface topography Phase window*.)

(No parameters)

Return: Single: The Ra, in [nm]

Requirements:

- A configuration must be loaded
- The phase window must be opened and must contain an image
- A roughness measurement zone must be selected
- The topography window must be opened
- A reconstruction object must be created
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements \Rightarrow *Invalid Operation* error

See also:

- [OpenFrmTopography](#)
- [SelectTopoZone](#)



SaveHolo2L

Saves a combined hologram recorded in alternate mode at different wavelengths

Koala UI equivalent:

(not applicable because this mode is not supported anymore)

Parameters:

- path (String): Full path and name of the destination file.

Return: void (nothing)

Requirements:

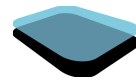
- A configuration must be loaded
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

See also:

- [SaveImageToFile](#)
- [SaveImageFloatToFile](#)



SetSurfLambdaCNm

Sets lambda C (the limit frequency between wavelengths considered as the rugosity and the wavelengths considered as the waviness of a topography), for topography measurements.

Koala UI equivalent:

(Not applicable, but roughness measurement parameters can be loaded via the *Surface topography Phase window*.)

Parameters:

- lambdaC (Single): Wavelength of lambda C, in [nm]

Return: void (nothing)

Requirements:

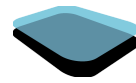
- A configuration must be loaded
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements \Rightarrow *Invalid Operation* error

See also:

- [SetSurfLambdaSNm](#)



SetSurfLambdaSNm

Sets lambda S (the limit frequency between wavelengths considered as noise and the wavelengths considered as the rugosity of a topography), for topography measurements.

Koala UI equivalent:

(Not applicable, but roughness measurement parameters can be loaded via the *Surface topography Phase window*.)

Parameters:

- lambdaS (Single): Wavelength of lambda S, in [nm]

Return: void (nothing)

Requirements:

- A configuration must be loaded
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements \Rightarrow *Invalid Operation* error

See also:

- [SetSurfLambdaCNm](#)

Appendix

Python remote client application example

Note: this example is copied on your computer when you install Koala with the *Remote* option. You can find it in: *C:\Program Files\LynceeTec\Koala\Remote\Remote Examples\Python\PythonRemoteExample.py*

```
# -*- coding: utf-8 -*-

#*****
# NOTES:
# - This example is designed to work with a 2 wavelengths configuration.
#   If you use a single wavelength configuration, you will have to comment
#   some of the lines for the script to run.
#
# - The example grabs an hologram, saves it, and then loads it to illustrate
#   those functionalities. If you comment the grab and save part, you can
#   run the example offline.
#
# - You might need to adapt some of the parameters (such as the save/load
#   location or the default configuration number) for the example to work
#   on your system with your database
#*****

import sys
import clr #Install pythonnet package to get clr
from pathlib import Path
import numpy as np
import matplotlib.pyplot as plt
import ctypes

#Add Koala remote librairies to Path
sys.path.append(r'C:\Program Files\LynceeTec\Koala\Remote Libraries\x64') #load
x86 for 32 bits applications
#Import KoalaRemoteClient
clr.AddReference("LynceeTec.KoalaRemote.Client")
from LynceeTec.KoalaRemote.Client import KoalaRemoteClient

#get input form console, if input is empty us default value
def get_input(question,default) :
    answer = input(question+'['+default+'] :')
    return answer or default

#Define KoalaRemoteClient host
```

```
host=KoalaRemoteClient()

#Ask IP adress
IP = get_input('Enter host IP adress','localhost')
#Log on Koala
username = ''
[ret,username] = host.Connect(IP,username,True);
password = get_input('Enter password for '+username+' account', username)
host.Login(password)

#Open a 2 wavelenghts configuration.
config = get_input('Enter configuration number', default='137')
host.OpenConfig(config);

#Open main display windows
host.OpenPhaseWin();
host.OpenIntensityWin();
host.OpenHoloWin();

#This block records an hologram so that you can later work offline with the
rest of the script.
#Set logical source 0 (the 1st source of the current configuration) to ON
host.SetSourceState(0, True, True)
#Set logical source 1 (the 2nd source of the current configuration) to ON
host.SetSourceState(1, True, True)
#Acquire on hologram
host.Acquisition2L()
host.ResetGrab()
#Save holo to file
path = Path(r'c:\tmp')
host.SaveImageToFile(1, str(path/'holo.tiff'))

#Load previously recorded hologram
host.LoadHolo(str(path/'holo.tiff'), 2);

#Display lambda 1 image in phase window
host.SelectDisplayWL(8192);

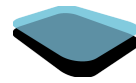
#Save hologram image
host.SaveImageToFile(1, str(path/'holo.tiff'));
#Save intensity image
host.SaveImageToFile(2, str(path/'intensity.tiff'));
#Save phase image (as displayed, which means phase lamabda 1)
host.SaveImageToFile(4, str(path/'phase.tif'));
##Save intensity float in bin
host.SaveImageFloatToFile(2, str(path/'intensity.bin'), True);
##Save phase float in bin
host.SaveImageFloatToFile(4, str(path/'phase.bin'), True);
```

```
#!/This block only works for 2 wavelengths configurations
#!/Display lambda 2 image in intensity window
host.SelectDisplayWL(4096);
#!/Display lambda 2 image in phase window
host.SelectDisplayWL(16384);
#!/Save intensity image (as displayed, which means intensity lambda 2)
host.SaveImageToFile(2, str(path/'intensity2.tiff'));
#!/Save phase image (as displayed, which means phase lambda 2)
host.SaveImageToFile(4, str(path/'phase2.tif'));
host.SaveImageFloatToFile(2, str(path/'intensity2.bin'), True);
host.SaveImageFloatToFile(4, str(path/'phase2.bin'), True);

#!/Gets the current reconstruction distance
recDist = host.GetRecDistCM();
#!/Set a new reconstruction distance
host.SetRecDistCM(recDist * 1.1);
#Do a reconstruction with this new distance
host.OnDistanceChange();

#!/Get phase image for computation
roiWidth = host.GetPhaseWidth();
roiHeight = host.GetPhaseHeight();
roiStride = (roiWidth / 4) * 4;
size = int(roiStride * roiHeight)
if (roiWidth % 4 != 0) :
    roiStride += 4;
clr.AddReference("System")
import System
from System import Array
bufPhase=Array.CreateInstance(System.Single,size)
host.GetPhase32fImage(bufPhase);
#Convert DotNET Array to numpy Array
phase = np.zeros(size, dtype=float)
for x in range(0,(int)(size-1)):
    phase[x]=float(bufPhase[x])
phase = np.reshape(phase, (roiHeight,roiWidth))
#plot phase numpy
plt.imshow(phase,cmap="Greys")

#!/Extract a profile
host.SetPhaseProfileState(True)
host.ExtractPhaseProfile(100, 100, 200, 200)
profileLength = host.GetPhaseProfileLength()
bufProfile = Array.CreateInstance(System.Double,profileLength)
host.GetPhaseProfile(bufProfile)
#Convert DotNET Array to numpy Array
profile = np.zeros(profileLength, dtype=np.double)
```

```
for x in range(0,(int)(profileLength-1)):
    profile[x]=float(bufProfile[x])
#Get xy values to plot calibrated profile
distance = np.arange(profileLength) * host.GetPxSizeUm()
plt.figure()
plt.plot(distance,profile)
plt.xlabel('dist [um]')
plt.ylabel('OPL [nm]')
plt.show()

#//Reset phase correction segments
host.ResetCorrSegment();
#Add new phase profile correction
host.AddCorrSegment(100, 100, 500, 1);
host.AddCorrSegment(200, 200, 600, 0);
#//Compute 1D phase correction using tilt method
host.ComputePhaseCorrection(0, 1);

#Logout
host.Logout()
```