Santec Hardware Library Programming Guide

Version 2.3.1.0

Table of Contents

- Breaking Changes
- Articles
  - Quick Start Guide
  - Using an RD-P
  - Other RD-P Usage
- Api Documentation
  - Santec
    - Version
  - Santec.Hardware.AssemblyResolvers
    - VisaResolver
  - Santec.Hardware.Converters
    - FiberTypeToStringConverter
  - Santec.Hardware.Devices.Attenuation
    - EBeamBlockState
    - EOVAModel
    - IOVA
    - OVA
  - Santec.Hardware.Devices.Backreflection
    - BRM
    - EBRMModel
    - IBackreflectionMeter
    - IBRM
    - SimulatedBRM
  - Santec.Hardware.Devices.General
    - CompositeSantecDevice
    - DeviceDetector
    - DeviceDetectorWithSerialNumber
    - DeviceFactory
    - ECoreSize
    - EDarkState
    - EFiberType
    - EInteractionMode
    - ERangeStatus

ETimeout
ETriggerEdge
FiberInformation
ICompositeDevice
IDetectorDevice
IDetectorDeviceWithDarking
IDevice
IDevice<TModelType>
IDeviceWithInputs
IDeviceWithWavelengthRange
IFiberDevice
ISantecDevice
ISimulatedDevice
IWavelengthDevice
PhysicalDevice
QueryResponsePair
RDSPDetector
ReadingValue
SantecDevice
SimulatedDeviceFactory
SimulatedSantecDevice
Santec.Hardware.Devices.InsertionLoss
IInsertionLossMeter
Santec.Hardware.Devices.Modular
EXCSModel
IModuleController
ISLS
ISubModuleController
IXCS
SimulatedSLS
SimulatedXCS
XCS
Santec.Hardware.Devices.Modular.Modules
IModule
IModuleInformation
Santec.Hardware.Devices.Modular.Modules.Attenuation
AttenuatorModule

EAttenuatorModuleType
IAttenuatorModule
SimulatedAttenuatorModule
SimulatedAttenuatorModuleInformation
Santec.Hardware.Devices.Modular.Modules.Polarization
IPolarizationControllerModule
PolarizationControllerModule
SimulatedPolarizationControllerModuleInformation
Santec.Hardware.Devices.Modular.Modules.Power
EPowerMeterModuleType
IOPMModule
IPowerMeterModule
OPMModule
Santec.Hardware.Devices.Modular.Modules.Source
DiscreteSourceSubModule
DiscreteSourceSubModuleInformation
ESourceModuleType
IDiscreteSourceSubModule
ISourceModule
ISourceSubModule
ISourceSubModuleInformation
SimulatedDiscreteSourceSubModule
SimulatedSourceModule
SimulatedSourceModuleInformation
SourceModule
SourceModuleWithCaseTemperature
SourceSubModuleWithTemperatureControl
Santec.Hardware.Devices.Modular.Modules.Switching
ESwitchModuleType
ISwitchModule
SimulatedSwitchModule
SimulatedSwitchModuleInformation
SwitchModule
Santec.Hardware.Devices.Polarity
CableInformation
CameraRecord
ECameraMode

ELossCompensationConfiguration
EMappingMode
EMappingSensitivity
EPTMModel
ERDPModel
ERDPRequirementFailureReason
ERDSPModel
ERDSPSystemHardwareRequirementsFailureReason
IPTM
IPTM<TModelType>
IRDP
IRDP<TModelType>
IRDPFactory
IRDSP
LegacyCustomCameraSettings
PTM
RDP
RDPFactory
RDSP
RDSPCameraValidator
SimulatedPTM
SimulatedRDP
SimulatedRDPFactory
SimulatedRDSP
Santec.Hardware.Devices.Polarization
EAutoPowerAdjustmentState
EPCUModel
EPEMModel
EPERMode
EPLMModel
EPolarizationMode
EPolarizationState
IPCU
IPEM
IPLM
IPolarizationController
PCU

PDLAndBRReading
PDLReading
PEM
PERReading
PLM
SimulatedPCU
SimulatedPEM
SimulatedPLM
Santec.Hardware.Devices.Power
EAutoGainRangeModeState
ELoggingStatus
EOPMModel
EReadingMode
EResolution
ETriggerBehaviourMode
GainRange
IContinuousPowerMeter
IDiscretePowerMeter
IOPM
IPowerMeter
IPowerMeterWithReferencePersistence
OPM
Santec.Hardware.Devices.Requirements
HardwareRequirementFailure<T>
HardwareRequirementsEvaluationResult<T>
SystemHardwareRequirementsEvaluationResult<T>
Santec.Hardware.Devices.ReturnLoss
EGainMode
ELengthMode
EPositionBMode
ERLMModel
ERLMode
IRLM
RLM
RLReading
SimulatedRLM
Santec.Hardware.Devices.ReturnLoss.Extensions

RLMExtensions
Santec.Hardware.Devices.ReturnLoss.Trace
ERLMType
TraceDiagnosticInfo
TraceDiagnostics
Santec.Hardware.Devices.Source
ESourceState
FactoryPowerReading
IDiscreteSource
ILaserSource
ILaserSourceWithMonitor
ISource
Santec.Hardware.Devices.Switching
DeviceSwitch
DeviceSwitchAndChannelPair
EOSXModel
ExternalDeviceSwitch
IOSX
ISwitchController
OSX
SimulatedExternalDeviceSwitchInformation
SimulatedOSX
Santec.Hardware.Exceptions
AmbientLightDetectedException
CameraNotFoundException
CompositeDeviceHardwareException
DeviceTimeoutException
PolarityMappingFailedException
UnableToPersistILReferenceException
UnexpectedDeviceResponseException
VisaNotFoundException
Santec.Hardware.Factories
ConverterFactory
Santec.Hardware.Services
EConnectionTypesToDetect
HardwareDetectionService
IHardwareDetectionService

SimulatedHardwareDetectionService

Breaking Changes

1.2.0.0

- IRDP: *void SetLossCompensation(int) → Task SetLossCompensationAsync(int)*

1.3.0.0

- *class DetectorDeviceWithSerialNumber → class DeviceDetectorWithSerialNumber*

1.4.0.0

- IRDP: *Task CollectOutputMappingDataAsync(int, CancellationToken) →*
  Task<bool> CollectOutputMappingDataAsync(int, CancellationToken)
- IRDP: *Task CollectOutputMappingDataAsync(int, int, CancellationToken) →*
  Task<bool> CollectOutputMappingDataAsync(int, int, CancellationToken)

1.5.0.0

- *enum Santec.Hardware.Devices.ETriggerEdge → enum Santec.Hardware.Devices.General.ETriggerEdge*

1.10.0.0

- ILaserSource: *Task<double> GetFactoryPowerAsync(int, int, CancellationToken) →*
  Task<FactoryPowerReading> GetFactoryPowerAsync(int, int, CancellationToken)

1.11.0.0

- IRLM: *Task<double> GetDUTILAsync(CancellationToken) →*
  Task<double> GetDUTILAsync(int, CancellationToken)
- IRLM: *Task SetDUTILAsync(double, CancellationToken) →*
  Task SetDUTILAsync(int, double, CancellationToken)

1.13.0.0

- IOVA: *Task ChangeAllModuleAttenuationAsync(double, CancellationToken) →*
  Task ChangeAllModuleAttenuationsAsync(double, CancellationToken)
- IOVA: *Task GetModuleBeamBlockStateAsync(int, CancellationToken) →*
  Task<EBeamBlockState> GetModuleBeamBlockStateAsync(int, CancellationToken)

1.14.0.0

- IPEM: *Santec.Hardware.Devices.Polarization.PEM.IPEM →*
  Santec.Hardware.Devices.Polarization.IPEM
- PEM: *Santec.Hardware.Devices.Polarization.PEM.PEM →*
  Santec.Hardware.Devices.Polarization.PEM
- SimulatedPEM: *Santec.Hardware.Devices.Polarization.PEM.SimulatedPEM →*
  Santec.Hardware.Devices.Polarization.SimulatedPEM

1.15.0.0

- IBackreflectionMeter: *Task<double> ReadBRAsync(int, CancellationToken) →*
  Task<ReadingValue> ReadBRAsync(int, CancellationToken)

- PDLAndBRReading: *double BR →*
  ReadingValue BR

1.16.0.0

- SimulatedRDPFactory: *void ConfigureCanDetectCamera(bool) →*
  void ConfigureCameras(IReadOnlyList<CameraRecord>)
- PolarityMappingFailedException: *Mat PolarityMappingImage →*
  IReadOnlyList<Mat> PolarityMappingDetectorImages

1.21.0.0

- IRDPFactory: Updated various *ArgumentException* to be *InvalidOperationException* to more accurately reflect the nature of the hardware related exceptions when creating an RD-P/RD-SP.
- IRDPFactory: Updated *CameraNotFoundException* to be *ArgumentException* for methods that accept a list of cameras as a parameter.
- HardwareRequirementsEvaluationResult: Obsoleted *RequirementFailureDetails* in favour of *RequirementFailures*.

1.23.0.0

- CompositeSantecDevice: *virtual void Uninitialize() →*
  void Uninitialize()

1.25.0.0

- *enum Santec.Hardware.Devices.Requirements.ERDPRequirementFailureReason → enum Santec.Hardware.Devices.Polarity.ERDPRequirementFailureReason*
- *class Santec.Hardware.Devices.Requirements.HardwareRequirementsEvaluationResult → class Santec.Hardware.Devices.Requirements.HardwareRequirementsEvaluationResult<T>*
- HardwareRequirementsEvaluationResult: ~~*RequirementFailureDetails*~~
- HardwareRequirementsEvaluationResult: ~~*HardwareRequirementsEvaluationResult(IDevice, bool, IReadOnlyList<string>)*~~
- IRDPFactory: *HardwareRequirementsEvaluationResult CheckIfRLMMeetsHardwareRequirementsWithDetails(IRLM) → HardwareRequirementsEvaluationResult<ERDPRequirementFailureReason> CheckIfRLMMeetsHardwareRequirementsWithDetails(IRLM)*
- IRDPFactory: *HardwareRequirementsEvaluationResult CheckIfRLMMeetsHardwareRequirementsWithDetails(IRLM, IReadOnlyList<CameraRecord>) → HardwareRequirementsEvaluationResult<ERDPRequirementFailureReason> CheckIfRLMMeetsHardwareRequirementsWithDetails(IRLM, IReadOnlyList<CameraRecord>)*

2.0.0.0

- IPEM: Overhauled the *IPEM* to reflect changes to its behaviour and its supported SCPI commands
- IPEM: USB PID- ~~0xD00A~~ → 0xD00E
- IPEM: ~~*IBackreflectionMeter*~~ *→ IDetectorDeviceWithDarking*
- IPEM: ~~*Task TurnOnPolarizerAsync(CancellationToken)*~~
- IPEM: ~~*Task TurnOffPolarizerAsync(CancellationToken)*~~
- IPEM: ~~*Task<double> GetPolarizerAngleAsync(CancellationToken)*~~
- IPEM: ~~*Task SetPolarizerAngleAsync(double, CancellationToken)*~~
- IPEM: ~~*enum EPolarizerSpeed*~~ *→ enum EPERMode*
- IPEM: ~~*Task<EPolarizerSpeed> GetPolarizerSpeedAsync(CancellationToken)*~~ *→ Task<EPERMode> GetPERModeAsync(CancellationToken)*
- IPEM: ~~*Task SetPolarizerSpeedAsync(EPolarizerSpeed, CancellationToken)*~~ *→ Task SetPERModeAsync(EPERMode, CancellationToken);*
- ~~*class Santec.Hardware.Devices.Polarization.PowerExtremaReading*~~

- IPEM: ~~*Task<PowerExtremaReading> ReadPowerExtremaAsync(int, int, CancellationToken)*~~ → *Task<double> ReadMinimumPowerAsync(int, int, CancellationToken) + Task<double> ReadMaximumPowerAsync(int, int, CancellationToken)*
- PERReading: ~~*PERReading(double, double, double)*~~ → *PERReading(double, double, double, double)*
- ILaserSource: *ILaserSource* → *ILaserSource + ILaserSourceWithMonitor*
- ILaserSource: ~~*Task<double> ReadMonitorPowerAsync(int, CancellationToken)*~~
- ILaserSource: ~~*Task<FactoryPowerReading> GetFactoryPowerAsync(int, int, CancellationToken)*~~
- IBRM: ~~*ILaserSource*~~ → *ILaserSourceWithMonitor*
- IPLM: ~~*ILaserSource*~~ → *ILaserSourceWithMonitor*
- IRLM: ~~*ILaserSource*~~ → *ILaserSourceWithMonitor*
- IPEM: ~~*IPowerMeter*~~ → *IInsertionLossMeter*
- IPEM: *Task<EDarkState> GetDetectorDarkStateAsync(int, CancellationToken)* now throws a *NotSupportedException*
- IPEM: *Task<double> ReadILAsync(int, int, CancellationToken)* now throws a *NotSupportedException*
- IPEM: *Task SetILReferenceAsync(int, int, double, CancellationToken)* now throws a *NotSupportedException*

Quick Start Guide

This library is designed to get you up and running as quickly as possible. The library will take care of all the nitty gritty details involved with setting up communications to the device (Ethernet or USB), initializing the device objects and sending and receiving data in the correct format with error handling built in.

Compatibility

The library is written in C# (.NET Framework 4.8). It can be easily referenced and used by any CLI (Common Language Infrastructure) language.

Finally, a VISA (Virtual Instrument Software Architecture) driver is required on the system if communicating over USB. We recommend the driver from Rhode & Schwarz but many others are available. See the list at IVI.

Ethernet communications will work out of the box with no specific driver install needed.

Setting Up

First, we will need to add *Santec.Hardware.dll*, *Santec.Core.dll*, *System.Text.Json*, *Zeroconf*, and *NGettext* references to your project.

(System.Text.Json can be found on NuGet in Visual Studio or downloaded here:
https://www.nuget.org/packages/System.Text.Json)
(Zeroconf can be found on NuGet in Visual Studio or downloaded here: https://www.nuget.org/packages/Zeroconf/)
(NGettext can be found on NuGet in Visual Studio or downloaded here: https://github.com/VitaliiTsilnyk/NGettext)

Accessing Hardware

There are two ways to work with this library:

- Use physical hardware (RLM, OSX, PTM, etc.)
- Use simulated devices

Simulated devices are intended for use in development if no physical hardware is available. First, we will assume you have access to a physical device via USB connection to the development PC.

In a new class create the following objects:

```
public IHardwareDetectionService hardwareDetectionService = new HardwareDetectionService();
public List<IRLM> detectedLocalDevices;
public IRLM santecDevice;
```

The hardware detection service (*IHardwareDetectionService*) is responsible for finding all available Santec Canada hardware devices. We also need a list of *IRLM*s to store the detected hardware. Finally, we create a single *IRLM* to store the connection to the particular RLM we will be using.

In this example, the device is specifically an *IRLM*. In general, you will get back a list of whatever device type is passed into the *DetectHardwareAsync* method call. See the *IRLM* type parameter passed in below.

```
detectedLocalDevices = await hardwareDetectionService.DetectHardwareAsync<IRLM>
(EConnectionTypesToDetect.All);
```

If we want to get a list of all types of devices, we would set up the code like this instead:

```
public IHardwareDetectionService hardwareDetectionService = new HardwareDetectionService();
public List<IDevice> detectedLocalDevices;
public IDevice santecDevice;
```

Notice the generic *IDevice* used instead of *IRLM*.

In this case, no type parameter should be passed to the *DetectHardwareAsync* method call.

```
detectedLocalDevices = await hardwareDetectionService.DetectHardwareAsync<>(EConnectionTypesToDetect.All);
```

You then cast the device to its type in order to gain access to all the functions. For example, instead of

```
il = await santecDevice.ReadILAsync(1,1310);
```

use

```
il = await ((IRLM)santecDevice).ReadILAsync(1,1310);
```

Using the Hardware Library

Time to get started. First, define a new *Detect* method to actually do the retrieval of the device list. As written this code will find all RLM devices on the local network (via the MDNS in Zeroconf) and connected through USB to the computer (via the VISA driver on the computer).

```
private async void Detect()
{
    //detect the RLMs only
    detectedLocalDevices = await hardwareDetectionService.DetectHardwareAsync<IRLM>
(EConnectionTypesToDetect.All);

    //grab the first device, use your own logic here to get the correct device.
    santecDevice  = detectedLocalDevices[0];

    label1.Text = santecDevice.Name;

    //now initialize the device
    await santecDevice.InitializeAsync();
}
```

Once that is done we can start using the device.

Start by referencing the IL. For this, we use the *ReferenceILAsync* method. To get a result back, we must *await* the command. Notice that all SCPI commands will have a corresponding method that can be called via the dot operator. Please refer to the API documentation to see all available commands.

```
private async void Reference()
{
    double refIL;

    //use last detector and first wavelength in the RLM
    await santecDevice.ReferenceILAsync(santecDevice.NumberOfDetectors, santecDevice.Wavelengths[0]);

    refIL = await santecDevice.GetILReferenceAsync(santecDevice.NumberOfDetectors,
santecDevice.Wavelengths[0]);

    //print reference value to the label
    label1.Text = refIL.ToString();
}
```

For example, run an IL measurement in a loop. Doing this asynchronously will enable the program to be responsive at the same time.

```
private async void Loop()
{
    double il;

    //loop 50 times as an example
    for (int i = 0; i < 50; i++)
    {
        //read IL for the last detector and first wavelength
        //in the real world, you should perform a reference first
        il = await santecDevice.ReadILAsync(santecDevice.NumberOfDetectors, santecDevice.Wavelengths[0]);

        label1.Text = il.ToString();
    }

    label1.Text = "done loop";
}
```

Using an RD-P

The RD-P is a composite device consisting of an RD-P connected to the computer through USB and either

- A dual-output RLM with two OSXs connected and either a 1310nm or 1300nm laser

  or

- An RLM with an OSX connected and either a 1310nm or 1300nm laser

  or

- A dual-output RLM with either a 1310nm or 1300nm laser

As a composite device, the RD-P will not be detected using the hardware detection service (*IHardwareDetectionService*) and, instead, must be created with an RD-P factory (*IRDPFactory*).

The RD-P does not support all operations of a PTM. It primarily supports the polarity mapping operation. Any unsupported operations will throw a not supported exception (*NotSupportedException*).

Setting Up

Using an RD-P requires some additional libraries. You will need to add *OpenCvSharp4.Windows*, *Nito.AsyncEx.Context*, and *TIS.Imaging.ICImagingControl35* references to your project.

(OpenCvSharp4.Windows can be found on NuGet in Visual Studio or downloaded here: https://www.nuget.org/packages/OpenCvSharp4.Windows)
(Nito.AsyncEx.Context can be found on NuGet in Visual Studio or downloaded here: https://www.nuget.org/packages/nito.asyncex.context/)
(TIS.Imaging.ICImagingControl35 can be downloaded here: https://www.theimagingsource.com/support/downloads-for-windows/software-development-kits-sdks/icimagingcontrolcsharp/)

Note: The hardware library was compiled against OpenCvSharp4 4.6.0.20220608. A breaking change was introduced to OpenCvSharp4 at a later point that prevents the hardware library from working with some newer versions of OpenCvSharp4.

General Usage

The RD-P has two mapping modes to choose from: MPO and Duplex. These mapping modes account for the difference in the orientation of fibers between MPO and Duplex connectors and should be selected to match the DUT being tested. Note: The default mapping mode is MPO.

The RD-P also has two mapping sensitivities: Connector and BareFiber. These mapping sensitivities account for the differences in how tight the rows of fibers are between connectors and bare fiber. MPO connectors can have rows of fibers packed quite tightly so the Connector sensitivity uses a tight threshold for differentiating the rows of the DUT. On the other hand, rows of bare fiber are packed loosely and there is large variance between the fibers in a row so the BareFiber sensitivity uses a loose threshold for differentiating the rows of the DUT. Note: The default mapping mode is Connector.

The RD-P also has two usage modes. In basic usage mode, the RD-P handles operating the RLM and OSX. It handles turning on the RLM laser, switching the OSX or RLM channel, capturing the polarity data, and analyzing the results to produce a polarity mapping. In basic mode, the first *n* channels are used for an *n* channel mapping. Ex: A 12 channel MPO will use OSX channels 1-12.

The segmented mapping advanced usage mode exists to handle cases where the DUT is not connected to the first *n* channels of the switch or where other hardware must be controlled during the polarity mapping operation. In this scenario, the RD-P exposes methods to start a polarity operation, capture the polarity data, analyze the results, and end the polarity mapping operation. The user must control the RLM, OSX, and any other hardware required. This includes turning on the RLM laser and switching the OSX

or RLM to the appropriate channel.

The RD-P has loss compensation to account for the total loss in the RD-P and DUT setup. The RD-P can compensate for input powers up to -22dBm. A binned approach is used to adjust the camera gain and exposure settings to compensate for the loss. From 0dBm to -10dBm, the RD-P uses its standard settings. From -10dBm to -14dBm, the camera gain is increased. From -14dBm to -18dBm, the camera exposure is increased and the camera gain is further increased. Acquiring images will take longer with these settings. From -18dBm to -22dBm, the exposure is further increased. Acquiring images will take even longer.

Lastly, it is important to note that the RD-P cannot always generate a polarity mapping for the DUT. In these cases it will throw an exception (*PolarityMappingFailedException*). This exception contains a composite image of the RD-P's captured images. It also contains a list of the fibers that the RD-P detected light for. Note that this list is not a mapping and is simply which fibers were not dark. The polarity mapping methods (*MapPolarityAsync* for basic usage and *ProcessMappingData* for advanced usage) can optionally take an expected polarity mapping as a parameter to expand the range of scenarios where the RD-P can generate a mapping. However, even with an expected mapping, it is still possible that the RD-P will not be able to generate a mapping and will throw the exception. In thise case, refer to the image and/or the list of detected fibers to determine the issue with the DUT.

Full Example

Below is a full sample showing how to create an RD-P and how to use the RD-P in both the basic usage mode and the segmented mapping advanced usage mode. Following the sample is a breakdown of the code with more information on each step involved.

```
using Santec.Hardware.Devices.General;
using Santec.Hardware.Devices.Polarity;
using Santec.Hardware.Devices.ReturnLoss;
using Santec.Hardware.Exceptions;
using Santec.Hardware.Services;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading;
using System.Threading.Tasks;

namespace RDPSample
{
    class Program
    {
        static async Task Main(string[] args)
        {
            IRDPFactory rdpFactory = new RDPFactory();
            IHardwareDetectionService hardwareDetectionService = new HardwareDetectionService();
            List<IRLM> detectedRLMs;
            IRLM rlm = null;
            IRDP rdp;

            int wavelength;
            IReadOnlyList<int?> polarityMapping;

            // Creating an RD-P
            Console.WriteLine("Creating the RD-P...");

            // Step 1. Check that an RD-P camera is connected to the computer.
            if (!rdpFactory.CheckForCamera())
            {
                throw new Exception("Could not find RD-P.");
            }

            // Step 2. Find all USB RLMs
            detectedRLMs = await hardwareDetectionService.DetectHardwareAsync<IRLM>
(EConnectionTypesToDetect.USB);

            // Step 3. Find the first RLM that meets the RD-P hardware requirements
            foreach(IRLM detectedRLM in detectedRLMs)
            {
                //RLMs must be initialized
                await detectedRLM.InitializeAsync();

                if (rdpFactory.CheckIfRLMMeetsHardwareRequirements(detectedRLM))
                {
                    //found an RLM that meets the hardware requirements
                    rlm = detectedRLM;
                    break;
                }
                else
                {
                    //uninitialize the RLM since it does not meet the requirements
                    detectedRLM.Uninitialize();
                }
            }
```

```
            if (rlm == null)
            {
                throw new Exception("Could not find RLM that meets the RD-P hardware requirements.");
            }

            // Step 4. Create the RD-P
            rdp = rdpFactory.CreateRDP(rlm);

            // Step 5. Initialize the RD-P
            await rdp.InitializeAsync();

            // Basic Usage- 12 channel MPO
            Console.WriteLine("Running basic polarity mapping...");

            // Step 1. Set the RD-P mapping mode, mapping sensitivity, and loss compensation
            rdp.SetMappingMode(EMappingMode.MPO);

            rdp.SetMappingSensitivity(EMappingSensitivity.Connector);

            await rdp.SetLossCompensationAsync(0);

            //Step 2. Perform the polarity mapping
            try
            {
                polarityMapping = await rdp.MapPolarityAsync(12, new List<int?> { 1, 2, 3, 4, 5, 6, 7, 8, 9,
10, 11, 12 });

                //show the polarity mapping
                Console.WriteLine("Basic polarity mapping: " + string.Join(",", polarityMapping));
            }
            catch(PolarityMappingFailedException ex)
            {
                Console.WriteLine("Basic polarity mapping failed. Saving image...");

                //save an image of the failed polarity mapping
                ex.PolarityMappingImage.SaveImage("Basic Polarity Mapping Failed.png");

                Console.WriteLine("Image saved.");
            }

            // Advanced Usage: Segmented Polarity Mapping- 12 channel MPO
            Console.WriteLine("Running segmented polarity mapping...");

            // Step 1. Set the RD-P mapping mode, mapping sensitivity, and loss compensation
            rdp.SetMappingMode(EMappingMode.MPO);

            rdp.SetMappingSensitivity(EMappingSensitivity.Connector);

            await rdp.SetLossCompensationAsync(0);

            // Step 2. Start the polarity mapping operation
            await rdp.StartMappingPolarityAsync(12);

            // Step 3. Setup the RLM for the test
            await rlm.SetInteractionModeAsync(EInteractionMode.Remote);

            wavelength = rlm.Wavelengths.First(rlmWavelength => rlmWavelength == 1310 || rlmWavelength ==
1300);

            await rlm.TurnOnLaserAsync(wavelength);

            // Step 4. Switch to the correct optical path for each fiber and collect the mapping data
            await rlm.ChangeOutputChannelAsync(1);
```

```
            for (int i = 0; i < 12; i++)
            {
                //in this case, we are just running through the channels in reverse
                await rlm.ChangeSwitchChannelAsync(1, 12 - i);

                // When we collect the mapping data, the method returns whether a fiber was detected
                fiberDetected = await rdp.CollectOutputMappingDataAsync(i + 1);

                if (fiberDetected)
                {
                    Console.WriteLine($"Fiber ${i + 1} detected.");
                }
                else
                {
                    Console.WriteLine($"Fiber ${i + 1} not detected.");
                }
            }

            // Step 5. Cleanup from the test
            await rlm.TurnOffLaserAsync();

            // Step 6. Process the collected mapping data into a mapping
            try
            {
                polarityMapping = rdp.ProcessMappingData(new List<int?> { 12, 11, 10, 9, 8, 7, 6, 5, 4, 3,
2, 1 });

                //show the polarity mapping
                Console.WriteLine("Advanced polarity mapping: " + string.Join(",", polarityMapping));
            }
            catch (PolarityMappingFailedException ex)
            {
                Console.WriteLine("Advanced polarity mapping failed. Saving image...");

                //save an image of the failed polarity mapping
                ex.PolarityMappingImage.SaveImage("Advanced Polarity Mapping Failed.png");

                Console.WriteLine("Image saved.");
            }

            // Step 7. End the polarity mapping operation
            await rdp.EndMappingPolarityAsync();

            // Cleanup
            // Step 1. Uninitialize the RD-P.
            rdp.Uninitialize();

            Console.WriteLine("Press any key to exit...");

            Console.ReadKey();
        }
    }
}
```

Creating an RD-P

The first step in using an RD-P is to create the RD-P using an RD-P factory (*IRDPFactory*). The factory contains methods for creating an RD-P, evaluating whether an RLM meets the RD-P hardware requirements, and checking whether an RD-P camera is connected to the computer. To meet the hardware requirements an RLM must be initialized, must be either an RLM with an OSX connected or a dual-output RLM, and must have a 1310nm or 1300nm laser.

The following steps shows the process of creating an RD-P:

```
// Creating an RD-P
Console.WriteLine("Creating the RD-P...");

// Step 1. Check that an RD-P camera is connected to the computer.
if (!rdpFactory.CheckForCamera())
{
    throw new Exception("Could not find RD-P.");
}

// Step 2. Find all USB RLMs
detectedRLMs = await hardwareDetectionService.DetectHardwareAsync<IRLM>(EConnectionTypesToDetect.USB);

// Step 3. Find the first RLM that meets the RD-P hardware requirements
foreach(IRLM detectedRLM in detectedRLMs)
{
    //RLMs must be initialized
    await detectedRLM.InitializeAsync();

    if (rdpFactory.CheckIfRLMMeetsHardwareRequirements(detectedRLM))
    {
        //found an RLM that meets the hardware requirements
        rlm = detectedRLM;
        break;
    }
    else
    {
        //uninitialize the RLM since it does not meet the requirements
        detectedRLM.Uninitialize();
    }
}

if (rlm == null)
{
    throw new Exception("Could not find RLM that meets the RD-P hardware requirements.");
}

// Step 4. Create the RD-P
rdp = rdpFactory.CreateRDP(rlm);

// Step 5. Initialize the RD-P
await rdp.InitializeAsync();
```

Step 1. Check that an RD-P camera is connected to the computer

We use the RD-P factory to check if an RD-P camera is connected to the computer. For simplicites sake, we throw an exception if a camera cannot be detected.

Step 2. Find all USB RLMs

We use the hardware detection service (*IHardwareDetectionService*) to find all the USB RLMs. These RLMs are candidates to use for the RD-P composite device. We use only USB connected RLMs for the sake of the sample. Etherent connected RLMs can also be used.

Step 3. Find the first RLM that meets the RD-P hardware requirements

We use the RD-P factory to evaluate each RLM to see if it meets the RD-P hardware requirements. To meet the hardware requirements, RLMs must be initialized, have either a 1300nm or 1310nm laser, and be either a dual-output RLM or have an OSX connected to the RLM. For the sake of the sample, we use the first RLM that meets the requirements and for simplicites sake, we throw an exception if no RLM can be found that meets the requirements. You can handle the RLM selection however you wish. As part of cleanup, we uninitialize any RLMs that do not meet the RD-P hardware requirements. For networked RLMs you should avoid initializing any RLMs that you do not have access to/possess as it will disrupt others who are using those devices.

Step 4. Create the RD-P

With the RLM selected, we use the RD-P factory to create the RD-P. This method will throw an exception if the RLM does not meet the hardware requirements or if no RD-P camera can be detected.

Step 5. Initialize the RD-P

We initialize the RD-P. It is now ready to be used.

Basic Usage- 12 Channel MPO

With the RD-P created and initialized, we can now use it to perform a polarity mapping operation. For basic usage, we use the *MapPolarityAsync* method to map the polarity. The operation will throw a polarity mapping failed exception (*PolarityMappingFailedException*) if not all of the fibers can be properly mapped.

```
// Basic Usage- 12 channel MPO
Console.WriteLine("Running basic polarity mapping...");

// Step 1. Set the RD-P mapping mode, mapping sensitivity, and loss compensation
rdp.SetMappingMode(EMappingMode.MPO);

rdp.SetMappingSensitivity(EMappingSensitivity.Connector);

await rdp.SetLossCompensationAsync(0);

//Step 2. Perform the polarity mapping
try
{
    polarityMapping = await rdp.MapPolarityAsync(12, new List<int?> { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12
});

    //show the polarity mapping
    Console.WriteLine("Basic polarity mapping: " + string.Join(",", polarityMapping));
}
catch(PolarityMappingFailedException ex)
{
    Console.WriteLine("Basic polarity mapping failed. Saving image...");

    //save an image of the failed polarity mapping
    ex.PolarityMappingImage.SaveImage("Basic Polarity Mapping Failed.png");

    Console.WriteLine("Image saved.");
}
```

Step 1. Set the RD-P mapping mode, mapping sensitivity, and loss compensation

We must set the polarity mapping mode, mapping sensitivity, and loss compensation before we run a polarity mapping operation. For this example, we are measuring a 12 channel MPO so we set the polarity mapping mode to MPO. For this example, we are measuring a DUT with a connector so we set the mapping sensitivity to Connector. For this example, we are measuring a DUT with a normal amount of loss in the setup so, for simplicities sake, we set the loss compensation to 0dB to use the standard camera settings.

Step 2. Perform the polarity mapping

We perform the polarity mapping operation using the *MapPolarityAsync* method. This handles the entire operation and returns the detected polarity mapping. For the sample, we assume the DUT has an A polarity and pass in an A polarity as the expected mapping to expand the range of scenarios where the RD-P can generate a mapping. The mapping can still fail and throw a polarity mapping failed exception (*PolarityMappingFailedException*). This exception contains an image showing the camera output of the failed mapping and a list of the detected fibers. Note, the list of detected fibers is not a mapping. For the sake of the sample, we save the image of any failed polarity mappings. You can handle a failed polarity mapping operation however you wish.

Advanced Usage: Segmented Polarity Mapping- 12 channel MPO

We can also use the RD-P to perform a segmented polarity mapping operation. In this case, it is up to us to control all hardware involved in the polarity mapping operation and to call the various RD-P methods to perform the operation. For this example, we are simply using the OSX switch in reverse, from channel 12 down to channel 1. For a segmented polarity operation, a 1300nm laser must be used for multimode and a 1310nm laser must be used for single mode.

```
// Advanced Usage: Segmented Polarity Mapping- 12 channel MPO
Console.WriteLine("Running segmented polarity mapping...");

// Step 1. Set the RD-P mapping mode and loss compensation
rdp.SetMappingMode(EMappingMode.MPO);

rdp.SetMappingSensitivity(EMappingSensitivity.Connector);

await rdp.SetLossCompensationAsync(0);

// Step 2. Start the polarity mapping operation
await rdp.StartMappingPolarityAsync(12);

// Step 3. Setup the RLM for the test
await rlm.SetInteractionModeAsync(EInteractionMode.Remote);

wavelength = rlm.Wavelengths.First(rlmWavelength => rlmWavelength == 1310 || rlmWavelength == 1300);

await rlm.TurnOnLaserAsync(wavelength);

// Step 4. Switch to the correct optical path for each fiber and collect the mapping data
await rlm.ChangeOutputChannelAsync(1);

for (int i = 0; i < 12; i++)
{
    //in this case, we are just running through the channels in reverse
    await rlm.ChangeSwitchChannelAsync(1, 12 - i);

    // When we collect the mapping data, the method returns whether a fiber was detected
    fiberDetected = await rdp.CollectOutputMappingDataAsync(i + 1);

    if (fiberDetected)
    {
        Console.WriteLine($"Fiber ${i + 1} detected.");
    }
    else
    {
        Console.WriteLine($"Fiber ${i + 1} not detected.");
    }
}

// Step 5. Cleanup from the test
await rlm.TurnOffLaserAsync();

// Step 6. Process the collected mapping data into a mapping
try
{
    polarityMapping = rdp.ProcessMappingData(new List<int?> { 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1 });


    //show the polarity mapping
    Console.WriteLine("Advanced polarity mapping: " + string.Join(",", polarityMapping));
}
catch (PolarityMappingFailedException ex)
{
    Console.WriteLine("Advanced polarity mapping failed. Saving image...");
```

```
    //save an image of the failed polarity mapping
    ex.PolarityMappingImage.SaveImage("Advanced Polarity Mapping Failed.png");

    Console.WriteLine("Image saved.");
}

// Step 7. End the polarity mapping operation
await rdp.EndMappingPolarityAsync();
```

Step 1. Set the RD-P mapping mode, mapping sensitivity, and loss compensation

As with the basic usage mode, we must set the polarity mapping mode, mapping sensitivity, and loss compensation before we run a polarity mapping operation. For this example, we are still measuring a 12 channel MPO so we set the polarity mapping mode to MPO. For this example, we are measuring a DUT with a connector so we set the mapping sensitivity to Connector. For this example, we are still measuring a DUT with a normal amount of loss in the setup so, for simplicities sake, we set the loss compensation to 0dB to use the standard camera settings.

Step 2. Start the polarity mapping operation

First, we start the polarity mapping operation. This will throw an exception if another polarity mapping operation is already in progress. Likewise, other operations will throw exceptions if a polarity mapping operation has not been started.

Step 3. Setup the RLM for the test

Next, we setup the RLM for the test. We set the interaction mode to *Remote* to prevent any RLM front screen operations from affecting the mapping operation. We also turn on the RLM laser for the test. For single mode, we must use a 1310nm laser and for multimode, we must use a 1300nm laser. We determine which of the appropriate wavelengths the RLM has and turn on that laser. We are now ready to run the polarity mapping operation.

Step 4. Switch to the correct optical path for each fiber and collect the mapping data

For the sake of the sample, we are running the OSX switch in reverse from channel 12 to channel 1. We switch all hardware to the correct optical path for the fiber. In this case, we only have the RLM and OSX in the optical path. We switch the RLM to output channel 1. We switch the OSX to the appropriate channel and collect the output mapping data for the fiber. When data is collected for the fiber, the method returns whether or not the fiber was detected. The output mapping data can be collected multiple times for each fiber if necessary. The fiber detection feedback can be used to make decisions regarding recapturing data and adjusting the loss compensation to improve detection of fibers. Data must be collected for each fiber before it can be processed and a mapping produced.

Step 5. Cleanup from the test

We cleanup the RLM from the test. We set the interaction mode back to *Local* and turn off the laser.

Step 6. Process the collected mapping data into a mapping

We now process the collected mapping data to produce a polarity mapping. A polarity mapping is returned by the method. Note, an exception is thrown if data has not been collected for all fibers. For the sample, we assume the DUT has an A polarity. Since we are running the switch from channel 12 to channel 1, we pass in a B polarity as the expected mapping to expand the range of scenarios where the RD-P can generate a mapping. The mapping can still fail and throw a polarity mapping failed exception (*PolarityMappingFailedException*). This exception contains an image showing the camera output of the failed mapping and a list of the detected fibers. Note, the list of detected fibers is not a mapping. For the sake of the sample, we save the image of any failed polarity mappings. You can handle a failed polarity mapping operation however you wish.

Step 7. End the polarity mapping operation

We finish by ending the operation. This will throw an exception if no polarity mapping operation is in progress.

Cleanup

The final step in using an RD-P is cleaning up after we are done using it.

```
// Cleanup
// Step 1. Uninitialize the RD-P.
rdp.Uninitialize();
```

Step 1. Uninitialize the RD-P

We cleanup by uninitializing the RD-P.

Other RD-P Usage

The RD-P provides additional functionality to support use cases that do not fall under the basic usage outlined in the previous section.

Custom Camera Settings

The RD-P is primarily designed for use with an RLM or PTM. The loss compensation settings adjust the camera settings to detect fibers for these devices. However, the RD-P also supports using custom camera gain and exposure settings. This allows other sources to be used with the RD-P.

Below is a small sample showing how to set the custom camera settings. Following the sample is an explanation with more detailed information about how it operates.

```
// Step 2. Set the RD-P custom camera settings
// Note: Gain is represented as an integer value from 0 to 480 where the value is the gain in dB multiplied by 10.
// Note: Exposure is represented as an integer value from -16 to 5 where the value is the exponent for equation: exposure = 2^x, in seconds.
await rdp.SetCustomCameraSettingsAsync(250, -5);
```

Gain

The camera gain can be adjusted between 0dB and 48dB. It is represented as a value between 0 and 480 where the value is the gain in dB multiplied by 10. For example, a value of 100 corresponds to a gain of 10dB.

Exposure

The camera exposure can be adjusted between ~15µs and 32s. It is represetned as a value between -16 and 5 where the value is the exponent, x, in the equation: exposure = 2^x, in seconds. For example, a value of -1 corresponds to an exposure of 0.5s.

Camera Mode

The RD-P has a camera mode setting that controls whether it uses the loss compensations settings or the custom settings for the camera gain and exposure. This can be set explcitly using the *SetCameraModeAsync* method. However, setting the loss compensation or the custom camera settings will automatically toggle the RD-P to the corresponding camera mode, so there is no need to explicitly set the camera mode in most use cases.

External Source RD-P

As previously mentioned, the RD-P is primarily designed for use with an RLM or PTM. However, it also supports using an external source. An external source RD-P can be created with an RD-P factory (*IRDPFactory*) using the *CreateRDPWithExternalSource* method. The external source RD-P does not require a PTM or RLM in the system.

Note: The external source RD-P only supports the segmented mapping advanced usage mode. Trying to perform a basic polarity mapping operation using the *MapPolarityAsync* method will throw a not supported exception (*NotSupportedException*)

When creating an external source RD-P, you must provide the wavelength, the number of channels, and an optional loss compensation configuration to use (RLM or PTM) for the RD-P.

Below is a small sample showing creation and initialization of an external source RD-P.

```
// Step 1. Check that an RD-P camera is connected to the computer.
if (!rdpFactory.CheckForCamera())
{
    throw new Exception("Could not find RD-P.");
}

// Step 2. Create the external source RD-P
rdp = rdpFactory.CreateRDPWithExternalSource(1310, 24, ELossCompensationConfiguration.RLM);

// Step 3. Initialize the RD-P
await rdp.InitializeAsync();
```

Wavelength

The wavelength of the source used for the RD-P. This is an informational field and does not impact the performance of the RD-P at all. The wavelength must be greater than 0.

Number of Channels

The number of channels for the RD-P. This is the number of channels that can be provided by the external setup used with the RD-P. The number of channels must be greater than 1.

Loss Compensation Configuration

The optional loss configuration specifies what configuration to use in loss compensation mode. If no configuration is specified, it defaults to the RLM loss compensation configuration. This setting allows the use of loss compensation mode if the external source is similar to either the RLM or PTM sources. Otherwise, custom camera settings must be used to set appropriate gain and exposure settings for the external source.

Retrieving RD-P Detector Images

The RD-P supports retrieving the detector images after the mapping data has been collected. This can be helpful in scenarios like using custom camera settings as the images provide more insight nito the mapping process.

Below is a small sample showing how to retrieve the images. Following the sample is an explanation with more detailed information.

```
//Get the mapping images
detectorImages = rdp.GetOutputMappingImages(i + 1);

for(int j = 0; j < detectorImages.Count; j++)
{
    //Save the Image
    _ = detectorImages[j].SaveImage($"Custom Settings Polarity Mapping - Fiber {i + 1} - Detector {j + 1}.png");

    Console.WriteLine($"Saved image for Fiber {i + 1} - Detector {j + 1}.");

    //Dispose the image to free the resources
    detectorImages[j].Dispose();
}
```

Mapping data must be collected for an output before the images can be retrieved. Otherwise, an *InvalidOperationException* will be thrown when the *GetOutputMappingImages* method is called. The images are returned as a list of *Mat* objects, one for each detector in the RD-P. These can then be saved or converted into other image formats for display.

Note: *Mat* objects must be properly disposed after use to free up their resources.

Santec Hardware API

Here is a detailed explanation of all available commands in the Santec Hardware Library. Most commands correspond to SCPI commands found in the user manuals available at https://inst.santec.com/resources/usermanuals.

Namespace Santec

Classes

Version

Represents a Santec version number.

Class Version

Represents a Santec version number.

Inheritance

System.Object
Version

Implements

System.IComparable<Version>
System.IEquatable<Version>

Inherited Members

System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Santec

Assembly: Santec.Hardware.dll

Syntax

```
public class Version : IComparable<Version>, IEquatable<Version>
```

Constructors

Version(Int32, Int32, Int32)

Initializes a new version number.

Declaration

```
public Version(int major, int minor = 0, int revision = 0)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	major	The major version number.
System.Int32	minor	The minor version number.
System.Int32	revision	The revision number.

Exceptions

TYPE	CONDITION
System.ArgumentOutOfRangeException	Thrown if the major, minor, or revision number is less than zero.

Version(String)

Initializes a new version number.

Declaration

```
public Version(string version)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	version	The string representation of the version number.

Properties

Major

Get the major version number.

Declaration

```
public int Major { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	The major version number.

Minor

Get the minor version number.

Declaration

```
public int Minor { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	The minor version number.

Revision

Get the revision number.

Declaration

```
public int Revision { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	The revision number.

Methods

CompareTo(Version)

Declaration

```
public int CompareTo(Version other)
```

Parameters

TYPE	NAME	DESCRIPTION
Version	other	

Returns

TYPE	DESCRIPTION
System.Int32	

Equals(Version)

Declaration

```
public bool Equals(Version other)
```

Parameters

TYPE	NAME	DESCRIPTION
Version	other	

Returns

TYPE	DESCRIPTION
System.Boolean	

Equals(Object)

Declaration

```
public override bool Equals(object o)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Object	o	

Returns

TYPE	DESCRIPTION
System.Boolean	

Overrides

System.Object.Equals(System.Object)

GetHashCode()

Declaration

```
public override int GetHashCode()
```

Returns

TYPE	DESCRIPTION
System.Int32	

Overrides

System.Object.GetHashCode()

Parse(String)

Converts the string representation of a version number to the equivalent Version object.

Declaration

```
public static Version Parse(string input)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	input	The string representation of the version number.

Returns

TYPE	DESCRIPTION
Version	A version number of the string.

Exceptions

TYPE	CONDITION
System.ArgumentNullException	Thrown if the input is `null`.
System.ArgumentException	Thrown if the input does not have three components.
System.ArgumentException	Thrown if one or more input components is not an integer.
System.ArgumentException	Thrown if one or more input components is less than zero.

ToString()

Converts the version number to its string representation.

Declaration

```
public override string ToString()
```

Returns

TYPE	DESCRIPTION
System.String	The string representation of the version number.

Overrides

System.Object.ToString()

TryParse(String, out Version)

Tries to converts the string representation of a version number to the equivalent Version object and returns whether or not the operation succeeded.

Declaration

```
public static bool TryParse(string input, out Version result)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	input	The string representation of the version number.
Version	result	The resulting Version object. `null` if the operation fails.

Returns

TYPE	DESCRIPTION
System.Boolean	`true` if the conversion succeeded; otherwise, `false`

Operators

Equality(Version, Version)

Declaration

```
public static bool operator ==(Version operand1, Version operand2)
```

Parameters

TYPE	NAME	DESCRIPTION
Version	operand1	
Version	operand2	

Returns

TYPE	DESCRIPTION
System.Boolean	

GreaterThan(Version, Version)

Declaration

```
public static bool operator>(Version operand1, Version operand2)
```

Parameters

TYPE	NAME	DESCRIPTION
Version	operand1	
Version	operand2	

Returns

TYPE	DESCRIPTION
System.Boolean	

GreaterThanOrEqual(Version, Version)

Declaration

```
public static bool operator >=(Version operand1, Version operand2)
```

Parameters

TYPE	NAME	DESCRIPTION
Version	operand1	
Version	operand2	

Returns

TYPE	DESCRIPTION
System.Boolean	

Inequality(Version, Version)

Declaration

```
public static bool operator !=(Version operand1, Version operand2)
```

Parameters

TYPE	NAME	DESCRIPTION
Version	operand1	
Version	operand2	

Returns

TYPE	DESCRIPTION
System.Boolean	

LessThan(Version, Version)

Declaration

```
public static bool operator <(Version operand1, Version operand2)
```

Parameters

TYPE	NAME	DESCRIPTION
Version	operand1	
Version	operand2	

Returns

TYPE	DESCRIPTION
System.Boolean	

LessThanOrEqual(Version, Version)

Declaration

```
public static bool operator <=(Version operand1, Version operand2)
```

Parameters

TYPE	NAME	DESCRIPTION
Version	operand1	
Version	operand2	

Returns

TYPE	DESCRIPTION
System.Boolean	

Implements

System.IComparable<T>
System.IEquatable<T>

Namespace Santec.Hardware.AssemblyResolvers

Classes

VisaResolver

Provides an assembly resolver that can resolve references to the Ivi.Visa assembly.

Class VisaResolver

Provides an assembly resolver that can resolve references to the Ivi.Visa assembly.

Inheritance

System.Object
VisaResolver

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.AssemblyResolvers

Assembly: Santec.Hardware.dll

Syntax

```
public static class VisaResolver
```

Methods

Initialize()

Initialize the assembly resolver to resolve references to the Ivi.Visa assembly.

Declaration

```
public static void Initialize()
```

Namespace Santec.Hardware.Converters

Classes

FiberTypeToStringConverter

Provides a converter to convert fiber types to strings.

Class FiberTypeToStringConverter

Provides a converter to convert fiber types to strings.

Inheritance

System.Object
FiberTypeToStringConverter

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Converters

Assembly: Santec.Hardware.dll

Syntax

```
public class FiberTypeToStringConverter
```

Constructors

FiberTypeToStringConverter(ICatalogProvider)

Initializes a new fiber type to string converter.

Declaration

```
public FiberTypeToStringConverter(ICatalogProvider catalogProvider)
```

Parameters

TYPE	NAME	DESCRIPTION
Santec.Core.Localization.ICatalogProvider	catalogProvider	The catalog provider for localization.

Methods

Convert(EFiberType, Boolean, Boolean)

Convert the fiber type.

Declaration

```
public string Convert(EFiberType fiberType, bool capitalize, bool translate = true)
```

Parameters

TYPE	NAME	DESCRIPTION
EFiberType	fiberType	The fiber type to convert.

TYPE	NAME	DESCRIPTION
System.Boolean	capitalize	Whether to capitalize the string.
System.Boolean	translate	Whether to translate the string or not. Default value is `true`.

Returns

TYPE	DESCRIPTION
System.String	The string representation of the fiber type.

Namespace Santec.Hardware.Devices.Attenuation

Classes

OVA

Provides a connection to an OVA along with properties of the device and operations that can be performed using it.

> **NOTE**
>
> This class cannot not be used directly. It can only be used when returned:
>
> 1. As part of a list of devices by a DetectHardwareAsync(EConnectionTypesToDetect) or DetectHardwareAsync<T>(EConnectionTypesToDetect) call.
> 2. As a device by a TryDetectIPDeviceAsync(IPAddress) or TryDetectIPDeviceAsync<T>(IPAddress) call.

Interfaces

IOVA

Defines the properties of an OVA and the operations that can be performed with a connection to the device.

Enums

EBeamBlockState

Specifies the beam block state of a device

EOVAModel

Specifies the model of an OVA.

Enum EBeamBlockState

Specifies the beam block state of a device

Namespace: Santec.Hardware.Devices.Attenuation

Assembly: Santec.Hardware.dll

Syntax

```
public enum EBeamBlockState
```

Fields

NAME	DESCRIPTION
Off	Beam block off.
On	Beam block on.

Enum EOVAModel

Specifies the model of an OVA.

Namespace: Santec.Hardware.Devices.Attenuation

Assembly: Santec.Hardware.dll

Syntax

```
public enum EOVAModel
```

Fields

NAME	DESCRIPTION
OVA100	The OVA-100 model.
OVB100	The OVB-100 model.

Interface IOVA

Defines the properties of an OVA and the operations that can be performed with a connection to the device.

Inherited Members

ISantecDevice.GetIPAddressAsync(CancellationToken)
ISantecDevice.SetIPAddressAsync(IPAddress, CancellationToken)
ISantecDevice.EnableDHCPAsync(CancellationToken)
ISantecDevice.GetGatewayAsync(CancellationToken)
ISantecDevice.SetGatewayAsync(IPAddress, CancellationToken)
ISantecDevice.GetSubnetPrefixLengthAsync(CancellationToken)
ISantecDevice.SetSubnetPrefixLengthAsync(Int32, CancellationToken)
ISantecDevice.GetHostnameAsync(CancellationToken)
ISantecDevice.SetHostnameAsync(String, CancellationToken)
ISantecDevice.GetInteractionModeAsync(CancellationToken)
ISantecDevice.SetInteractionModeAsync(EInteractionMode, CancellationToken)
IModuleController.NumberOfModules
IDevice<EOVAModel>.Model
IDevice.Name
IDevice.SerialNumber
IDevice.FirmwareVersion
IDevice.Address
IDevice.IsInitialized
IDevice.InitializeAsync()
IDevice.Uninitialize()
IDevice.GetTimeout()
IDevice.SetTimeout(ETimeout)
IDevice.ResetAsync()
IDevice.RefreshAsync()
IDevice.PingAsync()
IDevice.QueryAsync(String, CancellationToken)
IDevice.WriteAsync(String, CancellationToken)
IDevice.ReadAsync(CancellationToken)
System.IDisposable.Dispose()

Namespace: Santec.Hardware.Devices.Attenuation

Assembly: Santec.Hardware.dll

Syntax

```
public interface IOVA : ISantecDevice, IModuleController, IDevice<EOVAModel>, IDevice, IDisposable
```

Properties

Modules

Get the modules of the attenuator.

> **NOTE**
>
> The OVA must be initialized before this property contains the proper values.

Declaration

```
IReadOnlyList<IAttenuatorModule> Modules { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<IAttenuatorModule>	A list of the modules the attenuator contains.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the property is accessed before the OVA has been initialized.

Methods

ChangeAllModuleAttenuationsAsync(Double, CancellationToken)

Asynchronously change the attenuation of all modules.

Declaration

```
Task ChangeAllModuleAttenuationsAsync(double attenuation, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Double	attenuation	The attenuation for all modules.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	The task representing the asynchronous operation.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the specified attenuation is out of range for any of the OVA modules.
System.InvalidOperationException	Thrown when the method is called on an uninitialized OVA.
UnexpectedDeviceResponseException	Thrown when the OVA change all attenuations response does not match the expected response.

TYPE	CONDITION
DeviceTimeoutException	Thrown when a communication failure causes no response to be received from the OVA.

ChangeModuleAttenuationAsync(Int32, Double, CancellationToken)

Asynchronously change the attenuation of a given attenuator module.

Declaration

```
Task ChangeModuleAttenuationAsync(int moduleIndex, double att, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	moduleIndex	The index of the module to change the attenuation of.
System.Double	att	The attenuation to switch to.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the specified module is out of range for the OVA.
System.ArgumentException	Thrown when the specified attenuation is out of range for the OVA module.
System.InvalidOperationException	Thrown when the method is called on an uninitialized OVA.

TYPE	CONDITION
UnexpectedDeviceResponseException	Thrown when the OVA module attenuation response does not match the expected response.
DeviceTimeoutException	Thrown when a communication failure causes no response to be received from the OVA.

GetModuleAttenuationAsync(Int32, CancellationToken)

Asynchronously get the current attenuation of a given switch module.

Declaration

```
Task<double> GetModuleAttenuationAsync(int moduleIndex, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	moduleIndex	The index of the module to get the current attenuation for.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the specified module is out of range for the OVA.
System.InvalidOperationException	Thrown when the method is called on an uninitialized OVA.
UnexpectedDeviceResponseException	Thrown when the OVA module attenuation response is not a valid module attenuation.

TYPE	CONDITION
DeviceTimeoutException	Thrown when a communication failure causes no response to be received from the OVA.

GetModuleBeamBlockStateAsync(Int32, CancellationToken)

Asynchronously get the beam block state of an attenuator module.

Declaration

```
Task<EBeamBlockState> GetModuleBeamBlockStateAsync(int moduleIndex, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	moduleIndex	The index of the module to get the beam block state for
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<EBeamBlockState>	

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the specified module is out of range for the OVA.
System.InvalidOperationException	Thrown when the method is called on an uninitialized OVA.
UnexpectedDeviceResponseException	Thrown when the OVA response does not correspond to a beam block state.
DeviceTimeoutException	Thrown when a communication failure causes no response to be received from the OVA.

MoveModuleStepAsync(Int32, Int32, CancellationToken)

Asynchronously change the step position of an attenuator module. Do not use this command unless specifically instructed by a

santec engineer.

Declaration

```
Task MoveModuleStepAsync(int moduleIndex, int step, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	moduleIndex	The index of the module to change the attenuation of.
System.Int32	step	The step to move to
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the specified module is out of range for the OVA.
System.ArgumentException	Thrown when the specified attenuation is out of range for the OVA module.
System.InvalidOperationException	Thrown when the method is called on an uninitialized OVA.
UnexpectedDeviceResponseException	Thrown when the OVA module attenuation response does not match the expected response.
DeviceTimeoutException	Thrown when a communication failure causes no response to be received from the OVA.

SetModuleBeamBlockStateAsync(Int32, EBeamBlockState, CancellationToken)

Asynchronously change the beam block state of an attenuator module.

Declaration

```
Task SetModuleBeamBlockStateAsync(int moduleIndex, EBeamBlockState beamBlockState, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	moduleIndex	The index of the module to change the beam block state for
EBeamBlockState	beamBlockState	The state to set the attenuator module's beam block to.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the specified module is out of range for the OVA.
System.InvalidOperationException	Thrown when the method is called on an uninitialized OVA.
UnexpectedDeviceResponseException	Thrown when the OVA module beam block state response does not match the expected response.
DeviceTimeoutException	Thrown when a communication failure causes no response to be received from the OVA.

SetModuleCalibrationAsync(Int32, Int32, Int32, CancellationToken)

Asynchronously set a calibration factor for a module

Declaration

```
Task SetModuleCalibrationAsync(int moduleIndex, int index, int value, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	moduleIndex	The index of the module to change the calibration of
System.Int32	index	The index of the cal point
System.Int32	value	The cal value
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the specified module is out of range for the OVA.
System.ArgumentException	Thrown when the specified attenuation is out of range for the OVA module.
System.InvalidOperationException	Thrown when the method is called on an uninitialized OVA.
UnexpectedDeviceResponseException	Thrown when the OVA module attenuation response does not match the expected response.
DeviceTimeoutException	Thrown when a communication failure causes no response to be received from the OVA.

Class OVA

Provides a connection to an OVA along with properties of the device and operations that can be performed using it.

> **NOTE**
>
> This class cannot not be used directly. It can only be used when returned:
>
> 1. As part of a list of devices by a DetectHardwareAsync(EConnectionTypesToDetect) or DetectHardwareAsync<T>(EConnectionTypesToDetect) call.
> 2. As a device by a TryDetectIPDeviceAsync(IPAddress) or TryDetectIPDeviceAsync<T>(IPAddress) call.

Inheritance

System.Object
PhysicalDevice
SantecDevice
OVA

Implements

IOVA
ISantecDevice
IModuleController
IDevice<EOVAModel>
IDevice
System.IDisposable

Inherited Members

SantecDevice.HandleIDNResponse(String[])
SantecDevice.GetIPAddressAsync(CancellationToken)
SantecDevice.SetIPAddressAsync(IPAddress, CancellationToken)
SantecDevice.EnableDHCPAsync(CancellationToken)
SantecDevice.GetGatewayAsync(CancellationToken)
SantecDevice.SetGatewayAsync(IPAddress, CancellationToken)
SantecDevice.GetSubnetPrefixLengthAsync(CancellationToken)
SantecDevice.SetSubnetPrefixLengthAsync(Int32, CancellationToken)
SantecDevice.GetHostnameAsync(CancellationToken)
SantecDevice.SetHostnameAsync(String, CancellationToken)
SantecDevice.GetInteractionModeAsync(CancellationToken)
SantecDevice.SetInteractionModeAsync(EInteractionMode, CancellationToken)
PhysicalDevice.Address
PhysicalDevice.SerialNumber
PhysicalDevice.FirmwareVersion
PhysicalDevice.IsInitialized
PhysicalDevice.IsInitializing
PhysicalDevice.SetTimeout(ETimeout)
PhysicalDevice.GetTimeout()
PhysicalDevice.Dispose()
PhysicalDevice.Dispose(Boolean)
PhysicalDevice.PingAsync()
PhysicalDevice.ResetAsync()
PhysicalDevice.QueryAsync(String, CancellationToken)
PhysicalDevice.WriteAsync(String, CancellationToken)
PhysicalDevice.ReadAsync(CancellationToken)

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.Attenuation

Assembly: Santec.Hardware.dll

Syntax

```
public class OVA : SantecDevice, IOVA, ISantecDevice, IModuleController, IDevice<EOVAModel>, IDevice,
IDisposable
```

Properties

Model

Declaration

```
public EOVAModel Model { get; }
```

Property Value

TYPE	DESCRIPTION
EOVAModel	

Modules

Declaration

```
public IReadOnlyList<IAttenuatorModule> Modules { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<IAttenuatorModule>	

Name

Declaration

```
public override string Name { get; }
```

Property Value

TYPE	DESCRIPTION
System.String	

Overrides

PhysicalDevice.Name

Remarks

> **NOTE**
>
> This property will always return "OVA".

NumberOfModules

Declaration

```
public int NumberOfModules { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

Methods

ChangeAllModuleAttenuationsAsync(Double, CancellationToken)

Declaration

```
public async Task ChangeAllModuleAttenuationsAsync(double attenuation, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Double	attenuation	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

ChangeModuleAttenuationAsync(Int32, Double, CancellationToken)

Declaration

```
public Task ChangeModuleAttenuationAsync(int moduleIndex, double attenuation, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	moduleIndex	
System.Double	attenuation	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

GetModuleAttenuationAsync(Int32, CancellationToken)

Declaration

```
public Task<double> GetModuleAttenuationAsync(int moduleIndex, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	moduleIndex	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

GetModuleBeamBlockStateAsync(Int32, CancellationToken)

Declaration

```
public Task<EBeamBlockState> GetModuleBeamBlockStateAsync(int moduleIndex, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	moduleIndex	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<EBeamBlockState>	

InitializeAsync()

Declaration

```
public override async Task InitializeAsync()
```

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Overrides

SantecDevice.InitializeAsync()

MoveModuleStepAsync(Int32, Int32, CancellationToken)

Declaration

```
public async Task MoveModuleStepAsync(int moduleIndex, int step, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	moduleIndex	
System.Int32	step	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

RefreshAsync()

Declaration

```
public override async Task RefreshAsync()
```

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Overrides

PhysicalDevice.RefreshAsync()

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized RLM.
UnexpectedDeviceResponseException	Thrown when the OVA modules response is not a valid module count.
UnexpectedDeviceResponseException	Thrown when any OVA module information response contains invalid channel or common channel counts.

SetModuleBeamBlockStateAsync(Int32, EBeamBlockState, CancellationToken)

Declaration

```
public Task SetModuleBeamBlockStateAsync(int moduleIndex, EBeamBlockState beamBlockState, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	moduleIndex	
EBeamBlockState	beamBlockState	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetModuleCalibrationAsync(Int32, Int32, Int32, CancellationToken)

Declaration

```
public async Task SetModuleCalibrationAsync(int moduleIndex, int index, int value, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	moduleIndex	
System.Int32	index	
System.Int32	value	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Uninitialize()

Declaration

```
public override void Uninitialize()
```

Overrides

PhysicalDevice.Uninitialize()

Explicit Interface Implementations

IModuleController.Modules

Declaration

```
IReadOnlyList<IModule> IModuleController.Modules { get; }
```

Returns

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<IModule>	

Implements

IOVA
ISantecDevice
IModuleController
IDevice<TModelType>
IDevice
System.IDisposable

Namespace Santec.Hardware.Devices.Backreflection

Classes

BRM

Provides a connection to a BRM along with properties of the device and operations that can be performed using it.

> **NOTE**
>
> This class cannot not be used directly. It can only be used when returned:
>
> 1. As part of a list of devices by a DetectHardwareAsync(EConnectionTypesToDetect) or DetectHardwareAsync<T>(EConnectionTypesToDetect) call.
> 2. As a device by a TryDetectIPDeviceAsync(IPAddress) or TryDetectIPDeviceAsync<T>(IPAddress) call.

SimulatedBRM

Provides a simulated BRM with configurable reading values.

> **NOTE**
>
> This class is intended for testing/development purposes. It operates much quicker than an actual BRM. It can only be created using a SimulatedDeviceFactory. It should only be used when returned as part of a list of devices by a DetectHardwareAsync(EConnectionTypesToDetect) or DetectHardwareAsync<T>(EConnectionTypesToDetect) call to a SimulatedHardwareDetectionService. Each reading value contains a default value range but can be configured via the appropriate configuration method.

Interfaces

IBackreflectionMeter

Defines the properties of a backreflection meter and the operations that can be performed with a connection to the device.

IBRM

Defines the properties of a BRM and operations that can be performed with a connection to the device.

Enums

EBRMModel

Specifies the model of a BRM.

Class BRM

Provides a connection to a BRM along with properties of the device and operations that can be performed using it.

> **NOTE**
>
> This class cannot not be used directly. It can only be used when returned:
>
> 1. As part of a list of devices by a DetectHardwareAsync(EConnectionTypesToDetect) or DetectHardwareAsync<T>(EConnectionTypesToDetect) call.
> 2. As a device by a TryDetectIPDeviceAsync(IPAddress) or TryDetectIPDeviceAsync<T>(IPAddress) call.

Inheritance

System.Object
PhysicalDevice
SantecDevice
BRM

Implements

IBRM
ISantecDevice
IDiscretePowerMeter
IBackreflectionMeter
IDetectorDeviceWithDarking
ILaserSourceWithMonitor
ILaserSource
IWavelengthDevice
ISwitchController
IFiberDevice
IPowerMeterWithReferencePersistence
IPowerMeter
IInsertionLossMeter
IDetectorDevice
IDevice<EBRMModel>
IDevice
System.IDisposable

Inherited Members

SantecDevice.HandleIDNResponse(String[])
SantecDevice.GetIPAddressAsync(CancellationToken)
SantecDevice.SetIPAddressAsync(IPAddress, CancellationToken)
SantecDevice.EnableDHCPAsync(CancellationToken)
SantecDevice.GetGatewayAsync(CancellationToken)
SantecDevice.SetGatewayAsync(IPAddress, CancellationToken)
SantecDevice.GetSubnetPrefixLengthAsync(CancellationToken)
SantecDevice.SetSubnetPrefixLengthAsync(Int32, CancellationToken)
SantecDevice.GetHostnameAsync(CancellationToken)
SantecDevice.SetHostnameAsync(String, CancellationToken)
SantecDevice.GetInteractionModeAsync(CancellationToken)
SantecDevice.SetInteractionModeAsync(EInteractionMode, CancellationToken)
PhysicalDevice.Address
PhysicalDevice.SerialNumber
PhysicalDevice.FirmwareVersion

PhysicalDevice.IsInitialized
PhysicalDevice.IsInitializing
PhysicalDevice.SetTimeout(ETimeout)
PhysicalDevice.GetTimeout()
PhysicalDevice.Dispose()
PhysicalDevice.Dispose(Boolean)
PhysicalDevice.PingAsync()
PhysicalDevice.ResetAsync()
PhysicalDevice.QueryAsync(String, CancellationToken)
PhysicalDevice.WriteAsync(String, CancellationToken)
PhysicalDevice.ReadAsync(CancellationToken)
System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.Backreflection

Assembly: Santec.Hardware.dll

Syntax

```
public class BRM : SantecDevice, IBRM, ISantecDevice, IDiscretePowerMeter, IBackreflectionMeter,
IDetectorDeviceWithDarking, ILaserSourceWithMonitor, ILaserSource, IWavelengthDevice, ISwitchController,
IFiberDevice, IPowerMeterWithReferencePersistence, IPowerMeter, IInsertionLossMeter, IDetectorDevice,
IDevice<EBRMModel>, IDevice, IDisposable
```

Properties

Detectors

Declaration

```
public IReadOnlyList<IModule> Detectors { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<IModule>	

Fiber

Declaration

```
public FiberInformation Fiber { get; }
```

Property Value

TYPE	DESCRIPTION
FiberInformation	

Model

Declaration

```
public EBRMModel Model { get; }
```

Property Value

TYPE	DESCRIPTION
EBRMModel	

Name

Declaration

```
public override string Name { get; }
```

Property Value

TYPE	DESCRIPTION
System.String	

Overrides

PhysicalDevice.Name

Remarks

> **NOTE**
>
> This property will always return "BRM".

NumberOfDetectors

Declaration

```
public int NumberOfDetectors { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

NumberOfOutputs

Declaration

```
public int NumberOfOutputs { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

RequiresILReferenceSave

Declaration

```
public bool RequiresILReferenceSave { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	

SupportsTurningOffLaser

Declaration

```
public bool SupportsTurningOffLaser { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	

Remarks

> **NOTE**
>
> This property will always return `true`.

Switches

Declaration

```
public List<DeviceSwitch> Switches { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.List<DeviceSwitch>	

Wavelengths

Declaration

```
public List<int> Wavelengths { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.List<System.Int32>	

Methods

ChangeMultipleSwitchesChannelsAsync(IEnumerable<DeviceSwitchAndChannelPair>, CancellationToken)

Declaration

```
public Task ChangeMultipleSwitchesChannelsAsync(IEnumerable<DeviceSwitchAndChannelPair>
deviceSwitchesAndChannels, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Collections.Generic.IEnumerable<DeviceSwitchAndChannelPair>	deviceSwitchesAndChannels	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

ChangeOutputChannelAsync(Int32, CancellationToken)

Declaration

```
public Task ChangeOutputChannelAsync(int output, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	output	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

ChangeSwitchChannelAsync(Int32, Int32, CancellationToken)

Declaration

```
public Task ChangeSwitchChannelAsync(int switchIndex, int channel, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	switchIndex	
System.Int32	channel	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

TYPE	DESCRIPTION

DarkAllDetectorsAsync(CancellationToken)

Declaration

```
public Task DarkAllDetectorsAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

DarkBRDetectorAsync(CancellationToken)

Declaration

```
public Task DarkBRDetectorAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

DarkDetectorAsync(Int32, CancellationToken)

Declaration

```
public Task DarkDetectorAsync(int detector, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

GetActiveLaserAsync(CancellationToken)

Declaration

```
public Task<int> GetActiveLaserAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Int32>	

GetBRReferenceAsync(Int32, CancellationToken)

Declaration

```
public Task<double> GetBRReferenceAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

GetDetectorDarkStateAsync(Int32, CancellationToken)

Declaration

```
public Task<EDarkState> GetDetectorDarkStateAsync(int detector, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<EDarkState>	

GetFactoryPowerAsync(Int32, Int32, CancellationToken)

Declaration

```
public Task<FactoryPowerReading> GetFactoryPowerAsync(int output, int wavelength, CancellationToken
cancellationToken)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	output	
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<FactoryPowerReading>	

GetILCompensationEnabledAsync(CancellationToken)

Declaration

```
public async Task<bool> GetILCompensationEnabledAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Boolean>	

GetILReferenceAsync(Int32, Int32, CancellationToken)

Declaration

```
public Task<double> GetILReferenceAsync(int detector, int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

GetOutputChannelAsync(CancellationToken)

Declaration

```
public Task<int> GetOutputChannelAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Int32>	

GetSwitchChannelAsync(Int32, CancellationToken)

Declaration

```
public Task<int> GetSwitchChannelAsync(int switchIndex, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	switchIndex	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Int32>	

InitializeAsync()

Declaration

```
public override async Task InitializeAsync()
```

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Overrides

SantecDevice.InitializeAsync()

Exceptions

TYPE	CONDITION
UnexpectedDeviceResponseException	Thrown when the BRM wavelengths response contains one or more wavelengths that are not valid values.
UnexpectedDeviceResponseException	Thrown when the BRM detectors response is not a valid number.
UnexpectedDeviceResponseException	Thrown when the BRM switches response contains one or more switch channel counts that are not valid values.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the BRM.

ReadBRAsync(Int32, CancellationToken)

Declaration

```
public Task<ReadingValue> ReadBRAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<ReadingValue>	

ReadILAsync(Int32, Int32, CancellationToken)

Declaration

```
public Task<double> ReadILAsync(int detector, int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

ReadMonitorPowerAsync(Int32, CancellationToken)

Declaration

```
public Task<double> ReadMonitorPowerAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

ReadMultipleILsAsync(IEnumerable<Int32>, Int32, CancellationToken)

Declaration

```
public Task<List<double>> ReadMultipleILsAsync(IEnumerable<int> detectors, int wavelength, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Collections.Generic.IEnumerable<System.Int32>	detectors	
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Collections.Generic.List<System.Double>>	

ReadMultiplePowersAsync(IEnumerable<Int32>, Int32, CancellationToken)

Declaration

```
public Task<List<double>> ReadMultiplePowersAsync(IEnumerable<int> detectors, int wavelength,
CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Collections.Generic.IEnumerable<System.Int32>	detectors	
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Collections.Generic.List<System.Double>>	

ReadPowerAsync(Int32, Int32, CancellationToken)

Declaration

```
public Task<double> ReadPowerAsync(int detector, int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

ReferenceBRAsync(Int32, CancellationToken)

Declaration

```
public Task<double> ReferenceBRAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

ReferenceILAsync(Int32, Int32, Boolean, CancellationToken)

Declaration

```
public Task<double> ReferenceILAsync(int detector, int wavelength, bool applyReferenceToAllDetectors,
CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Int32	wavelength	
System.Boolean	applyReferenceToAllDetectors	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

ReferenceILAsync(Int32, Int32, CancellationToken)

Declaration

```
public Task<double> ReferenceILAsync(int detector, int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

RefreshAsync()

Declaration

```
public override async Task RefreshAsync()
```

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Overrides

PhysicalDevice.RefreshAsync()

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized BRM.
UnexpectedDeviceResponseException	Thrown when the BRM wavelengths response contains one or more wavelengths that are not valid values.
UnexpectedDeviceResponseException	Thrown when the BRM detectors response is not a valid number.
UnexpectedDeviceResponseException	Thrown when the BRM switches response contains one or more switch channel counts that are not valid values.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the BRM.

SaveILReferencesAsync(CancellationToken)

Declaration

```
public async Task SaveILReferencesAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetBRReferenceAsync(Int32, Double, CancellationToken)

Declaration

```
public Task SetBRReferenceAsync(int wavelength, double reference, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	
System.Double	reference	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetILCompensationEnabledAsync(Boolean, CancellationToken)

Declaration

```
public async Task SetILCompensationEnabledAsync(bool enabled, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Boolean	enabled	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetILReferenceAsync(Int32, Int32, Double, CancellationToken)

Declaration

```
public Task SetILReferenceAsync(int detector, int wavelength, double reference, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Int32	wavelength	
System.Double	reference	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

TurnOffLaserAsync(CancellationToken)

Declaration

```
public Task TurnOffLaserAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

TurnOnLaserAsync(Int32, CancellationToken)

Declaration

```
public Task TurnOnLaserAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Uninitialize()

Declaration

```
public override void Uninitialize()
```

Overrides

PhysicalDevice.Uninitialize()

Implements

IBRM
ISantecDevice
IDiscretePowerMeter
IBackreflectionMeter
IDetectorDeviceWithDarking
ILaserSourceWithMonitor
ILaserSource
IWavelengthDevice
ISwitchController
IFiberDevice
IPowerMeterWithReferencePersistence
IPowerMeter
IInsertionLossMeter
IDetectorDevice
IDevice<TModelType>
IDevice
System.IDisposable

Enum EBRMModel

Specifies the model of a BRM.

Namespace: Santec.Hardware.Devices.Backreflection

Assembly: Santec.Hardware.dll

Syntax

```
public enum EBRMModel
```

Fields

NAME	DESCRIPTION
BRM100	The BRM-100 model.

Interface IBackreflectionMeter

Defines the properties of a backreflection meter and the operations that can be performed with a connection to the device.

Inherited Members

IWavelengthDevice.Wavelengths
IDetectorDeviceWithDarking.DarkAllDetectorsAsync(CancellationToken)
IDetectorDeviceWithDarking.DarkDetectorAsync(Int32, CancellationToken)
IDetectorDeviceWithDarking.GetDetectorDarkStateAsync(Int32, CancellationToken)
IDetectorDevice.NumberOfDetectors
IDetectorDevice.Detectors
IDevice.Name
IDevice.SerialNumber
IDevice.FirmwareVersion
IDevice.Address
IDevice.IsInitialized
IDevice.InitializeAsync()
IDevice.Uninitialize()
IDevice.GetTimeout()
IDevice.SetTimeout(ETimeout)
IDevice.ResetAsync()
IDevice.RefreshAsync()
IDevice.PingAsync()
IDevice.QueryAsync(String, CancellationToken)
IDevice.WriteAsync(String, CancellationToken)
IDevice.ReadAsync(CancellationToken)
System.IDisposable.Dispose()

Namespace: Santec.Hardware.Devices.Backreflection

Assembly: Santec.Hardware.dll

Syntax

```
public interface IBackreflectionMeter : IWavelengthDevice, IDetectorDeviceWithDarking, IDetectorDevice,
IDevice, IDisposable
```

Methods

DarkBRDetectorAsync(CancellationToken)

Asynchronously dark the BR detector.

Declaration

```
Task DarkBRDetectorAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device dark BR detector response does not match the expected response.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

GetBRReferenceAsync(Int32, CancellationToken)

Asynchronously get the BR zero reference at a given wavelength.

Declaration

```
Task<double> GetBRReferenceAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	The wavelength to get the BR zero for.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the device does not support the specified wavelength.
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the BR reference query response is not a valid number.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

ReadBRAsync(Int32, CancellationToken)

Asynchronously read the backreflection at a given wavelength.

Declaration

```
Task<ReadingValue> ReadBRAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	The wavelength to read the backreflection at.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<ReadingValue>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION

TYPE	CONDITION
System.ArgumentException	Thrown when the device does not support the specified wavelength.
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device read BR response is not a valid number.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

ReferenceBRAsync(Int32, CancellationToken)

Asynchronously perform a BR zero reference at a given wavelength.

Declaration

```
Task<double> ReferenceBRAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	The wavelength to perform the BR zero reference at.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the device does not support the specified wavelength.

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device reference BR response is not a valid number.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

SetBRReferenceAsync(Int32, Double, CancellationToken)

Asynchronously set the BR zero reference at a given wavelength.

Declaration

```
Task SetBRReferenceAsync(int wavelength, double reference, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	The wavelength to set the BR zero for.
System.Double	reference	The BR zero value.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the device does not support the specified wavelength.

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the BR reference query response does not match the expected response.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

Interface IBRM

Defines the properties of a BRM and operations that can be performed with a connection to the device.

Inherited Members

ISantecDevice.GetIPAddressAsync(CancellationToken)
ISantecDevice.SetIPAddressAsync(IPAddress, CancellationToken)
ISantecDevice.EnableDHCPAsync(CancellationToken)
ISantecDevice.GetGatewayAsync(CancellationToken)
ISantecDevice.SetGatewayAsync(IPAddress, CancellationToken)
ISantecDevice.GetSubnetPrefixLengthAsync(CancellationToken)
ISantecDevice.SetSubnetPrefixLengthAsync(Int32, CancellationToken)
ISantecDevice.GetHostnameAsync(CancellationToken)
ISantecDevice.SetHostnameAsync(String, CancellationToken)
ISantecDevice.GetInteractionModeAsync(CancellationToken)
ISantecDevice.SetInteractionModeAsync(EInteractionMode, CancellationToken)
IDiscretePowerMeter.ReadMultiplePowersAsync(IEnumerable<Int32>, Int32, CancellationToken)
IDiscretePowerMeter.ReadMultipleILsAsync(IEnumerable<Int32>, Int32, CancellationToken)
IBackreflectionMeter.DarkBRDetectorAsync(CancellationToken)
IBackreflectionMeter.GetBRReferenceAsync(Int32, CancellationToken)
IBackreflectionMeter.SetBRReferenceAsync(Int32, Double, CancellationToken)
IBackreflectionMeter.ReferenceBRAsync(Int32, CancellationToken)
IBackreflectionMeter.ReadBRAsync(Int32, CancellationToken)
IDetectorDeviceWithDarking.DarkAllDetectorsAsync(CancellationToken)
IDetectorDeviceWithDarking.DarkDetectorAsync(Int32, CancellationToken)
IDetectorDeviceWithDarking.GetDetectorDarkStateAsync(Int32, CancellationToken)
ILaserSourceWithMonitor.ReadMonitorPowerAsync(Int32, CancellationToken)
ILaserSourceWithMonitor.GetFactoryPowerAsync(Int32, Int32, CancellationToken)
ILaserSource.NumberOfOutputs
ILaserSource.SupportsTurningOffLaser
ILaserSource.GetActiveLaserAsync(CancellationToken)
ILaserSource.TurnOnLaserAsync(Int32, CancellationToken)
ILaserSource.TurnOffLaserAsync(CancellationToken)
ILaserSource.GetOutputChannelAsync(CancellationToken)
ILaserSource.ChangeOutputChannelAsync(Int32, CancellationToken)
IWavelengthDevice.Wavelengths
ISwitchController.Switches
ISwitchController.GetSwitchChannelAsync(Int32, CancellationToken)
ISwitchController.ChangeSwitchChannelAsync(Int32, Int32, CancellationToken)
ISwitchController.ChangeMultipleSwitchesChannelsAsync(IEnumerable<DeviceSwitchAndChannelPair>, CancellationToken)
IFiberDevice.Fiber
IPowerMeterWithReferencePersistence.RequiresILReferenceSave
IPowerMeterWithReferencePersistence.SaveILReferencesAsync(CancellationToken)
IPowerMeter.ReadPowerAsync(Int32, Int32, CancellationToken)
IInsertionLossMeter.GetILReferenceAsync(Int32, Int32, CancellationToken)
IInsertionLossMeter.SetILReferenceAsync(Int32, Int32, Double, CancellationToken)
IInsertionLossMeter.ReferenceILAsync(Int32, Int32, CancellationToken)
IInsertionLossMeter.ReferenceILAsync(Int32, Int32, Boolean, CancellationToken)
IInsertionLossMeter.ReadILAsync(Int32, Int32, CancellationToken)
IDetectorDevice.NumberOfDetectors
IDetectorDevice.Detectors
IDevice<EBRMModel>.Model

IDevice.Name
IDevice.SerialNumber
IDevice.FirmwareVersion
IDevice.Address
IDevice.IsInitialized
IDevice.InitializeAsync()
IDevice.Uninitialize()
IDevice.GetTimeout()
IDevice.SetTimeout(ETimeout)
IDevice.ResetAsync()
IDevice.RefreshAsync()
IDevice.PingAsync()
IDevice.QueryAsync(String, CancellationToken)
IDevice.WriteAsync(String, CancellationToken)
IDevice.ReadAsync(CancellationToken)
System.IDisposable.Dispose()

Namespace: Santec.Hardware.Devices.Backreflection

Assembly: Santec.Hardware.dll

Syntax

```
public interface IBRM : ISantecDevice, IDiscretePowerMeter, IBackreflectionMeter,
IDetectorDeviceWithDarking, ILaserSourceWithMonitor, ILaserSource, IWavelengthDevice, ISwitchController,
IFiberDevice, IPowerMeterWithReferencePersistence, IPowerMeter, IInsertionLossMeter, IDetectorDevice,
IDevice<EBRMModel>, IDevice, IDisposable
```

Methods

GetILCompensationEnabledAsync(CancellationToken)

Asynchronously get whether IL compensation is enabled.

Declaration

```
Task<bool> GetILCompensationEnabledAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Boolean>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized BRM.
UnexpectedDeviceResponseException	Thrown when the BRM IL compensation enabled response is not a valid boolean value.
DeviceTimeoutException	Thrown when a communication failure causes no response to be received from the BRM.

SetILCompensationEnabledAsync(Boolean, CancellationToken)

Asynchronously set whether IL compensation is enabled.

Declaration

```
Task SetILCompensationEnabledAsync(bool enabled, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Boolean	enabled	Whether to enable IL compensation or not.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized BRM.
UnexpectedDeviceResponseException	Thrown when the BRM IL compensation enabled response does not match the expected response.

TYPE	CONDITION
DeviceTimeoutException	Thrown when a communication failure causes no response to be received from the BRM.

Class SimulatedBRM

Provides a simulated BRM with configurable reading values.

> **NOTE**
>
> This class is intended for testing/development purposes. It operates much quicker than an actual BRM. It can only be created using a SimulatedDeviceFactory. It should only be used when returned as part of a list of devices by a DetectHardwareAsync(EConnectionTypesToDetect) or DetectHardwareAsync<T>(EConnectionTypesToDetect) call to a SimulatedHardwareDetectionService. Each reading value contains a default value range but can be configured via the appropriate configuration method.

Inheritance

System.Object
SimulatedSantecDevice
SimulatedBRM

Implements

ISimulatedDevice
IBRM
ISantecDevice
IDiscretePowerMeter
ISwitchController
IFiberDevice
IPowerMeterWithReferencePersistence
IDevice<EBRMModel>
IBackreflectionMeter
IDetectorDeviceWithDarking
IPowerMeter
IInsertionLossMeter
IDetectorDevice
ILaserSourceWithMonitor
ILaserSource
IWavelengthDevice
IDevice
System.IDisposable

Inherited Members

SimulatedSantecDevice.creatingDevice
SimulatedSantecDevice.Dispose()
SimulatedSantecDevice.SerialNumber
SimulatedSantecDevice.FirmwareVersion
SimulatedSantecDevice.Address
SimulatedSantecDevice.IsInitialized
SimulatedSantecDevice.QueryResponses
SimulatedSantecDevice.ConfigureQueryResponses(IEnumerable<QueryResponsePair>)
SimulatedSantecDevice.GetTimeout()
SimulatedSantecDevice.SetTimeout(ETimeout)
SimulatedSantecDevice.GetInteractionModeAsync(CancellationToken)
SimulatedSantecDevice.GetIPAddressAsync(CancellationToken)
SimulatedSantecDevice.SetIPAddressAsync(IPAddress, CancellationToken)
SimulatedSantecDevice.EnableDHCPAsync(CancellationToken)
SimulatedSantecDevice.GetGatewayAsync(CancellationToken)
SimulatedSantecDevice.SetGatewayAsync(IPAddress, CancellationToken)

SimulatedSantecDevice.GetSubnetPrefixLengthAsync(CancellationToken)
SimulatedSantecDevice.SetSubnetPrefixLengthAsync(Int32, CancellationToken)
SimulatedSantecDevice.GetHostnameAsync(CancellationToken)
SimulatedSantecDevice.SetHostnameAsync(String, CancellationToken)
SimulatedSantecDevice.SetInteractionModeAsync(EInteractionMode, CancellationToken)
SimulatedSantecDevice.InitializeAsync()
SimulatedSantecDevice.PingAsync()
SimulatedSantecDevice.Uninitialize()
SimulatedSantecDevice.ResetAsync()
SimulatedSantecDevice.RefreshAsync()
SimulatedSantecDevice.QueryAsync(String, CancellationToken)
SimulatedSantecDevice.WriteAsync(String, CancellationToken)
SimulatedSantecDevice.ReadAsync(CancellationToken)
System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.Backreflection

Assembly: Santec.Hardware.dll

Syntax

```
public class SimulatedBRM : SimulatedSantecDevice, ISimulatedDevice, IBRM, ISantecDevice,
IDiscretePowerMeter, ISwitchController, IFiberDevice, IPowerMeterWithReferencePersistence,
IDevice<EBRMModel>, IBackreflectionMeter, IDetectorDeviceWithDarking, IPowerMeter, IInsertionLossMeter,
IDetectorDevice, ILaserSourceWithMonitor, ILaserSource, IWavelengthDevice, IDevice, IDisposable
```

Properties

BRMax

Declaration

```
public double BRMax { get; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The maximum BR reading value. The default value is `-65`.

BRMin

Declaration

```
public double BRMin { get; }
```

Property Value

TYPE	DESCRIPTION

TYPE	DESCRIPTION
System.Double	The minimum BR reading value. The default value is `-75`.

BRReferenceMax

Declaration

```
public double BRReferenceMax { get; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The maximum BR reference value. The default value is `-35`.

BRReferenceMin

Declaration

```
public double BRReferenceMin { get; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The minimum BR reference value. The default value is `-36`.

Detectors

Declaration

```
public IReadOnlyList<DeviceDetector> Detectors { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<DeviceDetector>	

Fiber

Declaration

```
public FiberInformation Fiber { get; }
```

Property Value

TYPE	DESCRIPTION
FiberInformation	

ILMax

Declaration

```
public double ILMax { get; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The maximum IL reading value. The default value is `0.15`.

ILMin

Declaration

```
public double ILMin { get; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The minimum IL reading value. The default value is `0`.

ILReferenceMax

Declaration

```
public double ILReferenceMax { get; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The maximum IL reference value. The default value is `5`.

ILReferenceMin

Declaration

```
public double ILReferenceMin { get; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The minimum IL reference value. The default value is `2`.

Model

Declaration

```
public EBRMModel Model { get; }
```

Property Value

TYPE	DESCRIPTION
EBRMModel	

MonitorPowerMax

Declaration

```
public double MonitorPowerMax { get; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The maximum monitor power value. The default value is `-30`.

MonitorPowerMin

Declaration

```
public double MonitorPowerMin { get; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The minimum monitor power value. The default value is `-30`.

Name

Declaration

```
public override string Name { get; }
```

Property Value

TYPE	DESCRIPTION
System.String	

Overrides

SimulatedSantecDevice.Name

NumberOfDetectors

Declaration

```
public int NumberOfDetectors { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

NumberOfOutputs

Declaration

```
public int NumberOfOutputs { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

PowerMax

Declaration

```
public double PowerMax { get; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The maximum power reading value. The default value is `-3`.

PowerMin

Declaration

```
public double PowerMin { get; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The minimum power reading value. The default value is `-3.15`.

RequiresILReferenceSave

Declaration

```
public bool RequiresILReferenceSave { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	

SupportsTurningOffLaser

Declaration

```
public bool SupportsTurningOffLaser { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	

Remarks

> **NOTE**
>
> This property will always return `true`.

Switches

Declaration

```
public List<DeviceSwitch> Switches { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.List<DeviceSwitch>	

Wavelengths

Declaration

```
public List<int> Wavelengths { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.List<System.Int32>	

Methods

ChangeMultipleSwitchesChannelsAsync(IEnumerable<DeviceSwitchAndChannelPair>, CancellationToken)

Declaration

```
public async Task ChangeMultipleSwitchesChannelsAsync(IEnumerable<DeviceSwitchAndChannelPair>
deviceSwitchesAndChannels, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Collections.Generic.IEnumerable<DeviceSwitchAndChannelPair>	deviceSwitchesAndChannels	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

ChangeOutputChannelAsync(Int32, CancellationToken)

Declaration

```
public async Task ChangeOutputChannelAsync(int output, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	output	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

ChangeSwitchChannelAsync(Int32, Int32, CancellationToken)

Declaration

```
public async Task ChangeSwitchChannelAsync(int switchIndex, int channel, CancellationToken cancellationToken
= default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	switchIndex	
System.Int32	channel	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

ConfigureBR(Double, Double)

Declaration

```
public void ConfigureBR(double brMin, double brMax)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Double	brMin	
System.Double	brMax	

ConfigureBRReference(Double, Double)

Declaration

```
public void ConfigureBRReference(double brReferenceMin, double brReferenceMax)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Double	brReferenceMin	
System.Double	brReferenceMax	

ConfigureIL(Double, Double)

Declaration

```
public void ConfigureIL(double ilMin, double ilMax)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Double	ilMin	
System.Double	ilMax	

ConfigureILReference(Double, Double)

Declaration

```
public void ConfigureILReference(double ilReferenceMin, double ilReferenceMax)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Double	ilReferenceMin	
System.Double	ilReferenceMax	

ConfigureMonitorPower(Double, Double)

Declaration

```
public void ConfigureMonitorPower(double monitorPowerMin, double monitorPowerMax)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Double	monitorPowerMin	
System.Double	monitorPowerMax	

ConfigurePower(Double, Double)

Declaration

```
public void ConfigurePower(double powerMin, double powerMax)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Double	powerMin	
System.Double	powerMax	

DarkAllDetectorsAsync(CancellationToken)

Declaration

```
public async Task DarkAllDetectorsAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

DarkBRDetectorAsync(CancellationToken)

Declaration

```
public Task DarkBRDetectorAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

DarkDetectorAsync(Int32, CancellationToken)

Declaration

```
public Task DarkDetectorAsync(int detector, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

GetActiveLaserAsync(CancellationToken)

Declaration

```
public async Task<int> GetActiveLaserAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Int32>	

GetBRReferenceAsync(Int32, CancellationToken)

Declaration

```
public async Task<double> GetBRReferenceAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

GetDetectorDarkStateAsync(Int32, CancellationToken)

Declaration

```
public Task<EDarkState> GetDetectorDarkStateAsync(int detector, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<EDarkState>	

GetFactoryPowerAsync(Int32, Int32, CancellationToken)

Declaration

```
public async Task<FactoryPowerReading> GetFactoryPowerAsync(int output, int wavelength, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	output	
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<FactoryPowerReading>	

GetILCompensationEnabledAsync(CancellationToken)

Declaration

```
public async Task<bool> GetILCompensationEnabledAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Boolean>	

GetILReferenceAsync(Int32, Int32, CancellationToken)

Declaration

```
public async Task<double> GetILReferenceAsync(int detector, int wavelength, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

GetOutputChannelAsync(CancellationToken)

Declaration

```
public async Task<int> GetOutputChannelAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Int32>	

GetSwitchChannelAsync(Int32, CancellationToken)

Declaration

```
public async Task<int> GetSwitchChannelAsync(int switchIndex, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	switchIndex	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Int32>	

ReadBRAsync(Int32, CancellationToken)

Declaration

```
public async Task<ReadingValue> ReadBRAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<ReadingValue>	

ReadILAsync(Int32, Int32, CancellationToken)

Declaration

```
public async Task<double> ReadILAsync(int detector, int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

ReadMonitorPowerAsync(Int32, CancellationToken)

Declaration

```
public async Task<double> ReadMonitorPowerAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

ReadMultipleILsAsync(IEnumerable<Int32>, Int32, CancellationToken)

Declaration

```
public async Task<List<double>> ReadMultipleILsAsync(IEnumerable<int> detectors, int wavelength,
CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Collections.Generic.IEnumerable<System.Int32>	detectors	
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Collections.Generic.List<System.Double>>	

ReadMultiplePowersAsync(IEnumerable<Int32>, Int32, CancellationToken)

Declaration

```
public async Task<List<double>> ReadMultiplePowersAsync(IEnumerable<int> detectors, int wavelength,
CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Collections.Generic.IEnumerable<System.Int32>	detectors	
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Collections.Generic.List<System.Double>>	

ReadPowerAsync(Int32, Int32, CancellationToken)

Declaration

```
public async Task<double> ReadPowerAsync(int detector, int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

ReferenceBRAsync(Int32, CancellationToken)

Declaration

```
public async Task<double> ReferenceBRAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

ReferenceILAsync(Int32, Int32, Boolean, CancellationToken)

Declaration

```
public async Task<double> ReferenceILAsync(int detector, int wavelength, bool applyReferenceToAllDetectors,
CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Int32	wavelength	
System.Boolean	applyReferenceToAllDetectors	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

ReferenceILAsync(Int32, Int32, CancellationToken)

Declaration

```
public async Task<double> ReferenceILAsync(int detector, int wavelength, CancellationToken cancellationToken
= default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

SaveILReferencesAsync(CancellationToken)

Declaration

```
public async Task SaveILReferencesAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetBRReferenceAsync(Int32, Double, CancellationToken)

Declaration

```
public async Task SetBRReferenceAsync(int wavelength, double reference, CancellationToken cancellationToken
= default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	
System.Double	reference	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetILCompensationEnabledAsync(Boolean, CancellationToken)

Declaration

```
public async Task SetILCompensationEnabledAsync(bool enabled, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Boolean	enabled	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetILReferenceAsync(Int32, Int32, Double, CancellationToken)

Declaration

```
public async Task SetILReferenceAsync(int detector, int wavelength, double reference, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Int32	wavelength	

TYPE	NAME	DESCRIPTION
System.Double	reference	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

TurnOffLaserAsync(CancellationToken)

Declaration

```
public async Task TurnOffLaserAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

TurnOnLaserAsync(Int32, CancellationToken)

Declaration

```
public async Task TurnOnLaserAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Explicit Interface Implementations

IDetectorDevice.Detectors

Declaration

```
IReadOnlyList<IModule> IDetectorDevice.Detectors { get; }
```

Returns

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<IModule>	

Implements

ISimulatedDevice
IBRM
ISantecDevice
IDiscretePowerMeter
ISwitchController
IFiberDevice
IPowerMeterWithReferencePersistence
IDevice<TModelType>
IBackreflectionMeter
IDetectorDeviceWithDarking
IPowerMeter
IInsertionLossMeter
IDetectorDevice
ILaserSourceWithMonitor
ILaserSource
IWavelengthDevice
IDevice
System.IDisposable

Namespace Santec.Hardware.Devices.General

Classes

CompositeSantecDevice

Provides a connection to a Santec Canada family composite device.

> **⚠ WARNING**
>
> A composite device does not support all operations of a Santec Canada family device. Unsupported operations will throw a System.NotSupportedException.

DeviceDetector

Represents a detector of a IDetectorDevice.

DeviceDetectorWithSerialNumber

Represents a detector of a IDetectorDevice and that has a serial number.

DeviceFactory

Provides a factory for creating device connections.

> **NOTE**
>
> This class should generally not be used directly. Devices should generally be created when returned:
>
> 1. As part of a list of devices by a DetectHardwareAsync(EConnectionTypesToDetect) or DetectHardwareAsync<T>(EConnectionTypesToDetect) call.
> 2. As a device by a TryDetectIPDeviceAsync(IPAddress) or TryDetectIPDeviceAsync<T>(IPAddress) call.

FiberInformation

Represents information about the fiber of a device

PhysicalDevice

Provides a connection to the device and general device properties and operations for physical hardware devices.

QueryResponsePair

Represents a single mapping between a query string and the response string that a simulated device should return for that query.

RDSPDetector

Represents an RD-SP detector of a IDetectorDevice.

ReadingValue

Represents a device reading value.

> **NOTE**
>
> This class should not be used directly. It should only be used when returned by a device reading.

SantecDevice

Provides a connection to a Santec Canada family device and provides the general IP functionality for the device.

SimulatedDeviceFactory

Provides a factory for creating simulated device connections.

> **NOTE**
>
> This class should be used for the creation of all simulated devices. However, the created simulated devices should not be used directly. They should only be used when returned as part of a list of devices by a DetectHardwareAsync(EConnectionTypesToDetect) or DetectHardwareAsync<T>(EConnectionTypesToDetect) call to a SimulatedHardwareDetectionService.

SimulatedSantecDevice

Provides the general functionality for a simulated Santec Canada family device.

Interfaces

ICompositeDevice

Defines general initialization and connection properties and methods for composite devices composed of other hardware devices.

IDetectorDevice

Defines the general operations for a device with one or more detectors.

IDetectorDeviceWithDarking

Defines the general operations for a device with one or more detectors that can be darked.

IDevice

Defines general initialization and connection properties and methods for hardware devices.

IDevice<TModelType>

Defines general initialization and connection properties and methods for hardware devices. This includes the model type of the device.

IDeviceWithInputs

Defines the properties of a device with one or more selectable input channels and the operations that can be performed with a connection to the device.

IDeviceWithWavelengthRange

Defines the properties of a device that supports a range of wavelengths. Defines the operations that can be performed with a connection to the device.

IFiberDevice

Defines the properties of a fiber device.

ISantecDevice

Defines the general IP functionality for the Santec Canada family of devices.

ISimulatedDevice

Defines the general configurable properties of a simulated device

> **NOTE**
>
> This interface extends IDevice with methods intended for configuring simulated device behaviour for testing and development. Implementations should be provided by simulated device classes created via SimulatedDeviceFactory.

IWavelengthDevice

Defines the properties of a discrete wavelength device.

Enums

ECoreSize

Specifies the core size of a device

EDarkState

Specifies the state of the user dark for a module/device

EFiberType

Specifies the fiber type of a device.

EInteractionMode

Specifies the interaction mode of a device.

ERangeStatus

Specifies the range status of a device reading.

ETimeout

Defines the timeout values for device operations.

ETriggerEdge

Specifies the trigger edge of a device.

Class CompositeSantecDevice

Provides a connection to a Santec Canada family composite device.

> **⚠ WARNING**
>
> A composite device does not support all operations of a Santec Canada family device. Unsupported operations will throw a System.NotSupportedException.

Inheritance

System.Object
CompositeSantecDevice
RDP
RDSP
SimulatedRDP
SimulatedRDSP

Implements

ISantecDevice
ICompositeDevice
IDevice
System.IDisposable

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.General

Assembly: Santec.Hardware.dll

Syntax

```
public abstract class CompositeSantecDevice : ISantecDevice, ICompositeDevice, IDevice, IDisposable
```

Properties

Address

Declaration

```
public virtual string Address { get; }
```

Property Value

TYPE	DESCRIPTION
System.String	

FirmwareVersion

Declaration

```
public Version FirmwareVersion { get; }
```

Property Value

TYPE	DESCRIPTION
Version	

IsInitialized

Declaration

```
public bool IsInitialized { get; protected set; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	

IsInitializing

Get whether the device and connection are initializing.

Declaration

```
protected bool IsInitializing { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	`true` if the device is initializing; otherwise, `false`.

Name

Declaration

```
public abstract string Name { get; }
```

Property Value

TYPE	DESCRIPTION
System.String	

SerialNumber

Declaration

```
public string SerialNumber { get; protected set; }
```

Property Value

TYPE	DESCRIPTION
System.String	

Subdevices

Declaration

```
public IReadOnlyList<IDevice> Subdevices { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<IDevice>	

Methods

Dispose()

Close and clean up the device, device connection, and any associated resources, including the subdevices.

Declaration

```
public void Dispose()
```

Dispose(Boolean)

Declaration

```
public void Dispose(bool disposeSubdevices)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Boolean	disposeSubdevices	

Dispose(Boolean, Boolean)

Close and clean up the device, device connection, and any associated resources.

Declaration

```
protected virtual void Dispose(bool disposing, bool disposeSubdevices)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Boolean	disposing	`true` for explicit dispose calls; otherwise, `false`.
System.Boolean	disposeSubdevices	`true` if subdevices should also be disposed; otherwise, `false`.

EnableDHCPAsync(CancellationToken)

> **⚠ CAUTION**
>
> This operation is not supported for all composite devices. This operation may throw a System.NotSupportedException.

Declaration

```
public virtual Task EnableDHCPAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Exceptions

TYPE	CONDITION
System.NotSupportedException	Thrown when the operation is not supported by the device.

Finalize()

Destroys the CompositeSantecDevice object and cleans up any resources.

Declaration

```
protected void Finalize()
```

GetGatewayAsync(CancellationToken)

> **⚠ CAUTION**
>
> This operation is not supported for all composite devices. This operation may throw a System.NotSupportedException.

Declaration

```
public virtual Task<IPAddress> GetGatewayAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Net.IPAddress>	

Exceptions

TYPE	CONDITION
System.NotSupportedException	Thrown when the operation is not supported by the device.

GetHostnameAsync(CancellationToken)

> **⚠ CAUTION**
>
> This operation is not supported for all composite devices. This operation may throw a System.NotSupportedException.

Declaration

```
public virtual Task<string> GetHostnameAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.String>	

Exceptions

TYPE	CONDITION
System.NotSupportedException	Thrown when the operation is not supported by the device.

GetInteractionModeAsync(CancellationToken)

> **⚠ CAUTION**
>
> This operation is not supported for all composite devices. This operation may throw a System.NotSupportedException.

Declaration

```
public virtual Task<EInteractionMode> GetInteractionModeAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<EInteractionMode>	

Exceptions

TYPE	CONDITION
System.NotSupportedException	Thrown when the operation is not supported by the device.

GetIPAddressAsync(CancellationToken)

⚠ CAUTION

This operation is not supported for all composite devices. This operation may throw a System.NotSupportedException.

Declaration

```
public virtual Task<IPAddress> GetIPAddressAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Net.IPAddress>	

Exceptions

TYPE	CONDITION
System.NotSupportedException	Thrown when the operation is not supported by the device.

GetSubnetPrefixLengthAsync(CancellationToken)

⚠ CAUTION

This operation is not supported for all composite devices. This operation may throw a System.NotSupportedException.

Declaration

```
public virtual Task<int> GetSubnetPrefixLengthAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Int32>	

Exceptions

TYPE	CONDITION
System.NotSupportedException	Thrown when the operation is not supported by the device.

GetTimeout()

Declaration

```
public ETimeout GetTimeout()
```

Returns

TYPE	DESCRIPTION
ETimeout	

InitializeAsync()

Declaration

```
public virtual async Task InitializeAsync()
```

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

PingAsync()

Declaration

```
public async Task<bool> PingAsync()
```

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Boolean>	

QueryAsync(String, CancellationToken)

> **⚠ CAUTION**
>
> This operation is not supported for all composite devices. This operation may throw a System.NotSupportedException.

Declaration

```
public virtual Task<string> QueryAsync(string command, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	command	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.String>	

Exceptions

TYPE	CONDITION
System.NotSupportedException	Thrown when the operation is not supported by the device.

ReadAsync(CancellationToken)

> **⚠ CAUTION**
>
> This operation is not supported for all composite devices. This operation may throw a System.NotSupportedException.

Declaration

```
public virtual Task<string> ReadAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.String>	

Exceptions

TYPE	CONDITION
System.NotSupportedException	Thrown when the operation is not supported by the device.

RefreshAsync()

Asynchronously refresh the device settings and the settings of all subdevices.

Declaration

```
public virtual async Task RefreshAsync()
```

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when a subdevice is uninitialized when the method is called.
UnexpectedDeviceResponseException	Thrown when a subdevice gives an unexpected response during its refresh operation.
System.TimeoutException	Thrown when a communication failure causes no response to be received from a subdevice during its refresh operation.

ResetAsync()

Declaration

```
public async Task ResetAsync()
```

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetGatewayAsync(IPAddress, CancellationToken)

⚠ CAUTION

This operation is not supported for all composite devices. This operation may throw a System.NotSupportedException.

Declaration

```
public virtual Task SetGatewayAsync(IPAddress address, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Net.IPAddress	address	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Exceptions

TYPE	CONDITION
System.NotSupportedException	Thrown when the operation is not supported by the device.

SetHostnameAsync(String, CancellationToken)

⚠ CAUTION

This operation is not supported for all composite devices. This operation may throw a System.NotSupportedException.

Declaration

```
public virtual Task SetHostnameAsync(string hostname, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	hostname	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Exceptions

TYPE	CONDITION
System.NotSupportedException	Thrown when the operation is not supported by the device.

SetInteractionModeAsync(EInteractionMode, CancellationToken)

⚠ CAUTION

This operation is not supported for all composite devices. This operation may throw a System.NotSupportedException.

Declaration

```
public virtual Task SetInteractionModeAsync(EInteractionMode interactionMode, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
EInteractionMode	interactionMode	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Exceptions

TYPE	CONDITION
System.NotSupportedException	Thrown when the operation is not supported by the device.

SetIPAddressAsync(IPAddress, CancellationToken)

⚠ CAUTION

This operation is not supported for all composite devices. This operation may throw a System.NotSupportedException.

Declaration

```
public virtual Task SetIPAddressAsync(IPAddress address, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Net.IPAddress	address	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Exceptions

TYPE	CONDITION
System.NotSupportedException	Thrown when the operation is not supported by the device.

SetSubnetPrefixLengthAsync(Int32, CancellationToken)

> **⚠ CAUTION**
>
> This operation is not supported for all composite devices. This operation may throw a System.NotSupportedException.

Declaration

```
public virtual Task SetSubnetPrefixLengthAsync(int prefixLength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	prefixLength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Exceptions

TYPE	CONDITION
System.NotSupportedException	Thrown when the operation is not supported by the device.

SetTimeout(ETimeout)

Declaration

```
public void SetTimeout(ETimeout timeout)
```

Parameters

TYPE	NAME	DESCRIPTION
ETimeout	timeout	

Uninitialize()

Declaration

```
public void Uninitialize()
```

Uninitialize(Boolean)

Declaration

```
public virtual void Uninitialize(bool uninitializeSubdevices)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Boolean	uninitializeSubdevices	

WriteAsync(String, CancellationToken)

> **⚠ CAUTION**
>
> This operation is not supported for all composite devices. This operation may throw a System.NotSupportedException.

Declaration

```
public virtual Task WriteAsync(string command, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	command	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Exceptions

TYPE	CONDITION
System.NotSupportedException	Thrown when the operation is not supported by the device.

Implements

ISantecDevice
ICompositeDevice
IDevice
System.IDisposable

Class DeviceDetector

Represents a detector of a IDetectorDevice.

Inheritance

System.Object
DeviceDetector
DeviceDetectorWithSerialNumber

Implements

IModule

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.General

Assembly: Santec.Hardware.dll

Syntax

```
public class DeviceDetector : IModule
```

Properties

FirmwareVersion

Get the firmware version of the detector.

Declaration

```
public Version FirmwareVersion { get; }
```

Property Value

TYPE	DESCRIPTION
Version	The firmware version of the detector.

Index

Get the detector index.

Declaration

```
public int Index { get; }
```

Property Value

TYPE	DESCRIPTION

TYPE	DESCRIPTION
System.Int32	The detector index.

IsRDSP

Get whether the detector is an RD-SP detector.

Declaration

```
public virtual bool IsRDSP { get; protected set; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	`true` if the detector is an RD-SP; otherwise, `false`.

Implements

IModule

Class DeviceDetectorWithSerialNumber

Represents a detector of a IDetectorDevice and that has a serial number.

Inheritance

System.Object
DeviceDetector
DeviceDetectorWithSerialNumber
RDSPDetector

Implements

IModule

Inherited Members

DeviceDetector.Index
DeviceDetector.IsRDSP
DeviceDetector.FirmwareVersion
System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.General

Assembly: Santec.Hardware.dll

Syntax

```
public class DeviceDetectorWithSerialNumber : DeviceDetector, IModule
```

Properties

SerialNumber

Get the detector serial number.

Declaration

```
public string SerialNumber { get; }
```

Property Value

TYPE	DESCRIPTION
System.String	The detector serial number.

Implements

IModule

Class DeviceFactory

Provides a factory for creating device connections.

> **NOTE**
>
> This class should generally not be used directly. Devices should generally be created when returned:
>
> 1. As part of a list of devices by a DetectHardwareAsync(EConnectionTypesToDetect) or DetectHardwareAsync<T>(EConnectionTypesToDetect) call.
> 2. As a device by a TryDetectIPDeviceAsync(IPAddress) or TryDetectIPDeviceAsync<T>(IPAddress) call.

Inheritance

System.Object
DeviceFactory

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.General

Assembly: Santec.Hardware.dll

Syntax

```
public class DeviceFactory
```

Constructors

DeviceFactory()

Initializes a new device factory.

Declaration

```
public DeviceFactory()
```

DeviceFactory(ICultureService)

Initializes a new device factory.

Declaration

```
public DeviceFactory(ICultureService cultureService)
```

Parameters

TYPE	NAME	DESCRIPTION
Santec.Core.Globalization.ICultureService	cultureService	The culture service for localization.

Methods

CreatePTM(IPAddress, String)

Creates a new IP connection to a PTM device.

Declaration

```
public PTM CreatePTM(IPAddress ipAddress, string serialNumber)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Net.IPAddress	ipAddress	The IP address of the RLM.
System.String	serialNumber	The serial number of the RLM.

Returns

TYPE	DESCRIPTION
PTM	The newly created RLM.

CreateRLM(IPAddress, String)

Creates a new IP connection to an RLM device.

Declaration

```
public RLM CreateRLM(IPAddress ipAddress, string serialNumber)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Net.IPAddress	ipAddress	The IP address of the RLM.
System.String	serialNumber	The serial number of the RLM.

Returns

TYPE	DESCRIPTION
RLM	The newly created RLM.

Enum ECoreSize

Specifies the core size of a device

Namespace: Santec.Hardware.Devices.General

Assembly: Santec.Hardware.dll

Syntax

```
public enum ECoreSize
```

Fields

NAME	DESCRIPTION
Core_100um	100µm core.
Core_50um	50µm core.
Core_62_5um	62.5µm core.
Core_9um	9µm core.
Core_Dual_50um_62_5um	Dual 50µm and 62.5µm core.
Unknown	Unknown core size.

Enum EDarkState

Specifies the state of the user dark for a module/device

Namespace: Santec.Hardware.Devices.General

Assembly: Santec.Hardware.dll

Syntax

```
public enum EDarkState
```

Fields

NAME	DESCRIPTION
Completed	The user dark has been completed and stored.
Incomplete	The user dark is incomplete.

Enum EFiberType

Specifies the fiber type of a device.

Namespace: Santec.Hardware.Devices.General

Assembly: Santec.Hardware.dll

Syntax

```
public enum EFiberType
```

Fields

NAME	DESCRIPTION
BowTie	Bow-tie Fiber
Flex	Flex Fiber
Multimode	Multimode fiber.
Panda	PM Fiber
SingleMode	Single mode fiber.
Unknown	Unknown fiber type.

Enum EInteractionMode

Specifies the interaction mode of a device.

Namespace: Santec.Hardware.Devices.General

Assembly: Santec.Hardware.dll

Syntax

```
public enum EInteractionMode
```

Fields

NAME	DESCRIPTION
Local	Local interaction mode.
Remote	Remote interaction mode.

Enum ERangeStatus

Specifies the range status of a device reading.

Namespace: Santec.Hardware.Devices.General

Assembly: Santec.Hardware.dll

Syntax

```
public enum ERangeStatus
```

Fields

NAME	DESCRIPTION
InRange	The reading is within the reading range of the device.
None	The reading has no range status.
OutOfLowerRange	The reading is lower than the lower range limit of the device.
OutOfUpperRange	The reading is greater than the upper range limit of the device.

Enum ETimeout

Defines the timeout values for device operations.

Namespace: Santec.Hardware.Devices.General

Assembly: Santec.Hardware.dll

Syntax

```
public enum ETimeout
```

Fields

NAME	DESCRIPTION
Initialization_2_5s	The initialization timeout value.
Long_20s	The long timeout value.
Standard_10s	The standard timeout value.
Unknown	Unknown timeout value.
Very_Long_30s	The very long timeout value

Enum ETriggerEdge

Specifies the trigger edge of a device.

Namespace: Santec.Hardware.Devices.General

Assembly: Santec.Hardware.dll

Syntax

```
public enum ETriggerEdge
```

Fields

NAME	DESCRIPTION
Falling	Trigger on the falling edge.
Rising	Trigger on the rising edge.

Class FiberInformation

Represents information about the fiber of a device

Inheritance

System.Object
FiberInformation

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.General

Assembly: Santec.Hardware.dll

Syntax

```
public class FiberInformation
```

Constructors

FiberInformation(EFiberType, ECoreSize)

Initializes a new record of fiber information.

Declaration

```
public FiberInformation(EFiberType fiberType, ECoreSize coreSize)
```

Parameters

TYPE	NAME	DESCRIPTION
EFiberType	fiberType	The fiber type of the fiber.
ECoreSize	coreSize	The core size of the fiber.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the fiber type and core size are not compatible.
System.ArgumentException	Thrown when the core size is known but the fiber type is unknown.

Properties

CoreSize

Get the fiber core size of the device.

Declaration

```
public ECoreSize CoreSize { get; }
```

Property Value

TYPE	DESCRIPTION
ECoreSize	The fiber core size of the device.

FiberType

Get the fiber type of the device.

Declaration

```
public EFiberType FiberType { get; }
```

Property Value

TYPE	DESCRIPTION
EFiberType	The fiber type of the device.

Interface ICompositeDevice

Defines general initialization and connection properties and methods for composite devices composed of other hardware devices.

Inherited Members

IDevice.Name
IDevice.SerialNumber
IDevice.FirmwareVersion
IDevice.Address
IDevice.IsInitialized
IDevice.InitializeAsync()
IDevice.Uninitialize()
IDevice.GetTimeout()
IDevice.SetTimeout(ETimeout)
IDevice.ResetAsync()
IDevice.RefreshAsync()
IDevice.PingAsync()
IDevice.QueryAsync(String, CancellationToken)
IDevice.WriteAsync(String, CancellationToken)
IDevice.ReadAsync(CancellationToken)
System.IDisposable.Dispose()

Namespace: Santec.Hardware.Devices.General

Assembly: Santec.Hardware.dll

Syntax

```
public interface ICompositeDevice : IDevice, IDisposable
```

Properties

Subdevices

Get the subdevices that make up the composite device.

Declaration

```
IReadOnlyList<IDevice> Subdevices { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<IDevice>	A list of the subdevices that make up the composite device.

Methods

Dispose(Boolean)

Close and clean up the device, device connection, and any associated resources.

Declaration

```
void Dispose(bool disposeSubdevices)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Boolean	disposeSubdevices	Whether to also dispose the subdevices. `true` if subdevices should be disposed; otherwise `false`.

Uninitialize(Boolean)

Synchronously uninitialize device settings and close the connection.

Declaration

```
void Uninitialize(bool uninitializeSubdevices)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Boolean	uninitializeSubdevices	Whether to also uninitialize the subdevices. `true` if subdevices should be uninitialized; otherwise `false`.

Interface IDetectorDevice

Defines the general operations for a device with one or more detectors.

Inherited Members

IDevice.Name
IDevice.SerialNumber
IDevice.FirmwareVersion
IDevice.Address
IDevice.IsInitialized
IDevice.InitializeAsync()
IDevice.Uninitialize()
IDevice.GetTimeout()
IDevice.SetTimeout(ETimeout)
IDevice.ResetAsync()
IDevice.RefreshAsync()
IDevice.PingAsync()
IDevice.QueryAsync(String, CancellationToken)
IDevice.WriteAsync(String, CancellationToken)
IDevice.ReadAsync(CancellationToken)
System.IDisposable.Dispose()

Namespace: Santec.Hardware.Devices.General

Assembly: Santec.Hardware.dll

Syntax

```
public interface IDetectorDevice : IDevice, IDisposable
```

Properties

Detectors

Get the detectors connected to the device.

> **NOTE**
>
> The device must be initialized before this property contains the proper value.

Declaration

```
IReadOnlyList<IModule> Detectors { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<IModule>	The list of the detectors connected to the device.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the property is accessed before the device has been initialized.

NumberOfDetectors

Get the number of detectors connected to the device.

> **NOTE**
>
> The device must be initialized before this property contains the proper value.

Declaration

```
int NumberOfDetectors { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	The number of detectors connected to the device.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the property is accessed before the device has been initialized.

Interface IDetectorDeviceWithDarking

Defines the general operations for a device with one or more detectors that can be darked.

Inherited Members

IDetectorDevice.NumberOfDetectors
IDetectorDevice.Detectors
IDevice.Name
IDevice.SerialNumber
IDevice.FirmwareVersion
IDevice.Address
IDevice.IsInitialized
IDevice.InitializeAsync()
IDevice.Uninitialize()
IDevice.GetTimeout()
IDevice.SetTimeout(ETimeout)
IDevice.ResetAsync()
IDevice.RefreshAsync()
IDevice.PingAsync()
IDevice.QueryAsync(String, CancellationToken)
IDevice.WriteAsync(String, CancellationToken)
IDevice.ReadAsync(CancellationToken)
System.IDisposable.Dispose()

Namespace: Santec.Hardware.Devices.General

Assembly: Santec.Hardware.dll

Syntax

```
public interface IDetectorDeviceWithDarking : IDetectorDevice, IDevice, IDisposable
```

Methods

DarkAllDetectorsAsync(CancellationToken)

Asynchronously dark all of the detectors that support darking.

Declaration

```
Task DarkAllDetectorsAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device dark all detectors response does not match the expected response.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.
Santec.Hardware.Exceptions.DarkFailedException	Thrown when the device fails to dark one or more detectors.

DarkDetectorAsync(Int32, CancellationToken)

Asynchronously dark the specified detector.

Declaration

```
Task DarkDetectorAsync(int detector, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	The detector to dark.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the specified detector is out of range for the device.
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device dark detector response does not match the expected response.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.
Santec.Hardware.Exceptions.DarkFailedException	Thrown when the device fails to dark one or more detectors.

GetDetectorDarkStateAsync(Int32, CancellationToken)

Asynchronously get the state of the specified detector's user dark.

Declaration

```
Task<EDarkState> GetDetectorDarkStateAsync(int detector, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	The detector to get the user dark state for.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<EDarkState>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the specified detector is out of range for the device.
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device detector dark state response is not a valid state.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

Interface IDevice

Defines general initialization and connection properties and methods for hardware devices.

Inherited Members

System.IDisposable.Dispose()

Namespace: Santec.Hardware.Devices.General

Assembly: Santec.Hardware.dll

Syntax

```
public interface IDevice : IDisposable
```

Properties

Address

Get the connection address of the device.

Declaration

```
string Address { get; }
```

Property Value

TYPE	DESCRIPTION
System.String	The connection address of the device.

FirmwareVersion

Get the firmware version of the device.

Declaration

```
Version FirmwareVersion { get; }
```

Property Value

TYPE	DESCRIPTION
Version	The firmware version of the device.

IsInitialized

Get whether the device and connection are initialized.

Declaration

```
bool IsInitialized { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	`true` if the device is initialized; otherwise, `false`.

Name

Get the name of the device.

Declaration

```
string Name { get; }
```

Property Value

TYPE	DESCRIPTION
System.String	The name of the device.

SerialNumber

Get the serial number of the device.

Declaration

```
string SerialNumber { get; }
```

Property Value

TYPE	DESCRIPTION
System.String	The serial number of the device.

Methods

GetTimeout()

Synchronously get the device timeout.

Declaration

```
ETimeout GetTimeout()
```

Returns

TYPE	DESCRIPTION
ETimeout	The device timeout.

InitializeAsync()

Asynchronously open the device connection and initialize the device settings.

Declaration

```
Task InitializeAsync()
```

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	The task representing the operation.

Remarks

> **NOTE**
>
> This operation will not block.

PingAsync()

Asynchronously ping the device to check if the connection is still open.

Declaration

```
Task<bool> PingAsync()
```

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Boolean>	The task representing the operation.

Remarks

> **NOTE**
>
> This operation will not block.

QueryAsync(String, CancellationToken)

Asynchronously send a query to the device.

Declaration

```
Task<string> QueryAsync(string command, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	command	The SCPI command to send to the device.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.String>	The task representing the operation.

Remarks

> **NOTE**
>
> This operation will not block.

> **NOTE**
>
> This operation automatically appends a newline to the command.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.

ReadAsync(CancellationToken)

Asynchronously read from the device.

Declaration

```
Task<string> ReadAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.String>	The task representing the operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.

RefreshAsync()

Asynchronously refresh the device settings.

Declaration

```
Task RefreshAsync()
```

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

ResetAsync()

Asynchronously reset the device.

Declaration

```
Task ResetAsync()
```

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

SetTimeout(ETimeout)

Synchronously set the device timeout

Declaration

```
void SetTimeout(ETimeout timeout)
```

Parameters

TYPE	NAME	DESCRIPTION
ETimeout	timeout	The device timeout.

Uninitialize()

Synchronously uninitialize device settings and close the connection.

Declaration

```
void Uninitialize()
```

WriteAsync(String, CancellationToken)

Asynchronously write a command to the device.

Declaration

```
Task WriteAsync(string command, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	command	The SCPI command to write to the device.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	The task representing the operation.

Remarks

NOTE

This operation will not block.

NOTE

This operation automatically appends a newline to the command.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.

Interface IDevice<TModelType>

Defines general initialization and connection properties and methods for hardware devices. This includes the model type of the device.

Inherited Members

IDevice.Name
IDevice.SerialNumber
IDevice.FirmwareVersion
IDevice.Address
IDevice.IsInitialized
IDevice.InitializeAsync()
IDevice.Uninitialize()
IDevice.GetTimeout()
IDevice.SetTimeout(ETimeout)
IDevice.ResetAsync()
IDevice.RefreshAsync()
IDevice.PingAsync()
IDevice.QueryAsync(String, CancellationToken)
IDevice.WriteAsync(String, CancellationToken)
IDevice.ReadAsync(CancellationToken)
System.IDisposable.Dispose()

Namespace: Santec.Hardware.Devices.General

Assembly: Santec.Hardware.dll

Syntax

```
public interface IDevice<out TModelType> : IDevice, IDisposable where TModelType : Enum
```

Type Parameters

NAME	DESCRIPTION
TModelType	

Properties

Model

Get the model type of the device.

Declaration

```
TModelType Model { get; }
```

Property Value

TYPE	DESCRIPTION
TModelType	The model type of the device.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the property is accessed before the device has been initialized.

Interface IDeviceWithInputs

Defines the properties of a device with one or more selectable input channels and the operations that can be performed with a connection to the device.

Inherited Members

IDevice.Name
IDevice.SerialNumber
IDevice.FirmwareVersion
IDevice.Address
IDevice.IsInitialized
IDevice.InitializeAsync()
IDevice.Uninitialize()
IDevice.GetTimeout()
IDevice.SetTimeout(ETimeout)
IDevice.ResetAsync()
IDevice.RefreshAsync()
IDevice.PingAsync()
IDevice.QueryAsync(String, CancellationToken)
IDevice.WriteAsync(String, CancellationToken)
IDevice.ReadAsync(CancellationToken)
System.IDisposable.Dispose()

Namespace: Santec.Hardware.Devices.General

Assembly: Santec.Hardware.dll

Syntax

```
public interface IDeviceWithInputs : IDevice, IDisposable
```

Properties

NumberOfInputs

Get the number of inputs.

> **NOTE**
>
> The device must be initialized before this property contains the proper value.

Declaration

```
int NumberOfInputs { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	The number of device inputs.

Exceptions

TYPE	CONDITION

TYPE	CONDITION
System.InvalidOperationException	Thrown when the property is accessed before the device has been initialized.

Methods

ChangeInputChannelAsync(Int32, CancellationToken)

Asynchronously change the input channel of the device.

Declaration

```
Task ChangeInputChannelAsync(int input, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	input	The input channel to switch to.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	The task representing the operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the specified input channel is out of range for the device.
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device input channel response does not match the expected response.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

GetInputChannelAsync(CancellationToken)

Asynchronously get the current input channel of the device.

Declaration

```
Task<int> GetInputChannelAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Int32>	The task representing the operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device input channel response is not a valid input channel.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

Interface IDeviceWithWavelengthRange

Defines the properties of a device that supports a range of wavelengths. Defines the operations that can be performed with a connection to the device.

Namespace: Santec.Hardware.Devices.General

Assembly: Santec.Hardware.dll

Syntax

```
public interface IDeviceWithWavelengthRange
```

Properties

MaximumWavelength

Get the maximum wavelength of the device.

Declaration

```
int MaximumWavelength { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	The maximum wavelength of the device.

MinimumWavelength

Get the minimum wavelength of the device.

Declaration

```
int MinimumWavelength { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	The minimum wavelength of the device.

Methods

GetWavelengthAsync(CancellationToken)

Asynchronously get the wavelength of the device.

Declaration

```
Task<int> GetWavelengthAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Int32>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device wavelength response is not a valid wavelength.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

SetWavelengthAsync(Int32, CancellationToken)

Asynchronously set the wavelength of the device.

Declaration

```
Task SetWavelengthAsync(int wavelength, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	The wavelength to set the device to.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the specified wavelength is out of range for the device.
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device set wavelength response does not match the expected response.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

Interface IFiberDevice

Defines the properties of a fiber device.

Namespace: Santec.Hardware.Devices.General

Assembly: Santec.Hardware.dll

Syntax

```
public interface IFiberDevice
```

Properties

Fiber

Get the fiber information of the device.

> **NOTE**
>
> The device must be initialized before this property contains the proper value.

Declaration

```
FiberInformation Fiber { get; }
```

Property Value

TYPE	DESCRIPTION
FiberInformation	The fiber information of the device.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the property is accessed before the device has been initialized.

Interface ISantecDevice

Defines the general IP functionality for the Santec Canada family of devices.

Inherited Members

IDevice.Name
IDevice.SerialNumber
IDevice.FirmwareVersion
IDevice.Address
IDevice.IsInitialized
IDevice.InitializeAsync()
IDevice.Uninitialize()
IDevice.GetTimeout()
IDevice.SetTimeout(ETimeout)
IDevice.ResetAsync()
IDevice.RefreshAsync()
IDevice.PingAsync()
IDevice.QueryAsync(String, CancellationToken)
IDevice.WriteAsync(String, CancellationToken)
IDevice.ReadAsync(CancellationToken)
System.IDisposable.Dispose()

Namespace: Santec.Hardware.Devices.General

Assembly: Santec.Hardware.dll

Syntax

```
public interface ISantecDevice : IDevice, IDisposable
```

Methods

EnableDHCPAsync(CancellationToken)

Asynchronously enable the device to use DHCP to acquire its IP address.

Declaration

```
Task EnableDHCPAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device enable DHCP response does not match the expected response.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

GetGatewayAsync(CancellationToken)

Asynchronously get the gateway address of the device.

Declaration

```
Task<IPAddress> GetGatewayAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Net.IPAddress>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device gateway address response is not a valid IP address.

TYPE	CONDITION
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

GetHostnameAsync(CancellationToken)

Asynchronously get the mDNS hostname of the device.

Declaration

```
Task<string> GetHostnameAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.String>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device mDNS hostname response is not a valid hostname.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

GetInteractionModeAsync(CancellationToken)

Asynchronously get the device interaction mode.

> **NOTE**
>
> The device must be initialized before this property contains the proper value.

Declaration

```
Task<EInteractionMode> GetInteractionModeAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<EInteractionMode>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device response does not correspond to an interaction mode.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

GetIPAddressAsync(CancellationToken)

Asynchronously get the IP address of the device.

Declaration

```
Task<IPAddress> GetIPAddressAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Net.IPAddress>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device IP address response is not a valid IP address.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

GetSubnetPrefixLengthAsync(CancellationToken)

Asynchronously get the subnet prefix length of the device.

Declaration

```
Task<int> GetSubnetPrefixLengthAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Int32>	The task representing the asynchronous operation.

Remarks

The subnet prefix length is between `0` and `31`.

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device prefix length response is not a valid prefix length.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

SetGatewayAsync(IPAddress, CancellationToken)

Asynchronously set the gateway address of the device.

Declaration

```
Task SetGatewayAsync(IPAddress address, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Net.IPAddress	address	The gateway IP address.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device gateway address response does not match the expected response.

TYPE	CONDITION
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

SetHostnameAsync(String, CancellationToken)

Asynchronously set the mDNS hostname of the device.

Declaration

```
Task SetHostnameAsync(string hostname, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	hostname	The mDNS hostname.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the specified mDNS hostname is null, empty, or whitespace.
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the mDNS hostname response does not match the expected response.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

SetInteractionModeAsync(EInteractionMode, CancellationToken)

Asynchronously set the device interaction mode.

Declaration

```
Task SetInteractionModeAsync(EInteractionMode interactionMode, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
EInteractionMode	interactionMode	The interaction mode to set the device to.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device interaction mode response does not match the expected response.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

SetIPAddressAsync(IPAddress, CancellationToken)

Asynchronously set the IP address of the device.

Declaration

```
Task SetIPAddressAsync(IPAddress address, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Net.IPAddress	address	The IP address.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device IP address response does not match the expected response.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

SetSubnetPrefixLengthAsync(Int32, CancellationToken)

Asynchronously set the gateway address of the device.

Declaration

```
Task SetSubnetPrefixLengthAsync(int prefixLength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	prefixLength	The subnet prefix length.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	The task representing the asynchronous operation.

Remarks

The subnet prefix length must be between `0` and `31`.

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the subnet prefix length is not valid (less than `0` or greater than `31`).
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device subnet prefix length response does not match the expected response.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

Interface ISimulatedDevice

Defines the general configurable properties of a simulated device

> **NOTE**
>
> This interface extends IDevice with methods intended for configuring simulated device behaviour for testing and development. Implementations should be provided by simulated device classes created via SimulatedDeviceFactory.

Inherited Members

IDevice.Name
IDevice.SerialNumber
IDevice.FirmwareVersion
IDevice.Address
IDevice.IsInitialized
IDevice.InitializeAsync()
IDevice.Uninitialize()
IDevice.GetTimeout()
IDevice.SetTimeout(ETimeout)
IDevice.ResetAsync()
IDevice.RefreshAsync()
IDevice.PingAsync()
IDevice.QueryAsync(String, CancellationToken)
IDevice.WriteAsync(String, CancellationToken)
IDevice.ReadAsync(CancellationToken)
System.IDisposable.Dispose()

Namespace: Santec.Hardware.Devices.General

Assembly: Santec.Hardware.dll

Syntax

```
public interface ISimulatedDevice : IDevice, IDisposable
```

Properties

QueryResponses

Get the query responses returned by QueryAsync(String, CancellationToken).

Declaration

```
IReadOnlyList<QueryResponsePair> QueryResponses { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<QueryResponsePair>	The query responses configured for the simulated device.

Remarks

Use ConfigureQueryResponses(IEnumerable<QueryResponsePair>) to set this value.

Methods

ConfigureQueryResponses(IEnumerable<QueryResponsePair>)

Configure the query responses query/response pairs used by the simulated device when handling SCPI or instrument queries.

Declaration

```
void ConfigureQueryResponses(IEnumerable<QueryResponsePair> queryResponses)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Collections.Generic.IEnumerable<QueryResponsePair>	queryResponses	The collection of query/response pairs to configure. Each pair maps a query string to the response string that the simulated device should return when that query is received.

Remarks

The simulated device should match incoming queries against the configured pairs and return the associated response. This method is intended for tests and development only; production devices do not support this operation.

Exceptions

TYPE	CONDITION
System.ArgumentNullException	Thrown when `queryResponses` is null.

Interface IWavelengthDevice

Defines the properties of a discrete wavelength device.

Inherited Members

IDevice.Name
IDevice.SerialNumber
IDevice.FirmwareVersion
IDevice.Address
IDevice.IsInitialized
IDevice.InitializeAsync()
IDevice.Uninitialize()
IDevice.GetTimeout()
IDevice.SetTimeout(ETimeout)
IDevice.ResetAsync()
IDevice.RefreshAsync()
IDevice.PingAsync()
IDevice.QueryAsync(String, CancellationToken)
IDevice.WriteAsync(String, CancellationToken)
IDevice.ReadAsync(CancellationToken)
System.IDisposable.Dispose()

Namespace: Santec.Hardware.Devices.General

Assembly: Santec.Hardware.dll

Syntax

```
public interface IWavelengthDevice : IDevice, IDisposable
```

Properties

Wavelengths

Get the wavelengths supported by the device.

> **NOTE**
>
> The device must be initialized before this property contains the proper value.

Declaration

```
List<int> Wavelengths { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.List<System.Int32>	A list of the device's wavelengths.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the property is accessed before the device has been initialized.

Class PhysicalDevice

Provides a connection to the device and general device properties and operations for physical hardware devices.

Inheritance

System.Object
PhysicalDevice
SantecDevice

Implements

IDevice
System.IDisposable

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.General

Assembly: Santec.Hardware.dll

Syntax

```
public abstract class PhysicalDevice : IDevice, IDisposable
```

Properties

Address

Declaration

```
public string Address { get; }
```

Property Value

TYPE	DESCRIPTION
System.String	

FirmwareVersion

Declaration

```
public Version FirmwareVersion { get; protected set; }
```

Property Value

TYPE	DESCRIPTION
Version	

IsInitialized

Declaration

```
public bool IsInitialized { get; protected set; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	

IsInitializing

Get whether the device and connection are initializing.

Declaration

```
protected bool IsInitializing { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	`true` if the device is initializing; otherwise, `false`.

Name

Declaration

```
public abstract string Name { get; }
```

Property Value

TYPE	DESCRIPTION
System.String	

SerialNumber

Declaration

```
public string SerialNumber { get; }
```

Property Value

TYPE	DESCRIPTION
System.String	

Methods

Dispose()

Close and clean up the device, device connection, and any associated resources.

Declaration

```
public void Dispose()
```

Dispose(Boolean)

Close and clean up the device, device connection, and any associated resources.

Declaration

```
protected virtual void Dispose(bool disposing)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Boolean	disposing	`true` if the method is called from Dispose; otherwise, `false`.

Finalize()

Finalizer for the PhysicalDevice class. This finalizer is called when the object is being garbage collected.

Declaration

```
protected void Finalize()
```

GetTimeout()

Declaration

```
public ETimeout GetTimeout()
```

Returns

TYPE	DESCRIPTION
ETimeout	

InitializeAsync()

Declaration

```
public virtual async Task InitializeAsync()
```

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

PingAsync()

Declaration

```
public async Task<bool> PingAsync()
```

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Boolean>	

QueryAsync(String, CancellationToken)

Declaration

```
public Task<string> QueryAsync(string command, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	command	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.String>	

ReadAsync(CancellationToken)

Declaration

```
public Task<string> ReadAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.String>	

RefreshAsync()

Declaration

```
public abstract Task RefreshAsync()
```

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

ResetAsync()

Declaration

```
public async Task ResetAsync()
```

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device reset response does not match the expected response.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

SetTimeout(ETimeout)

Declaration

```
public void SetTimeout(ETimeout timeout)
```

Parameters

TYPE	NAME	DESCRIPTION
ETimeout	timeout	

Uninitialize()

Declaration

```
public virtual void Uninitialize()
```

WriteAsync(String, CancellationToken)

Declaration

```
public Task WriteAsync(string command, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	command	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Implements

IDevice
System.IDisposable

Class QueryResponsePair

Represents a single mapping between a query string and the response string that a simulated device should return for that query.

Inheritance

System.Object
QueryResponsePair

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.General

Assembly: Santec.Hardware.dll

Syntax

```
public sealed class QueryResponsePair
```

Constructors

QueryResponsePair(String, String)

Initializes a new instance of the QueryResponsePair class.

Declaration

```
public QueryResponsePair(string query, string response)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	query	The query string to match (e.g. a SCPI query).
System.String	response	The response string to return when `query` is received.

Exceptions

TYPE	CONDITION
System.ArgumentNullException	Thrown when `query` or `response` is null.

Properties

Query

The query to match.

Declaration

```
public string Query { get; }
```

Property Value

TYPE	DESCRIPTION
System.String	The query to match.

Response

The response to return when the associated query is received.

Declaration

```
public string Response { get; }
```

Property Value

TYPE	DESCRIPTION
System.String	The response to return.

Class RDSPDetector

Represents an RD-SP detector of a IDetectorDevice.

Inheritance

System.Object
DeviceDetector
DeviceDetectorWithSerialNumber
RDSPDetector

Implements

IModule

Inherited Members

DeviceDetectorWithSerialNumber.SerialNumber
DeviceDetector.Index
DeviceDetector.IsRDSP
DeviceDetector.FirmwareVersion
System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.General

Assembly: Santec.Hardware.dll

Syntax

```
public class RDSPDetector : DeviceDetectorWithSerialNumber, IModule
```

Properties

CameraSerialNumber

Get the RD-SP camera serial number.

Declaration

```
public string CameraSerialNumber { get; }
```

Property Value

TYPE	DESCRIPTION
System.String	The RD-SP camera serial number.

Implements

IModule

Class ReadingValue

Represents a device reading value.

> **NOTE**
>
> This class should not be used directly. It should only be used when returned by a device reading.

Inheritance

System.Object
ReadingValue

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.General

Assembly: Santec.Hardware.dll

Syntax

```
public class ReadingValue
```

Constructors

ReadingValue(Double, ERangeStatus)

Initializes a new reading value.

Declaration

```
public ReadingValue(double value, ERangeStatus rangeStatus)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Double	value	The numeric value of the reading.
ERangeStatus	rangeStatus	The range status of the reading.

Properties

RangeStatus

Get the range status of the reading.

Declaration

```
public ERangeStatus RangeStatus { get; }
```

Property Value

TYPE	DESCRIPTION
ERangeStatus	The range status of the reading.

Value

Get the numeric value of the reading.

Declaration

```
public double Value { get; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The numeric value of the reading.

Remarks

> **NOTE**
>
> This property may be set to `double.NaN` if the device could not properly measure the reading.

Methods

FromDouble(Double)

Create a reading value from a double reading value.

> **NOTE**
>
> If the value is `double.NaN`, the RangeStatus of the reading will be set to none. Otherwise, the RangeStatus of the reading will be set to `ERangeStatus.InRange`.

Declaration

```
public static ReadingValue FromDouble(double value)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Double	value	The numeric value of the reading.

Returns

TYPE	DESCRIPTION
ReadingValue	The reading value.

Class SantecDevice

Provides a connection to a Santec Canada family device and provides the general IP functionality for the device.

Inheritance

System.Object
PhysicalDevice
SantecDevice
OVA
BRM
XCS
PTM
PCU
PEM
PLM
OPM
RLM
OSX

Implements

ISantecDevice
IDevice
System.IDisposable

Inherited Members

PhysicalDevice.Address
PhysicalDevice.Name
PhysicalDevice.SerialNumber
PhysicalDevice.FirmwareVersion
PhysicalDevice.IsInitialized
PhysicalDevice.IsInitializing
PhysicalDevice.Uninitialize()
PhysicalDevice.SetTimeout(ETimeout)
PhysicalDevice.GetTimeout()
PhysicalDevice.Dispose()
PhysicalDevice.Dispose(Boolean)
PhysicalDevice.PingAsync()
PhysicalDevice.ResetAsync()
PhysicalDevice.RefreshAsync()
PhysicalDevice.QueryAsync(String, CancellationToken)
PhysicalDevice.WriteAsync(String, CancellationToken)
PhysicalDevice.ReadAsync(CancellationToken)
System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.General

Assembly: Santec.Hardware.dll

Syntax

```
public abstract class SantecDevice : PhysicalDevice, ISantecDevice, IDevice, IDisposable
```

Methods

EnableDHCPAsync(CancellationToken)

Declaration

```
public async Task EnableDHCPAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

GetGatewayAsync(CancellationToken)

Declaration

```
public async Task<IPAddress> GetGatewayAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Net.IPAddress>	

GetHostnameAsync(CancellationToken)

Declaration

```
public async Task<string> GetHostnameAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.String>	

GetInteractionModeAsync(CancellationToken)

Declaration

```
public async Task<EInteractionMode> GetInteractionModeAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<EInteractionMode>	

GetIPAddressAsync(CancellationToken)

Declaration

```
public async Task<IPAddress> GetIPAddressAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Net.IPAddress>	

GetSubnetPrefixLengthAsync(CancellationToken)

Declaration

```
public async Task<int> GetSubnetPrefixLengthAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Int32>	

HandleIDNResponse(String[])

Handle the *IDN? response parts.

> **NOTE**
>
> Use this method to handle the *IDN? response parts as needed for Santec devices.

Declaration

```
protected virtual void HandleIDNResponse(string[] idnResponseParts)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String[]	idnResponseParts	

InitializeAsync()

Declaration

```
public override async Task InitializeAsync()
```

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Overrides

PhysicalDevice.InitializeAsync()

Exceptions

TYPE	CONDITION
UnexpectedDeviceResponseException	Thrown when the device *IDN? response does not have the correct format.
UnexpectedDeviceResponseException	Thrown when the device firmware version part of the *IDN? response does not have the correct format.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

SetGatewayAsync(IPAddress, CancellationToken)

Declaration

```
public async Task SetGatewayAsync(IPAddress address, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Net.IPAddress	address	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetHostnameAsync(String, CancellationToken)

Declaration

```
public async Task SetHostnameAsync(string hostname, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	hostname	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetInteractionModeAsync(EInteractionMode, CancellationToken)

Declaration

```
public async Task SetInteractionModeAsync(EInteractionMode interactionMode, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
EInteractionMode	interactionMode	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetIPAddressAsync(IPAddress, CancellationToken)

Declaration

```
public async Task SetIPAddressAsync(IPAddress address, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Net.IPAddress	address	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetSubnetPrefixLengthAsync(Int32, CancellationToken)

Declaration

```
public async Task SetSubnetPrefixLengthAsync(int prefixLength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	prefixLength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Implements

ISantecDevice

IDevice

System.IDisposable

Class SimulatedDeviceFactory

Provides a factory for creating simulated device connections.

> **NOTE**
>
> This class should be used for the creation of all simulated devices. However, the created simulated devices should not be used directly. They should only be used when returned as part of a list of devices by a DetectHardwareAsync(EConnectionTypesToDetect) or DetectHardwareAsync<T>(EConnectionTypesToDetect) call to a SimulatedHardwareDetectionService.

Inheritance

System.Object
SimulatedDeviceFactory

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.General

Assembly: Santec.Hardware.dll

Syntax

```
public class SimulatedDeviceFactory
```

Constructors

SimulatedDeviceFactory()

Initializes a new simulated device factory.

Declaration

```
public SimulatedDeviceFactory()
```

SimulatedDeviceFactory(ICultureService)

Initializes a new simulated device factory.

Declaration

```
public SimulatedDeviceFactory(ICultureService cultureService)
```

Parameters

TYPE	NAME	DESCRIPTION
Santec.Core.Globalization.ICultureService	cultureService	The culture service for localization.

Methods

CreateSimulatedBRM(String, IEnumerable<Int32>, Int32, IEnumerable<Int32>, Version, EFiberType, ECoreSize)

Creates a new simulated BRM with default reading ranges.

Declaration

```
public SimulatedBRM CreateSimulatedBRM(string serialNumber, IEnumerable<int> wavelengths, int
numberOfDetectors, IEnumerable<int> switchChannelCounts, Version firmwareVersion = null, EFiberType
fiberType = EFiberType.Unknown, ECoreSize coreSize = ECoreSize.Unknown)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	serialNumber	The serial number of the BRM.
System.Collections.Generic.IEnumerable<System.Int32>	wavelengths	The wavelengths of the BRM.
System.Int32	numberOfDetectors	The number of detectors for the BRM.
System.Collections.Generic.IEnumerable<System.Int32>	switchChannelCounts	The channel count for BRM switches.
Version	firmwareVersion	The firmware version of the BRM.
EFiberType	fiberType	The fiber type of the BRM.
ECoreSize	coreSize	The core size of the BRM.

Returns

TYPE	DESCRIPTION
SimulatedBRM	

Remarks

The switch channel counts will be assigned to the switch indexes 0, 1, and 2 in order. The output switch must have between 1 and 2 channels. A value of 0 will be converted into no switch at the given index.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the simulated BRM is given an invalid number of wavelengths or detectors.
System.ArgumentException	Thrown when the simulated BRM is provided an invalid number of switches.

TYPE	CONDITION
System.ArgumentException	Thrown when any simulated BRM switch is given an invalid number of channels.

CreateSimulatedOSX(String, IEnumerable<SimulatedSwitchModuleInformation>, Version)

Creates a new simulated OSX.

Declaration

```
public SimulatedOSX CreateSimulatedOSX(string serialNumber, IEnumerable<SimulatedSwitchModuleInformation>
modulesInformation, Version firmwareVersion = null)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	serialNumber	The serial number of the OSX.
System.Collections.Generic.IEnumerable<SimulatedSwitchModuleInformation>	modulesInformation	The information of the simulated switch modules.
Version	firmwareVersion	The firmware version of the OSX.

Returns

TYPE	DESCRIPTION
SimulatedOSX	

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the simulated OSX is given an invalid number of modules.
System.ArgumentException	Thrown when any OSX module is given an invalid number of channels or commons.

CreateSimulatedPCU(String, Int32, Version)

Creates a new simulated PCU.

Declaration

```
public SimulatedPCU CreateSimulatedPCU(string serialNumber, int numberOfInputs, Version firmwareVersion =
null)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	serialNumber	The serial number of the PCU.
System.Int32	numberOfInputs	The number of inputs of the PCU.
Version	firmwareVersion	The firmware version of the PCU.

Returns

TYPE	DESCRIPTION
SimulatedPCU	

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the simulated PCU is given an invalid number of inputs.

CreateSimulatedPEM(String, Int32, Int32, Version)

Creates a new simulated PEM with default reading ranges.

Declaration

```
public SimulatedPEM CreateSimulatedPEM(string serialNumber, int wavelength, int numberOfOutputs, Version
firmwareVersion = null)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	serialNumber	The serial number of the PEM.
System.Int32	wavelength	The wavelength of the PEM.
System.Int32	numberOfOutputs	The number of outputs on the PEM.
Version	firmwareVersion	The firmware version of the PEM.

Returns

TYPE	DESCRIPTION
SimulatedPEM	

CreateSimulatedPLM(String, IEnumerable<Int32>, Int32, IEnumerable<Int32>, Version, EFiberType, ECoreSize)

Creates a new simulated PLM with default reading ranges.

Declaration

```
public SimulatedPLM CreateSimulatedPLM(string serialNumber, IEnumerable<int> wavelengths, int
numberOfDetectors, IEnumerable<int> switchChannelCounts, Version firmwareVersion = null, EFiberType
fiberType = EFiberType.Unknown, ECoreSize coreSize = ECoreSize.Unknown)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	serialNumber	The serial number of the PLM.
System.Collections.Generic.IEnumerable<System.Int32>	wavelengths	The wavelengths of the PLM.
System.Int32	numberOfDetectors	The number of detectors for the PLM.
System.Collections.Generic.IEnumerable<System.Int32>	switchChannelCounts	The channel count for PLM switches.
Version	firmwareVersion	The firmware version of the PLM.
EFiberType	fiberType	The fiber type of the RLM.
ECoreSize	coreSize	The core size of the RLM.

Returns

TYPE	DESCRIPTION
SimulatedPLM	

Remarks

The switch channel counts will be assigned to the switch indexes 0, 1, and 2 in order. The output switch must have between 1 and 2 channels. A value of 0 will be converted into no switch at the given index.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the simulated PLM is given an invalid number of wavelengths or detectors.
System.ArgumentException	Thrown when the simulated PLM is provided an invalid number of switches.

TYPE	CONDITION
System.ArgumentException	Thrown when any simulated PLM switch is given an invalid number of channels.

CreateSimulatedPTM(String, Int32, Version)

Initialize a new simulated PTM with a default polarity mapping.

Declaration

```
public SimulatedPTM CreateSimulatedPTM(string serialNumber, int numberOfChannels, Version firmwareVersion =
null)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	serialNumber	The serial number of the PTM.
System.Int32	numberOfChannels	The number of channels on the PTM.
Version	firmwareVersion	The firmware version of the PTM.

Returns

TYPE	DESCRIPTION
SimulatedPTM	

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the simulated PTM is given an invalid number of channels.

CreateSimulatedRLM(String, IEnumerable<Int32>, Int32, IEnumerable<Int32>, Version, EFiberType, ECoreSize, ERLMModel)

Creates a new simulated RLM with default reading ranges.

Declaration

```
public SimulatedRLM CreateSimulatedRLM(string serialNumber, IEnumerable<int> wavelengths, int
numberOfDetectors, IEnumerable<int> switchChannelCounts, Version firmwareVersion = null, EFiberType
fiberType = EFiberType.Unknown, ECoreSize coreSize = ECoreSize.Unknown, ERLMModel model = ERLMModel.RLM100)
```

Parameters

TYPE	NAME	DESCRIPTION

TYPE	NAME	DESCRIPTION
System.String	serialNumber	The serial number of the RLM.
System.Collections.Generic.IEnumerable<System.Int32>	wavelengths	The wavelengths of the RLM.
System.Int32	numberOfDetectors	The number of detectors for the RLM.
System.Collections.Generic.IEnumerable<System.Int32>	switchChannelCounts	The channel count for RLM switches.
Version	firmwareVersion	The firmware version of the RLM.
EFiberType	fiberType	The fiber type of the RLM.
ECoreSize	coreSize	The core size of the RLM.
ERLMModel	model	The model of the RLM.

Returns

TYPE	DESCRIPTION
SimulatedRLM	

Remarks

The switch channel counts will be assigned to the switch indexes 0, 1, and 2 in order. The output switch must have between 1 and 2 channels. A value of 0 will be converted into no switch at the given index.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the simulated RLM is given an invalid number of wavelengths or detectors.
System.ArgumentException	Thrown when the simulated RLM is provided an invalid number of switches.
System.ArgumentException	Thrown when any simulated RLM switch is given an invalid number of channels.

CreateSimulatedRLM(String, IEnumerable<Int32>, Int32, Int32, IEnumerable<SimulatedExternalDeviceSwitchInformation>, Version, EFiberType, ECoreSize, ERLMModel)

Creates a new simulated RLM with default reading ranges.

Declaration

```
public SimulatedRLM CreateSimulatedRLM(string serialNumber, IEnumerable<int> wavelengths, int
numberOfDetectors, int numberOfOutputs, IEnumerable<SimulatedExternalDeviceSwitchInformation>
externalDeviceSwitches, Version firmwareVersion = null, EFiberType fiberType = EFiberType.Unknown, ECoreSize
coreSize = ECoreSize.Unknown, ERLMModel model = ERLMModel.RLM100)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	serialNumber	The serial number of the RLM.
System.Collections.Generic.IEnumerable<System.Int32>	wavelengths	The wavelengths of the RLM.
System.Int32	numberOfDetectors	The number of detectors for the RLM.
System.Int32	numberOfOutputs	The number of outputs for the RLM.
System.Collections.Generic.IEnumerable<SimulatedExternalDeviceSwitchInformation>	externalDeviceSwitches	The information of external switches for the RLM.
Version	firmwareVersion	The firmware version of the RLM.
EFiberType	fiberType	The fiber type of the RLM.
ECoreSize	coreSize	The core size of the RLM.
ERLMModel	model	The model of the RLM.

Returns

TYPE	DESCRIPTION
SimulatedRLM	

Remarks

The number of outputs must be between 1 and 2. The external device switches will be assigned to the switch indexes 1, and 2 in

order. A null value will be converted into no switch at the given index.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the simulated RLM is given an invalid number of wavelengths or detectors.
System.ArgumentException	Thrown when the simulated RLM is given an invalid number of outputs.
System.ArgumentException	Thrown when any simulated RLM is given an invalid number of external switches.

CreateSimulatedSLS(String, IEnumerable<SimulatedSourceModuleInformation>, Version)

Creates a new simulated SLS with no modules.

Declaration

```
public SimulatedSLS CreateSimulatedSLS(string serialNumber, IEnumerable<SimulatedSourceModuleInformation>
modules, Version firmwareVersion = null)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	serialNumber	The serial number of the SLS
System.Collections.Generic.IEnumerable<SimulatedSourceModuleInformation>	modules	
Version	firmwareVersion	The firmware version of the SLS

Returns

TYPE	DESCRIPTION
SimulatedSLS	

CreateSimulatedXCS(String, IEnumerable<IModuleInformation>, Version)

Creates a new simulated XCS with no modules.

Declaration

```
public SimulatedXCS CreateSimulatedXCS(string serialNumber, IEnumerable<IModuleInformation> modules, Version
firmwareVersion = null)
```

Parameters

TYPE	NAME	DESCRIPTION

TYPE	NAME	DESCRIPTION
System.String	serialNumber	The serial number of the XCS
System.Collections.Generic.IEnumerable<IModuleInformation>	modules	The modules of the XCS
Version	firmwareVersion	The firmware version of the SLS

Returns

TYPE	DESCRIPTION
SimulatedXCS	

Class SimulatedSantecDevice

Provides the general functionality for a simulated Santec Canada family device.

Inheritance

System.Object
SimulatedSantecDevice
SimulatedBRM
SimulatedXCS
SimulatedPTM
SimulatedPCU
SimulatedPEM
SimulatedPLM
SimulatedRLM
SimulatedOSX

Implements

ISantecDevice
ISimulatedDevice
IDevice
System.IDisposable

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.General

Assembly: Santec.Hardware.dll

Syntax

```
public abstract class SimulatedSantecDevice : ISantecDevice, ISimulatedDevice, IDevice, IDisposable
```

Fields

creatingDevice

Field to indicate if the simulated device is being created.

Declaration

```
protected bool creatingDevice
```

Field Value

TYPE	DESCRIPTION
System.Boolean	

Properties

Address

Declaration

```
public string Address { get; }
```

Property Value

TYPE	DESCRIPTION
System.String	

Remarks

> **NOTE**
>
> This property returns "N/A" for the simulated device unless an IP address has been explicitly set.

FirmwareVersion

Declaration

```
public Version FirmwareVersion { get; protected set; }
```

Property Value

TYPE	DESCRIPTION
Version	

IsInitialized

Declaration

```
public bool IsInitialized { get; protected set; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	

Name

Declaration

```
public abstract string Name { get; }
```

Property Value

TYPE	DESCRIPTION
System.String	

QueryResponses

Declaration

```
public IReadOnlyList<QueryResponsePair> QueryResponses { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<QueryResponsePair>	

SerialNumber

Declaration

```
public string SerialNumber { get; }
```

Property Value

TYPE	DESCRIPTION
System.String	

Methods

ConfigureQueryResponses(IEnumerable<QueryResponsePair>)

Declaration

```
public void ConfigureQueryResponses(IEnumerable<QueryResponsePair> queryResponsePairs)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Collections.Generic.IEnumerable<QueryResponsePair>	queryResponsePairs	

Dispose()

Declaration

```
public void Dispose()
```

EnableDHCPAsync(CancellationToken)

Declaration

```
public async Task EnableDHCPAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

GetGatewayAsync(CancellationToken)

Declaration

```
public async Task<IPAddress> GetGatewayAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Net.IPAddress>	

GetHostnameAsync(CancellationToken)

Declaration

```
public async Task<string> GetHostnameAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.String>	

GetInteractionModeAsync(CancellationToken)

Declaration

```
public async Task<EInteractionMode> GetInteractionModeAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<EInteractionMode>	

GetIPAddressAsync(CancellationToken)

Declaration

```
public async Task<IPAddress> GetIPAddressAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Net.IPAddress>	

GetSubnetPrefixLengthAsync(CancellationToken)

Declaration

```
public async Task<int> GetSubnetPrefixLengthAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Int32>	

GetTimeout()

Declaration

```
public ETimeout GetTimeout()
```

Returns

TYPE	DESCRIPTION
ETimeout	

InitializeAsync()

Declaration

```
public async Task InitializeAsync()
```

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

PingAsync()

Declaration

```
public async Task<bool> PingAsync()
```

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Boolean>	

QueryAsync(String, CancellationToken)

Declaration

```
public async Task<string> QueryAsync(string command, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	command	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.String>	

ReadAsync(CancellationToken)

Declaration

```
public Task<string> ReadAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.String>	

RefreshAsync()

Declaration

```
public async Task RefreshAsync()
```

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

ResetAsync()

Declaration

```
public async Task ResetAsync()
```

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetGatewayAsync(IPAddress, CancellationToken)

Declaration

```
public async Task SetGatewayAsync(IPAddress address, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Net.IPAddress	address	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetHostnameAsync(String, CancellationToken)

Declaration

```
public async Task SetHostnameAsync(string hostname, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	hostname	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetInteractionModeAsync(EInteractionMode, CancellationToken)

Declaration

```
public async Task SetInteractionModeAsync(EInteractionMode interactionMode, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
EInteractionMode	interactionMode	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetIPAddressAsync(IPAddress, CancellationToken)

Declaration

```
public async Task SetIPAddressAsync(IPAddress address, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Net.IPAddress	address	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetSubnetPrefixLengthAsync(Int32, CancellationToken)

Declaration

```
public async Task SetSubnetPrefixLengthAsync(int prefixLength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	prefixLength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetTimeout(ETimeout)

Declaration

```
public void SetTimeout(ETimeout timeout)
```

Parameters

TYPE	NAME	DESCRIPTION
ETimeout	timeout	

Uninitialize()

Declaration

```
public void Uninitialize()
```

WriteAsync(String, CancellationToken)

Declaration

```
public Task WriteAsync(string command, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	command	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Implements

ISantecDevice
ISimulatedDevice
IDevice
System.IDisposable

Namespace Santec.Hardware.Devices.InsertionLoss

Interfaces

IInsertionLossMeter

Defines the properties of a insertion loss meter and the operations that can be performed with a connection to the device.

Interface IInsertionLossMeter

Defines the properties of a insertion loss meter and the operations that can be performed with a connection to the device.

Inherited Members

IDetectorDevice.NumberOfDetectors
IDetectorDevice.Detectors
IDevice.Name
IDevice.SerialNumber
IDevice.FirmwareVersion
IDevice.Address
IDevice.IsInitialized
IDevice.InitializeAsync()
IDevice.Uninitialize()
IDevice.GetTimeout()
IDevice.SetTimeout(ETimeout)
IDevice.ResetAsync()
IDevice.RefreshAsync()
IDevice.PingAsync()
IDevice.QueryAsync(String, CancellationToken)
IDevice.WriteAsync(String, CancellationToken)
IDevice.ReadAsync(CancellationToken)
System.IDisposable.Dispose()

Namespace: Santec.Hardware.Devices.InsertionLoss

Assembly: Santec.Hardware.dll

Syntax

```
public interface IInsertionLossMeter : IDetectorDevice, IDevice, IDisposable
```

Methods

GetILReferenceAsync(Int32, Int32, CancellationToken)

Asynchronously get the stored IL reference for a detector at a given wavelength.

Declaration

```
Task<double> GetILReferenceAsync(int detector, int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	The detector to get the reference for.
System.Int32	wavelength	The wavelength to get the reference for.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the device does not support the specified wavelength.
System.ArgumentException	Thrown when the specified detector is out of range for the device.
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device IL reference query response is not a valid number.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

ReadILAsync(Int32, Int32, CancellationToken)

Asynchronously read the insertion loss from a detector at a given wavelength.

Declaration

```
Task<double> ReadILAsync(int detector, int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	The detector number of the detector to read the insertion loss from.
System.Int32	wavelength	The wavelength to read the insertion loss at.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the device does not support the specified wavelength.
System.ArgumentException	Thrown when the specified detector is out of range for the device.
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device read IL response is not a valid number.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

ReferenceILAsync(Int32, Int32, Boolean, CancellationToken)

Asynchronously perform an insertion loss reference for all detectors at a given wavelength.

Declaration

```
Task<double> ReferenceILAsync(int detector, int wavelength, bool applyReferenceToAllDetectors,
CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	The detector number of the detector to perform the reference for.
System.Int32	wavelength	The wavelength to perform the reference at.
System.Boolean	applyReferenceToAllDetectors	Indicate if the reference read should be applied to all detectors for the current channel.

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the device does not support the specified wavelength.
System.ArgumentException	Thrown when the specified detector is out of range for the device.
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device reference IL response is not a valid number.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

ReferenceILAsync(Int32, Int32, CancellationToken)

Asynchronously perform an insertion loss reference for a detector at a given wavelength.

Declaration

```
Task<double> ReferenceILAsync(int detector, int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	The detector number of the detector to perform the reference for.

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	The wavelength to perform the reference at.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the device does not support the specified wavelength.
System.ArgumentException	Thrown when the specified detector is out of range for the device.
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device reference IL response is not a valid number.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

SetILReferenceAsync(Int32, Int32, Double, CancellationToken)

Asynchronously set the IL reference for a detector at a given wavelength.

Declaration

```
Task SetILReferenceAsync(int detector, int wavelength, double reference, CancellationToken cancellationToken
= default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	The detector to set the reference for.
System.Int32	wavelength	The wavelength to set the reference for.
System.Double	reference	The IL reference value.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	The task representing the asynchronous operation.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the device does not support the specified wavelength.
System.ArgumentException	Thrown when the specified detector is out of range for the device.
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device IL reference query response does not match the expected response.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

Namespace Santec.Hardware.Devices.Modular

Classes

SimulatedSLS

Simulated SLS device.

SimulatedXCS

Provides a simulated XCS device with a single source module.

XCS

Provides a connection to an XCS along with properties of the device and operations that can be performed using it.

> **NOTE**
>
> This class cannot not be used directly. It can only be used when returned:
>
> 1. As part of a list of devices by a DetectHardwareAsync(EConnectionTypesToDetect) or DetectHardwareAsync<T>(EConnectionTypesToDetect) call.
> 2. As a device by a TryDetectIPDeviceAsync(IPAddress) or TryDetectIPDeviceAsync<T>(IPAddress) call.

Interfaces

IModuleController

Defines the properties of a device that contains and controls one or more modules. Defines the operations that can be performed with a connection to the device.

ISLS

Defines the properties of an SLS and operations that can be performed with a connection to the device.

ISubModuleController

Defines the properties of a module that contains and controls one or more submodules.

IXCS

Defines the properties of an XCS and operations that can be performed with a connection to the device.

Enums

EXCSModel

Specifies the model of an XCS.

Enum EXCSModel

Specifies the model of an XCS.

Namespace: Santec.Hardware.Devices.Modular

Assembly: Santec.Hardware.dll

Syntax

```
public enum EXCSModel
```

Fields

NAME	DESCRIPTION
XCS100	The XCS-100 model.

Interface IModuleController

Defines the properties of a device that contains and controls one or more modules. Defines the operations that can be performed with a connection to the device.

Inherited Members

IDevice.Name
IDevice.SerialNumber
IDevice.FirmwareVersion
IDevice.Address
IDevice.IsInitialized
IDevice.InitializeAsync()
IDevice.Uninitialize()
IDevice.GetTimeout()
IDevice.SetTimeout(ETimeout)
IDevice.ResetAsync()
IDevice.RefreshAsync()
IDevice.PingAsync()
IDevice.QueryAsync(String, CancellationToken)
IDevice.WriteAsync(String, CancellationToken)
IDevice.ReadAsync(CancellationToken)
System.IDisposable.Dispose()

Namespace: Santec.Hardware.Devices.Modular

Assembly: Santec.Hardware.dll

Syntax

```
public interface IModuleController : IDevice, IDisposable
```

Properties

Modules

Get the modules of the device.

> **NOTE**
>
> The device must be initialized before this property contains the proper value.

Declaration

```
IReadOnlyList<IModule> Modules { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<IModule>	The modules of the device.

Exceptions

TYPE	CONDITION

TYPE	CONDITION
System.InvalidOperationException	Thrown when the property is accessed before the device has been initialized.

NumberOfModules

Get the number of modules in the device.

> **NOTE**
>
> The device must be initialized before this property contains the proper value.

Declaration

```
int NumberOfModules { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	The number of modules in the device.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the property is accessed before the device has been initialized.

Interface ISLS

Defines the properties of an SLS and operations that can be performed with a connection to the device.

Inherited Members

ISantecDevice.GetIPAddressAsync(CancellationToken)
ISantecDevice.SetIPAddressAsync(IPAddress, CancellationToken)
ISantecDevice.EnableDHCPAsync(CancellationToken)
ISantecDevice.GetGatewayAsync(CancellationToken)
ISantecDevice.SetGatewayAsync(IPAddress, CancellationToken)
ISantecDevice.GetSubnetPrefixLengthAsync(CancellationToken)
ISantecDevice.SetSubnetPrefixLengthAsync(Int32, CancellationToken)
ISantecDevice.GetHostnameAsync(CancellationToken)
ISantecDevice.SetHostnameAsync(String, CancellationToken)
ISantecDevice.GetInteractionModeAsync(CancellationToken)
ISantecDevice.SetInteractionModeAsync(EInteractionMode, CancellationToken)
IModuleController.NumberOfModules
IModuleController.Modules
IDevice<EXCSModel>.Model
IDevice.Name
IDevice.SerialNumber
IDevice.FirmwareVersion
IDevice.Address
IDevice.IsInitialized
IDevice.InitializeAsync()
IDevice.Uninitialize()
IDevice.GetTimeout()
IDevice.SetTimeout(ETimeout)
IDevice.ResetAsync()
IDevice.RefreshAsync()
IDevice.PingAsync()
IDevice.QueryAsync(String, CancellationToken)
IDevice.WriteAsync(String, CancellationToken)
IDevice.ReadAsync(CancellationToken)
System.IDisposable.Dispose()

Namespace: Santec.Hardware.Devices.Modular

Assembly: Santec.Hardware.dll

Syntax

```
public interface ISLS : IXCS, ISantecDevice, IModuleController, IDevice<EXCSModel>, IDevice, IDisposable
```

Interface ISubModuleController

Defines the properties of a module that contains and controls one or more submodules.

Inherited Members

IModule.Index

Namespace: Santec.Hardware.Devices.Modular

Assembly: Santec.Hardware.dll

Syntax

```
public interface ISubModuleController : IModule
```

Properties

NumberOfSubModules

Get the number of submodules in the module.

> **NOTE**
>
> The module must be initialized before this property contains the proper value.

Declaration

```
int NumberOfSubModules { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	The number of submodules in the module.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the property is accessed before the module has been initialized.

SubModules

Get the submodules of the module.

> **NOTE**
>
> The module must be initialized before this property contains the proper value.

Declaration

```
IReadOnlyList<IModule> SubModules { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<IModule>	The submodules of the module.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the property is accessed before the module has been initialized.

Interface IXCS

Defines the properties of an XCS and operations that can be performed with a connection to the device.

Inherited Members

ISantecDevice.GetIPAddressAsync(CancellationToken)
ISantecDevice.SetIPAddressAsync(IPAddress, CancellationToken)
ISantecDevice.EnableDHCPAsync(CancellationToken)
ISantecDevice.GetGatewayAsync(CancellationToken)
ISantecDevice.SetGatewayAsync(IPAddress, CancellationToken)
ISantecDevice.GetSubnetPrefixLengthAsync(CancellationToken)
ISantecDevice.SetSubnetPrefixLengthAsync(Int32, CancellationToken)
ISantecDevice.GetHostnameAsync(CancellationToken)
ISantecDevice.SetHostnameAsync(String, CancellationToken)
ISantecDevice.GetInteractionModeAsync(CancellationToken)
ISantecDevice.SetInteractionModeAsync(EInteractionMode, CancellationToken)
IModuleController.NumberOfModules
IModuleController.Modules
IDevice<EXCSModel>.Model
IDevice.Name
IDevice.SerialNumber
IDevice.FirmwareVersion
IDevice.Address
IDevice.IsInitialized
IDevice.InitializeAsync()
IDevice.Uninitialize()
IDevice.GetTimeout()
IDevice.SetTimeout(ETimeout)
IDevice.ResetAsync()
IDevice.RefreshAsync()
IDevice.PingAsync()
IDevice.QueryAsync(String, CancellationToken)
IDevice.WriteAsync(String, CancellationToken)
IDevice.ReadAsync(CancellationToken)
System.IDisposable.Dispose()

Namespace: Santec.Hardware.Devices.Modular

Assembly: Santec.Hardware.dll

Syntax

```
public interface IXCS : ISantecDevice, IModuleController, IDevice<EXCSModel>, IDevice, IDisposable
```

Class SimulatedSLS

Simulated SLS device.

Inheritance

System.Object
SimulatedSantecDevice
SimulatedXCS
SimulatedSLS

Implements

ISimulatedDevice
IXCS
ISantecDevice
IModuleController
IDevice<EXCSModel>
IDevice
System.IDisposable

Inherited Members

SimulatedXCS.NumberOfModules
SimulatedXCS.Modules
SimulatedXCS.Model
SimulatedXCS.AddModules(IEnumerable<IModule>)
SimulatedSantecDevice.creatingDevice
SimulatedSantecDevice.Dispose()
SimulatedSantecDevice.SerialNumber
SimulatedSantecDevice.FirmwareVersion
SimulatedSantecDevice.Address
SimulatedSantecDevice.IsInitialized
SimulatedSantecDevice.QueryResponses
SimulatedSantecDevice.ConfigureQueryResponses(IEnumerable<QueryResponsePair>)
SimulatedSantecDevice.GetTimeout()
SimulatedSantecDevice.SetTimeout(ETimeout)
SimulatedSantecDevice.GetInteractionModeAsync(CancellationToken)
SimulatedSantecDevice.GetIPAddressAsync(CancellationToken)
SimulatedSantecDevice.SetIPAddressAsync(IPAddress, CancellationToken)
SimulatedSantecDevice.EnableDHCPAsync(CancellationToken)
SimulatedSantecDevice.GetGatewayAsync(CancellationToken)
SimulatedSantecDevice.SetGatewayAsync(IPAddress, CancellationToken)
SimulatedSantecDevice.GetSubnetPrefixLengthAsync(CancellationToken)
SimulatedSantecDevice.SetSubnetPrefixLengthAsync(Int32, CancellationToken)
SimulatedSantecDevice.GetHostnameAsync(CancellationToken)
SimulatedSantecDevice.SetHostnameAsync(String, CancellationToken)
SimulatedSantecDevice.SetInteractionModeAsync(EInteractionMode, CancellationToken)
SimulatedSantecDevice.InitializeAsync()
SimulatedSantecDevice.PingAsync()
SimulatedSantecDevice.Uninitialize()
SimulatedSantecDevice.ResetAsync()
SimulatedSantecDevice.RefreshAsync()
SimulatedSantecDevice.QueryAsync(String, CancellationToken)
SimulatedSantecDevice.WriteAsync(String, CancellationToken)
SimulatedSantecDevice.ReadAsync(CancellationToken)

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.Modular

Assembly: Santec.Hardware.dll

Syntax

```
public class SimulatedSLS : SimulatedXCS, ISimulatedDevice, IXCS, ISantecDevice, IModuleController,
IDevice<EXCSModel>, IDevice, IDisposable
```

Properties

Name

Declaration

```
public override string Name { get; }
```

Property Value

TYPE	DESCRIPTION
System.String	

Overrides

SimulatedXCS.Name

Implements

ISimulatedDevice
IXCS
ISantecDevice
IModuleController
IDevice<TModelType>
IDevice
System.IDisposable

Class SimulatedXCS

Provides a simulated XCS device with a single source module.

Inheritance

System.Object
SimulatedSantecDevice
SimulatedXCS
SimulatedSLS

Implements

ISimulatedDevice
IXCS
ISantecDevice
IModuleController
IDevice<EXCSModel>
IDevice
System.IDisposable

Inherited Members

SimulatedSantecDevice.creatingDevice
SimulatedSantecDevice.Dispose()
SimulatedSantecDevice.SerialNumber
SimulatedSantecDevice.FirmwareVersion
SimulatedSantecDevice.Address
SimulatedSantecDevice.IsInitialized
SimulatedSantecDevice.QueryResponses
SimulatedSantecDevice.ConfigureQueryResponses(IEnumerable<QueryResponsePair>)
SimulatedSantecDevice.GetTimeout()
SimulatedSantecDevice.SetTimeout(ETimeout)
SimulatedSantecDevice.GetInteractionModeAsync(CancellationToken)
SimulatedSantecDevice.GetIPAddressAsync(CancellationToken)
SimulatedSantecDevice.SetIPAddressAsync(IPAddress, CancellationToken)
SimulatedSantecDevice.EnableDHCPAsync(CancellationToken)
SimulatedSantecDevice.GetGatewayAsync(CancellationToken)
SimulatedSantecDevice.SetGatewayAsync(IPAddress, CancellationToken)
SimulatedSantecDevice.GetSubnetPrefixLengthAsync(CancellationToken)
SimulatedSantecDevice.SetSubnetPrefixLengthAsync(Int32, CancellationToken)
SimulatedSantecDevice.GetHostnameAsync(CancellationToken)
SimulatedSantecDevice.SetHostnameAsync(String, CancellationToken)
SimulatedSantecDevice.SetInteractionModeAsync(EInteractionMode, CancellationToken)
SimulatedSantecDevice.InitializeAsync()
SimulatedSantecDevice.PingAsync()
SimulatedSantecDevice.Uninitialize()
SimulatedSantecDevice.ResetAsync()
SimulatedSantecDevice.RefreshAsync()
SimulatedSantecDevice.QueryAsync(String, CancellationToken)
SimulatedSantecDevice.WriteAsync(String, CancellationToken)
SimulatedSantecDevice.ReadAsync(CancellationToken)
System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)

System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.Modular

Assembly: Santec.Hardware.dll

Syntax

```
public class SimulatedXCS : SimulatedSantecDevice, ISimulatedDevice, IXCS, ISantecDevice, IModuleController,
IDevice<EXCSModel>, IDevice, IDisposable
```

Properties

Model

Declaration

```
public EXCSModel Model { get; }
```

Property Value

TYPE	DESCRIPTION
EXCSModel	

Modules

Gets the modules in the device.

Declaration

```
public IReadOnlyList<IModule> Modules { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<IModule>	

Name

Declaration

```
public override string Name { get; }
```

Property Value

TYPE	DESCRIPTION
System.String	

Overrides

SimulatedSantecDevice.Name

NumberOfModules

Gets the number of modules in the device.

Declaration

```
public int NumberOfModules { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

Methods

AddModules(IEnumerable<IModule>)

Adds modules to the device.

Declaration

```
public void AddModules(IEnumerable<IModule> modules)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Collections.Generic.IEnumerable<IModule>	modules	

Implements

ISimulatedDevice
IXCS
ISantecDevice
IModuleController
IDevice<TModelType>
IDevice
System.IDisposable

Class XCS

Provides a connection to an XCS along with properties of the device and operations that can be performed using it.

> **NOTE**
>
> This class cannot not be used directly. It can only be used when returned:
>
> 1. As part of a list of devices by a DetectHardwareAsync(EConnectionTypesToDetect) or DetectHardwareAsync<T>(EConnectionTypesToDetect) call.
> 2. As a device by a TryDetectIPDeviceAsync(IPAddress) or TryDetectIPDeviceAsync<T>(IPAddress) call.

Inheritance

System.Object
PhysicalDevice
SantecDevice
XCS

Implements

IXCS
ISantecDevice
IModuleController
IDevice<EXCSModel>
IDevice
System.IDisposable

Inherited Members

SantecDevice.HandleIDNResponse(String[])
SantecDevice.GetIPAddressAsync(CancellationToken)
SantecDevice.SetIPAddressAsync(IPAddress, CancellationToken)
SantecDevice.EnableDHCPAsync(CancellationToken)
SantecDevice.GetGatewayAsync(CancellationToken)
SantecDevice.SetGatewayAsync(IPAddress, CancellationToken)
SantecDevice.GetSubnetPrefixLengthAsync(CancellationToken)
SantecDevice.SetSubnetPrefixLengthAsync(Int32, CancellationToken)
SantecDevice.GetHostnameAsync(CancellationToken)
SantecDevice.SetHostnameAsync(String, CancellationToken)
SantecDevice.GetInteractionModeAsync(CancellationToken)
SantecDevice.SetInteractionModeAsync(EInteractionMode, CancellationToken)
PhysicalDevice.Address
PhysicalDevice.SerialNumber
PhysicalDevice.FirmwareVersion
PhysicalDevice.IsInitialized
PhysicalDevice.IsInitializing
PhysicalDevice.SetTimeout(ETimeout)
PhysicalDevice.GetTimeout()
PhysicalDevice.Dispose()
PhysicalDevice.Dispose(Boolean)
PhysicalDevice.PingAsync()
PhysicalDevice.ResetAsync()
PhysicalDevice.QueryAsync(String, CancellationToken)
PhysicalDevice.WriteAsync(String, CancellationToken)
PhysicalDevice.ReadAsync(CancellationToken)

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.Modular

Assembly: Santec.Hardware.dll

Syntax

```
public class XCS : SantecDevice, IXCS, ISantecDevice, IModuleController, IDevice<EXCSModel>, IDevice,
IDisposable
```

Properties

Model

Declaration

```
public EXCSModel Model { get; }
```

Property Value

TYPE	DESCRIPTION
EXCSModel	

Modules

Declaration

```
public IReadOnlyList<IModule> Modules { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<IModule>	

Name

Declaration

```
public override string Name { get; }
```

Property Value

TYPE	DESCRIPTION
System.String	

Overrides

PhysicalDevice.Name

Remarks

NOTE

This property will always return "XCS".

NumberOfModules

Declaration

```
public int NumberOfModules { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

Methods

InitializeAsync()

Declaration

```
public override async Task InitializeAsync()
```

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Overrides

SantecDevice.InitializeAsync()

RefreshAsync()

Declaration

```
public override async Task RefreshAsync()
```

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Overrides

PhysicalDevice.RefreshAsync()

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized OSX.
UnexpectedDeviceResponseException	Thrown when the XCS modules response is not a valid module count.

TYPE	CONDITION
UnexpectedDeviceResponseException	Thrown when any XCS module information response is not a valid response for the corresponding module type.

Uninitialize()

Declaration

```
public override void Uninitialize()
```

Overrides

PhysicalDevice.Uninitialize()

Implements

IXCS
ISantecDevice
IModuleController
IDevice<TModelType>
IDevice
System.IDisposable

Namespace Santec.Hardware.Devices.Modular.Modules

Interfaces

IModule

Defines the properties of a module.

IModuleInformation

Represents a module information interface for modular devices.

Interface IModule

Defines the properties of a module.

Namespace: Santec.Hardware.Devices.Modular.Modules

Assembly: Santec.Hardware.dll

Syntax

```
public interface IModule
```

Properties

Index

Get the index of the module.

Declaration

```
int Index { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	The index of the module.

Interface IModuleInformation

Represents a module information interface for modular devices.

Namespace: Santec.Hardware.Devices.Modular.Modules

Assembly: Santec.Hardware.dll

Syntax

```
public interface IModuleInformation
```

Namespace Santec.Hardware.Devices.Modular.Modules.Attenuation

Classes

AttenuatorModule

Provides a connection to an attenuator module of a module controller along with properties of the module and operations that can be performed using it.

> **NOTE**
>
> This class cannot not be used directly. It can only be used when returned as a module of an IModuleController (ex. IOVA).

SimulatedAttenuatorModule

Provides a simulated attenuator module.

> **NOTE**
>
> This class is intended for testing/development purposes. It operates much quicker than an actual attenuator module. It can only be created as part of a simulated attenuator using a SimulatedDeviceFactory.

SimulatedAttenuatorModuleInformation

Represents the properties of a simulated attenuator module.

> **NOTE**
>
> This class is intended for testing/development purposes. It supports the creation of simulated attenuator modules for simulated attenuators.

Interfaces

IAttenuatorModule

Defines the properties and operations of an attenuator module that is part of a modular device.

Enums

EAttenuatorModuleType

Represents the types of attenuator modules available.

Class AttenuatorModule

Provides a connection to an attenuator module of a module controller along with properties of the module and operations that can be performed using it.

> **NOTE**
>
> This class cannot not be used directly. It can only be used when returned as a module of an IModuleController (ex. IOVA).

Inheritance

System.Object
AttenuatorModule

Implements

IAttenuatorModule
IModule
IFiberDevice
IDeviceWithWavelengthRange

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.Modular.Modules.Attenuation

Assembly: Santec.Hardware.dll

Syntax

```
public class AttenuatorModule : IAttenuatorModule, IModule, IFiberDevice, IDeviceWithWavelengthRange
```

Properties

Fiber

Declaration

```
public FiberInformation Fiber { get; }
```

Property Value

TYPE	DESCRIPTION
FiberInformation	

Index

Declaration

```
public int Index { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

MaximumAttenuation

Declaration

```
public double MaximumAttenuation { get; }
```

Property Value

TYPE	DESCRIPTION
System.Double	

MaximumWavelength

Declaration

```
public int MaximumWavelength { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

MinimumWavelength

Declaration

```
public int MinimumWavelength { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

Type

Declaration

```
public EAttenuatorModuleType Type { get; }
```

Property Value

TYPE	DESCRIPTION
EAttenuatorModuleType	

Methods

ChangeAttenuationAsync(Double, CancellationToken)

Declaration

```
public async Task ChangeAttenuationAsync(double attenuation, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Double	attenuation	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

GetAttenuationAsync(CancellationToken)

Declaration

```
public async Task<double> GetAttenuationAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

GetBeamBlockStateAsync(CancellationToken)

Declaration

```
public async Task<EBeamBlockState> GetBeamBlockStateAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<EBeamBlockState>	

GetWavelengthAsync(CancellationToken)

Declaration

```
public Task<int> GetWavelengthAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Int32>	

SetBeamBlockStateAsync(EBeamBlockState, CancellationToken)

Declaration

```
public async Task SetBeamBlockStateAsync(EBeamBlockState beamBlockState, CancellationToken cancellationToken
= default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
EBeamBlockState	beamBlockState	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetWavelengthAsync(Int32, CancellationToken)

Declaration

```
public Task SetWavelengthAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Implements

IAttenuatorModule
IModule
IFiberDevice
IDeviceWithWavelengthRange

Enum EAttenuatorModuleType

Represents the types of attenuator modules available.

Namespace: Santec.Hardware.Devices.Modular.Modules.Attenuation

Assembly: Santec.Hardware.dll

Syntax

```
public enum EAttenuatorModuleType
```

Fields

NAME	DESCRIPTION
MEMS	MEMS Attenuator Module Type
Optomechanical	Optomechanical Attenuator Module Type
Unknown	Default value indicating an unknown attenuator module type.

Interface IAttenuatorModule

Defines the properties and operations of an attenuator module that is part of a modular device.

Inherited Members

IModule.Index
IFiberDevice.Fiber
IDeviceWithWavelengthRange.MinimumWavelength
IDeviceWithWavelengthRange.MaximumWavelength
IDeviceWithWavelengthRange.SetWavelengthAsync(Int32, CancellationToken)
IDeviceWithWavelengthRange.GetWavelengthAsync(CancellationToken)

Namespace: Santec.Hardware.Devices.Modular.Modules.Attenuation

Assembly: Santec.Hardware.dll

Syntax

```
public interface IAttenuatorModule : IModule, IFiberDevice, IDeviceWithWavelengthRange
```

Properties

MaximumAttenuation

Get the maximum attenuation of the attenuator module.

Declaration

```
double MaximumAttenuation { get; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The maximum attenuation of the attenuator module.

Type

Gets the type of the attenuator module.

Declaration

```
EAttenuatorModuleType Type { get; }
```

Property Value

TYPE	DESCRIPTION
EAttenuatorModuleType	

Methods

ChangeAttenuationAsync(Double, CancellationToken)

Asynchronously change the attenuation of the attenuator module.

Declaration

```
Task ChangeAttenuationAsync(double attenuation, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Double	attenuation	The attenuation to change to.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the specified attenuation is out of range for the attenuator module.
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the attenuator module attenuation response does not match the expected response.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

GetAttenuationAsync(CancellationToken)

Asynchronously get the attenuation of the attenuator module.

Declaration

```
Task<double> GetAttenuationAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the attenuator module attenuation response is not a valid attenuation.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

GetBeamBlockStateAsync(CancellationToken)

Asynchronously get the state of the attenuator module's beam block.

Declaration

```
Task<EBeamBlockState> GetBeamBlockStateAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<EBeamBlockState>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the attenuator module beam block state response is not a valid state.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

SetBeamBlockStateAsync(EBeamBlockState, CancellationToken)

Asynchronously set the state of the attenuator module's beam block.

Declaration

```
Task SetBeamBlockStateAsync(EBeamBlockState beamBlockState, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
EBeamBlockState	beamBlockState	The state to set the attenuator module's beam block to.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device set beam block state response does not match the expected response.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

Class SimulatedAttenuatorModule

Provides a simulated attenuator module.

> **NOTE**
>
> This class is intended for testing/development purposes. It operates much quicker than an actual attenuator module. It can only be created as part of a simulated attenuator using a SimulatedDeviceFactory.

Inheritance

System.Object
SimulatedAttenuatorModule

Implements

IAttenuatorModule
IModule
IFiberDevice
IDeviceWithWavelengthRange

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.Modular.Modules.Attenuation

Assembly: Santec.Hardware.dll

Syntax

```
public class SimulatedAttenuatorModule : IAttenuatorModule, IModule, IFiberDevice,
IDeviceWithWavelengthRange
```

Properties

Fiber

Declaration

```
public FiberInformation Fiber { get; }
```

Property Value

TYPE	DESCRIPTION
FiberInformation	

Index

Declaration

```
public int Index { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

MaximumAttenuation

Declaration

```
public double MaximumAttenuation { get; }
```

Property Value

TYPE	DESCRIPTION
System.Double	

MaximumWavelength

Declaration

```
public int MaximumWavelength { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

MinimumWavelength

Declaration

```
public int MinimumWavelength { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

Type

Declaration

```
public EAttenuatorModuleType Type { get; }
```

Property Value

TYPE	DESCRIPTION
EAttenuatorModuleType	

Methods

ChangeAttenuationAsync(Double, CancellationToken)

Declaration

```
public async Task ChangeAttenuationAsync(double attenuation, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Double	attenuation	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

GetAttenuationAsync(CancellationToken)

Declaration

```
public async Task<double> GetAttenuationAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

GetBeamBlockStateAsync(CancellationToken)

Declaration

```
public async Task<EBeamBlockState> GetBeamBlockStateAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<EBeamBlockState>	

GetWavelengthAsync(CancellationToken)

Declaration

```
public async Task<int> GetWavelengthAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Int32>	

SetBeamBlockStateAsync(EBeamBlockState, CancellationToken)

Declaration

```
public async Task SetBeamBlockStateAsync(EBeamBlockState beamBlockState, CancellationToken cancellationToken
= default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
EBeamBlockState	beamBlockState	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetWavelengthAsync(Int32, CancellationToken)

Declaration

```
public async Task SetWavelengthAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Implements

IAttenuatorModule
IModule
IFiberDevice
IDeviceWithWavelengthRange

Class SimulatedAttenuatorModuleInformation

Represents the properties of a simulated attenuator module.

> **NOTE**
>
> This class is intended for testing/development purposes. It supports the creation of simulated attenuator modules for simulated attenuators.

Inheritance

System.Object
SimulatedAttenuatorModuleInformation

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.Modular.Modules.Attenuation

Assembly: Santec.Hardware.dll

Syntax

```
public class SimulatedAttenuatorModuleInformation
```

Constructors

SimulatedAttenuatorModuleInformation(Double, Int32, Int32, FiberInformation)

Initialize a new record of simulated attenuation module information.

Declaration

```
public SimulatedAttenuatorModuleInformation(double maximumAttenuation, int minimumWavelength, int
maximumWavelength, FiberInformation fiber)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Double	maximumAttenuation	The maximum attenuation of the simulated attenuator module.
System.Int32	minimumWavelength	The minimum wavelength of the simulated attenuator module.
System.Int32	maximumWavelength	The maximum wavelength of the simulated attenuator module.
FiberInformation	fiber	The fiber information of the simulated switch module.

Properties

Fiber

Get the fiber information of the simulated switch module.

Declaration

```
public FiberInformation Fiber { get; }
```

Property Value

TYPE	DESCRIPTION
FiberInformation	The fiber information of the simulated switch module.

MaximumAttenuation

Get the maximum attenuation of the simulated attenuator module.

Declaration

```
public double MaximumAttenuation { get; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The maximum attenuation of the simulated attenuator module.

MaximumWavelength

Get the maximum wavelength of the simulated attenuator module.

Declaration

```
public int MaximumWavelength { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	The maximum wavelength of the simulated attenuator module.

MinimumWavelength

Get the minimum wavelength of the simulated attenuator module.

Declaration

```
public int MinimumWavelength { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	The minimum wavelength of the simulated attenuator module.

Namespace Santec.Hardware.Devices.Modular.Modules.Polarization

Classes

PolarizationControllerModule

Provides a connection to a polarization controller module of a module controller along with properties of the module and operations that can be performed using it.

> **NOTE**
>
> This class cannot not be used directly. It can only be used when returned as a module of an IModuleController (ex. IXCS).

SimulatedPolarizationControllerModuleInformation

Simulated Polarization Controller Module Information

Interfaces

IPolarizationControllerModule

Defines the properties and operations of a polarization controller module that is part of a modular device.

Interface IPolarizationControllerModule

Defines the properties and operations of a polarization controller module that is part of a modular device.

Inherited Members

IModule.Index
IPolarizationController.GetPolarizationStateAsync(CancellationToken)
IPolarizationController.SetPolarizationStateAsync(EPolarizationState, CancellationToken)
IPolarizationController.GetWaveplatePositionAsync(Int32, CancellationToken)
IPolarizationController.SetWaveplatePositionAsync(Int32, Double, CancellationToken)

Namespace: Santec.Hardware.Devices.Modular.Modules.Polarization

Assembly: Santec.Hardware.dll

Syntax

```
public interface IPolarizationControllerModule : IModule, IPolarizationController
```

Class PolarizationControllerModule

Provides a connection to a polarization controller module of a module controller along with properties of the module and operations that can be performed using it.

> **NOTE**
>
> This class cannot not be used directly. It can only be used when returned as a module of an IModuleController (ex. IXCS).

Inheritance

System.Object
PolarizationControllerModule

Implements

IPolarizationControllerModule
IModule
IPolarizationController

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.Modular.Modules.Polarization

Assembly: Santec.Hardware.dll

Syntax

```
public class PolarizationControllerModule : IPolarizationControllerModule, IModule, IPolarizationController
```

Properties

Index

Declaration

```
public int Index { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

Methods

GetPolarizationStateAsync(CancellationToken)

Declaration

```
public Task<EPolarizationState> GetPolarizationStateAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<EPolarizationState>	

GetWaveplatePositionAsync(Int32, CancellationToken)

Declaration

```
public Task<double> GetWaveplatePositionAsync(int waveplateIndex, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	waveplateIndex	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

SetPolarizationStateAsync(EPolarizationState, CancellationToken)

Declaration

```
public Task SetPolarizationStateAsync(EPolarizationState polarizationState, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
EPolarizationState	polarizationState	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetWaveplatePositionAsync(Int32, Double, CancellationToken)

Declaration

```
public Task SetWaveplatePositionAsync(int waveplateIndex, double position, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	waveplateIndex	
System.Double	position	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Implements

IPolarizationControllerModule
IModule
IPolarizationController

Class SimulatedPolarizationControllerModuleInformation

Simulated Polarization Controller Module Information

Inheritance

System.Object
SimulatedPolarizationControllerModuleInformation

Implements

IModuleInformation

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.Modular.Modules.Polarization

Assembly: Santec.Hardware.dll

Syntax

```
public class SimulatedPolarizationControllerModuleInformation : IModuleInformation
```

Implements

IModuleInformation

Namespace Santec.Hardware.Devices.Modular.Modules.Power

Classes

OPMModule

Provides a connection to an OPM power meter module of a module controller along with properties of the module and operations that can be performed using it.

> **NOTE**
>
> This class cannot not be used directly. It can only be used when returned as a module of an IModuleController (ex. IOPM).

Interfaces

IOPMModule

Defines the properties and operations of an OPM power meter module that is part of a modular device.

IPowerMeterModule

Defines the properties and operations of a power meter module that is part of a modular device.

Enums

EPowerMeterModuleType

Represents the types of power meter modules available.

Enum EPowerMeterModuleType

Represents the types of power meter modules available.

Namespace: Santec.Hardware.Devices.Modular.Modules.Power

Assembly: Santec.Hardware.dll

Syntax

```
public enum EPowerMeterModuleType
```

Fields

NAME	DESCRIPTION
IPD	Inline Photodetector Module Type
OPM	Linear Optical Power Meter Module Type
Unknown	Default value indicating an unknown power meter module type.

Interface IOPMModule

Defines the properties and operations of an OPM power meter module that is part of a modular device.

Inherited Members

IPowerMeterModule.MinimumAveragingTime
IPowerMeterModule.MaximumAveragingTime
IPowerMeterModule.GainRanges
IPowerMeterModule.GetResolutionAsync(CancellationToken)
IPowerMeterModule.SetResolutionAsync(EResolution, CancellationToken)
IPowerMeterModule.GetAveragingTimeAsync(CancellationToken)
IPowerMeterModule.SetAveragingTimeAsync(Double, CancellationToken)
IPowerMeterModule.GetReadingModeAsync(CancellationToken)
IPowerMeterModule.SetReadingModeAsync(EReadingMode, CancellationToken)
IPowerMeterModule.GetGainRangeAsync(CancellationToken)
IPowerMeterModule.SetGainRangeAsync(GainRange, CancellationToken)
IPowerMeterModule.TurnOnAutoGainRangeModeAsync(CancellationToken)
IPowerMeterModule.GetAutoGainRangeModeStateAsync(CancellationToken)
IPowerMeterModule.ReadAsync(CancellationToken)
IPowerMeterModule.ReadAsync(Int32, CancellationToken)
IPowerMeterModule.ReadPowerAsync(CancellationToken)
IPowerMeterModule.ReadPowerAsync(Int32, CancellationToken)
IPowerMeterModule.ReadPowerInWattsAsync(CancellationToken)
IPowerMeterModule.ReadPowerInWattsAsync(Int32, CancellationToken)
IPowerMeterModule.GetILReferenceAsync(CancellationToken)
IPowerMeterModule.GetILReferenceAsync(Int32, CancellationToken)
IPowerMeterModule.SetILReferenceAsync(Double, CancellationToken)
IPowerMeterModule.SetILReferenceAsync(Int32, Double, CancellationToken)
IPowerMeterModule.ReferenceILAsync(CancellationToken)
IPowerMeterModule.ReferenceILAsync(Int32, CancellationToken)
IPowerMeterModule.ReadILAsync(CancellationToken)
IPowerMeterModule.ReadILAsync(Int32, CancellationToken)
IPowerMeterModule.DarkAsync(CancellationToken)
IPowerMeterModule.GetDarkStateAsync(CancellationToken)
IModule.Index
IDeviceWithWavelengthRange.MinimumWavelength
IDeviceWithWavelengthRange.MaximumWavelength
IDeviceWithWavelengthRange.SetWavelengthAsync(Int32, CancellationToken)
IDeviceWithWavelengthRange.GetWavelengthAsync(CancellationToken)

Namespace: Santec.Hardware.Devices.Modular.Modules.Power

Assembly: Santec.Hardware.dll

Syntax

```
public interface IOPMModule : IPowerMeterModule, IModule, IDeviceWithWavelengthRange
```

Properties

MaximumNumberOfLoggingPoints

Get the maximum number of logging points supported by the OPM module.

Declaration

```
int MaximumNumberOfLoggingPoints { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	The maximum number of logging points.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the property is accessed on an uninitialized module.

MinimumNumberOfLoggingPoints

Get the minimum number of logging points supported by the OPM module.

Declaration

```
int MinimumNumberOfLoggingPoints { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	The minimum number of logging points.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the property is accessed on an uninitialized module.

Type

Get the type of the OPM module.

Declaration

```
EPowerMeterModuleType Type { get; }
```

Property Value

TYPE	DESCRIPTION
EPowerMeterModuleType	

Methods

LogDetectorAsync(Int32, GainRange, Double, ETriggerEdge, ETriggerBehaviourMode, CancellationToken)

Asynchronously perform a logging operation using the OPM module.

Declaration

```
Task<IReadOnlyList<double>> LogDetectorAsync(int numberOfPoints, GainRange gainRange, double averagingTime, ETriggerEdge triggerEdge, ETriggerBehaviourMode triggerBehaviourMode, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	numberOfPoints	The number of points for the logging operation.
GainRange	gainRange	The gain range to use for the logging operation.
System.Double	averagingTime	The averaging time, in milliseconds, for the logging operation.
ETriggerEdge	triggerEdge	The trigger edge setting for the logging operation.
ETriggerBehaviourMode	triggerBehaviourMode	The trigger behaviour mode for the logging operation.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Collections.Generic.IReadOnlyList<System.Double>>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the specified number of logging points is out of range for the OPM module.
System.ArgumentException	Thrown when the OPM module does not support the specified gain range.

TYPE	CONDITION
System.ArgumentException	Thrown when the specified averaging time is out of range for the OPM module.
System.ArgumentException	Thrown when the specified trigger behaviour mode is ignore triggers.
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when a logging response does not match the expected response.
UnexpectedDeviceResponseException	Thrown when a logging response is not a valid number.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

LogDetectorAsync(Int32, Int32, GainRange, Double, ETriggerEdge, ETriggerBehaviourMode, CancellationToken)

Asynchronously perform a logging operation using the OPM module.

Declaration

```
Task<IReadOnlyList<double>> LogDetectorAsync(int numberOfPoints, int wavelength, GainRange gainRange, double
averagingTime, ETriggerEdge triggerEdge, ETriggerBehaviourMode triggerBehaviourMode, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	numberOfPoints	The number of points for the logging operation.
System.Int32	wavelength	The wavelength to perform the logging operation at.
GainRange	gainRange	The gain range to use for the logging operation.
System.Double	averagingTime	The averaging time, in milliseconds, for the logging operation.
ETriggerEdge	triggerEdge	The trigger edge setting for the logging operation.
ETriggerBehaviourMode	triggerBehaviourMode	The trigger behaviour mode for the logging operation.

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Collections.Generic.IReadOnlyList<System.Double>>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the specified number of logging points is out of range for the OPM module.
System.ArgumentException	Thrown when the specified wavelength is out of range for the OPM module.
System.ArgumentException	Thrown when the OPM module does not support the specified gain range.
System.ArgumentException	Thrown when the specified averaging time is out of range for the OPM module.
System.ArgumentException	Thrown when the specified trigger behaviour mode is ignore triggers.
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when a logging response does not match the expected response.
UnexpectedDeviceResponseException	Thrown when a logging response is not a valid number.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

SendSoftwareTriggerAsync(CancellationToken)

Asynchronously send a software trigger to the OPM module.

Declaration

```
Task SendSoftwareTriggerAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when a software trigger response does not match the expected response.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

Interface IPowerMeterModule

Defines the properties and operations of a power meter module that is part of a modular device.

Inherited Members

IModule.Index
IDeviceWithWavelengthRange.MinimumWavelength
IDeviceWithWavelengthRange.MaximumWavelength
IDeviceWithWavelengthRange.SetWavelengthAsync(Int32, CancellationToken)
IDeviceWithWavelengthRange.GetWavelengthAsync(CancellationToken)

Namespace: Santec.Hardware.Devices.Modular.Modules.Power

Assembly: Santec.Hardware.dll

Syntax

```
public interface IPowerMeterModule : IModule, IDeviceWithWavelengthRange
```

Properties

GainRanges

Get the gain ranges supported by the module.

Declaration

```
IReadOnlyList<GainRange> GainRanges { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<GainRange>	The gain ranges supported by the module.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the property is accessed on an uninitialized module.

MaximumAveragingTime

Get the maximum averaging time, in milliseconds, supported by the module.

Declaration

```
double MaximumAveragingTime { get; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The maximum averaging time in milliseconds.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the property is accessed on an uninitialized module.

MinimumAveragingTime

Get the minimum averaging time, in milliseconds, supported by the module.

Declaration

```
double MinimumAveragingTime { get; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The minimum averaging time in milliseconds.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the property is accessed on an uninitialized module.

Methods

DarkAsync(CancellationToken)

Asynchronously perform a user dark for the power meter module.

Declaration

```
Task DarkAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the power meter module dark response does not match the expected response.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

GetAutoGainRangeModeStateAsync(CancellationToken)

Asynchronously get whether auto gain range mode is turned on for the power meter module.

Declaration

```
Task<EAutoGainRangeModeState> GetAutoGainRangeModeStateAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<EAutoGainRangeModeState>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the power meter module auto gain range mode response is not a valid state.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

GetAveragingTimeAsync(CancellationToken)

Asynchronously get the averaging time, in milliseconds, of the power meter module.

Declaration

```
Task<double> GetAveragingTimeAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the power meter module averaging time response is not a valid averaging time.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

GetDarkStateAsync(CancellationToken)

Asynchronously get the state of the power meter module's user dark.

Declaration

```
Task<EDarkState> GetDarkStateAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<EDarkState>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the power meter module dark state response is not a valid state.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

GetGainRangeAsync(CancellationToken)

Asynchronously get the gain range of the power meter module.

Declaration

```
Task<GainRange> GetGainRangeAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<GainRange>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the power meter module gain range response does not correspond to a power meter gain range.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

GetILReferenceAsync(Int32, CancellationToken)

Asynchronously get the stored IL reference from the power meter module at the specified wavelength.

Declaration

```
Task<double> GetILReferenceAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	The wavelength to get the IL reference at.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the specified wavelength is out of range for the power meter module.
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the power meter module IL reference query response is not a valid number.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

GetILReferenceAsync(CancellationToken)

Asynchronously get the stored IL reference from the power meter module at the currently selected wavelength.

Declaration

```
Task<double> GetILReferenceAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the power meter module IL reference query response is not a valid number.

TYPE	CONDITION
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

GetReadingModeAsync(CancellationToken)

Asynchronously get the reading mode of the power meter module.

Declaration

```
Task<EReadingMode> GetReadingModeAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<EReadingMode>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the power meter module reading mode response does not correspond to a reading mode.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

GetResolutionAsync(CancellationToken)

Asynchronously get the reading resolution of the power meter module.

Declaration

```
Task<EResolution> GetResolutionAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<EResolution>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the power meter module reading resolution response does not correspond to a resolution.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

ReadAsync(Int32, CancellationToken)

Asynchronously read the power/power in watts/IL from the power meter module at the specified wavelength.

Declaration

```
Task<double> ReadAsync(int wavelength, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	The wavelength to read at.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the specified wavelength is out of range for the power meter module.
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the power meter module read response is not a valid number.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

ReadAsync(CancellationToken)

Asynchronously read the power/power in watts/IL from the power meter module at the currently selected wavelength.

Declaration

```
Task<double> ReadAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the power meter module read response is not a valid number.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

ReadILAsync(Int32, CancellationToken)

Asynchronously read the insertion loss from the power meter module at the specified wavelength.

Declaration

```
Task<double> ReadILAsync(int wavelength, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	The wavelength to read at.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the specified wavelength is out of range for the power meter module.
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.

TYPE	CONDITION
UnexpectedDeviceResponseException	Thrown when the power meter module read IL response is not a valid number.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

ReadILAsync(CancellationToken)

Asynchronously read the insertion loss from the power meter module at the currently selected wavelength.

Declaration

```
Task<double> ReadILAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the power meter module read IL response is not a valid number.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

ReadPowerAsync(Int32, CancellationToken)

Asynchronously read the power from the power meter module at the specified wavelength.

Declaration

```
Task<double> ReadPowerAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	The wavelength to read at.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the specified wavelength is out of range for the power meter module.
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the power meter module read power response is not a valid number.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

ReadPowerAsync(CancellationToken)

Asynchronously read the power from the power meter module at the currently selected wavelength.

Declaration

```
Task<double> ReadPowerAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the power meter module read power response is not a valid number.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

ReadPowerInWattsAsync(Int32, CancellationToken)

Asynchronously read the power in watts from the power meter module at the specified wavelength.

Declaration

```
Task<double> ReadPowerInWattsAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	The wavelength to read at.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the specified wavelength is out of range for the power meter module.
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the power meter module read power in watts response is not a valid number.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

ReadPowerInWattsAsync(CancellationToken)

Asynchronously read the power in watts from the power meter module at the currently selected wavelength.

Declaration

```
Task<double> ReadPowerInWattsAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the power meter module read power in watts response is not a valid number.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

ReferenceILAsync(Int32, CancellationToken)

Asynchronously perform an insertion loss reference for the power meter module at the specified wavelength.

Declaration

```
Task<double> ReferenceILAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	The wavelength to reference IL at.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the specified wavelength is out of range for the power meter module.
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.

TYPE	CONDITION
UnexpectedDeviceResponseException	Thrown when the power meter module reference IL response is not a valid number.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

ReferenceILAsync(CancellationToken)

Asynchronously perform an insertion loss reference for the power meter module at the currently selected wavelength.

Declaration

```
Task<double> ReferenceILAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the power meter module reference IL response is not a valid number.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

SetAveragingTimeAsync(Double, CancellationToken)

Asynchronously set the averaging time, in milliseconds, of the power meter module.

Declaration

```
Task SetAveragingTimeAsync(double averagingTime, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Double	averagingTime	The averaging time, in milliseconds, to set the module to.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the specified averaging time is out of range for the power meter module.
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the power meter module set averaging time response does not match the expected response.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

SetGainRangeAsync(GainRange, CancellationToken)

Asynchronously set the gain range of the power meter module.

Declaration

```
Task SetGainRangeAsync(GainRange gainRange, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
GainRange	gainRange	The gain range to set the module to.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the power meter module does not support the specified gain range.
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the power meter module set gain range response does not match the expected response.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

SetILReferenceAsync(Double, CancellationToken)

Asynchronously set the IL reference for the power meter module at the currently selected wavelength.

Declaration

```
Task SetILReferenceAsync(double reference, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Double	reference	The IL reference value.

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	The task representing the asynchronous operation.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the power meter module IL reference query response does not match the expected response.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

SetILReferenceAsync(Int32, Double, CancellationToken)

Asynchronously set the IL reference for the power meter module at the specified wavelength.

Declaration

```
Task SetILReferenceAsync(int wavelength, double reference, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	The wavelength to set the IL reference at.
System.Double	reference	The IL reference value.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION

TYPE	DESCRIPTION
System.Threading.Tasks.Task	The task representing the asynchronous operation.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the specified wavelength is out of range for the power meter module.
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the power meter module IL reference query response does not match the expected response.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

SetReadingModeAsync(EReadingMode, CancellationToken)

Asynchronously set the reading mode of the power meter module.

Declaration

```
Task SetReadingModeAsync(EReadingMode readingMode, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
EReadingMode	readingMode	The reading mode to set the module to.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the power meter module set reading mode response does not match the expected response.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

SetResolutionAsync(EResolution, CancellationToken)

Asynchronously set the reading resolution of the power meter module.

Declaration

```
Task SetResolutionAsync(EResolution resolution, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
EResolution	resolution	The resolution to set the module to.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the power meter set resolution response does not match the expected response.

TYPE	CONDITION
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

TurnOnAutoGainRangeModeAsync(CancellationToken)

Asynchronously turn on auto gain range mode for the power meter module.

Declaration

```
Task TurnOnAutoGainRangeModeAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the power meter module turn on auto gain range mode response does not match the expected response.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

Class OPMModule

Provides a connection to an OPM power meter module of a module controller along with properties of the module and operations that can be performed using it.

> **NOTE**
>
> This class cannot not be used directly. It can only be used when returned as a module of an IModuleController (ex. IOPM).

Inheritance

System.Object
OPMModule

Implements

IOPMModule
IPowerMeterModule
IModule
IDeviceWithWavelengthRange

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.Modular.Modules.Power

Assembly: Santec.Hardware.dll

Syntax

```
public class OPMModule : IOPMModule, IPowerMeterModule, IModule, IDeviceWithWavelengthRange
```

Properties

GainRanges

Declaration

```
public IReadOnlyList<GainRange> GainRanges { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<GainRange>	

Index

Declaration

```
public int Index { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

MaximumAveragingTime

Declaration

```
public double MaximumAveragingTime { get; }
```

Property Value

TYPE	DESCRIPTION
System.Double	

MaximumNumberOfLoggingPoints

Declaration

```
public int MaximumNumberOfLoggingPoints { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

MaximumWavelength

Declaration

```
public int MaximumWavelength { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

MinimumAveragingTime

Declaration

```
public double MinimumAveragingTime { get; }
```

Property Value

TYPE	DESCRIPTION
System.Double	

MinimumNumberOfLoggingPoints

Declaration

```
public int MinimumNumberOfLoggingPoints { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

MinimumWavelength

Declaration

```
public int MinimumWavelength { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

Type

Declaration

```
public EPowerMeterModuleType Type { get; }
```

Property Value

TYPE	DESCRIPTION
EPowerMeterModuleType	

Methods

DarkAsync(CancellationToken)

Declaration

```
public Task DarkAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

GetAutoGainRangeModeStateAsync(CancellationToken)

Declaration

```
public Task<EAutoGainRangeModeState> GetAutoGainRangeModeStateAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<EAutoGainRangeModeState>	

GetAveragingTimeAsync(CancellationToken)

Declaration

```
public Task<double> GetAveragingTimeAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

GetDarkStateAsync(CancellationToken)

Declaration

```
public Task<EDarkState> GetDarkStateAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<EDarkState>	

GetGainRangeAsync(CancellationToken)

Declaration

```
public Task<GainRange> GetGainRangeAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<GainRange>	

GetILReferenceAsync(Int32, CancellationToken)

Declaration

```
public Task<double> GetILReferenceAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

GetILReferenceAsync(CancellationToken)

Declaration

```
public Task<double> GetILReferenceAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

GetReadingModeAsync(CancellationToken)

Declaration

```
public Task<EReadingMode> GetReadingModeAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<EReadingMode>	

GetResolutionAsync(CancellationToken)

Declaration

```
public Task<EResolution> GetResolutionAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<EResolution>	

GetWavelengthAsync(CancellationToken)

Declaration

```
public Task<int> GetWavelengthAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Int32>	

LogDetectorAsync(Int32, GainRange, Double, ETriggerEdge, ETriggerBehaviourMode, CancellationToken)

Declaration

```
public Task<IReadOnlyList<double>> LogDetectorAsync(int numberOfPoints, GainRange gainRange, double
averagingTime, ETriggerEdge triggerEdge, ETriggerBehaviourMode triggerBehaviourMode, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	numberOfPoints	
GainRange	gainRange	

TYPE	NAME	DESCRIPTION
System.Double	averagingTime	
ETriggerEdge	triggerEdge	
ETriggerBehaviourMode	triggerBehaviourMode	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Collections.Generic.IReadOnlyList<System.Double>>	

LogDetectorAsync(Int32, Int32, GainRange, Double, ETriggerEdge, ETriggerBehaviourMode, CancellationToken)

Declaration

```
public Task<IReadOnlyList<double>> LogDetectorAsync(int numberOfPoints, int wavelength, GainRange gainRange,
double averagingTime, ETriggerEdge triggerEdge, ETriggerBehaviourMode triggerBehaviourMode,
CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	numberOfPoints	
System.Int32	wavelength	
GainRange	gainRange	
System.Double	averagingTime	
ETriggerEdge	triggerEdge	
ETriggerBehaviourMode	triggerBehaviourMode	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Collections.Generic.IReadOnlyList<System.Double>>	

ReadAsync(Int32, CancellationToken)

Declaration

```
public Task<double> ReadAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

ReadAsync(CancellationToken)

Declaration

```
public Task<double> ReadAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

ReadILAsync(Int32, CancellationToken)

Declaration

```
public Task<double> ReadILAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

ReadILAsync(CancellationToken)

Declaration

```
public Task<double> ReadILAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

ReadPowerAsync(Int32, CancellationToken)

Declaration

```
public Task<double> ReadPowerAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

ReadPowerAsync(CancellationToken)

Declaration

```
public Task<double> ReadPowerAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

ReadPowerInWattsAsync(Int32, CancellationToken)

Declaration

```
public Task<double> ReadPowerInWattsAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

ReadPowerInWattsAsync(CancellationToken)

Declaration

```
public Task<double> ReadPowerInWattsAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

ReferenceILAsync(Int32, CancellationToken)

Declaration

```
public Task<double> ReferenceILAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

ReferenceILAsync(CancellationToken)

Declaration

```
public Task<double> ReferenceILAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

SendSoftwareTriggerAsync(CancellationToken)

Declaration

```
public Task SendSoftwareTriggerAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetAveragingTimeAsync(Double, CancellationToken)

Declaration

```
public Task SetAveragingTimeAsync(double averagingTime, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Double	averagingTime	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetGainRangeAsync(GainRange, CancellationToken)

Declaration

```
public Task SetGainRangeAsync(GainRange gainRange, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
GainRange	gainRange	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetILReferenceAsync(Double, CancellationToken)

Declaration

```
public Task SetILReferenceAsync(double reference, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Double	reference	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetILReferenceAsync(Int32, Double, CancellationToken)

Declaration

```
public Task SetILReferenceAsync(int wavelength, double reference, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	
System.Double	reference	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetReadingModeAsync(EReadingMode, CancellationToken)

Declaration

```
public Task SetReadingModeAsync(EReadingMode readingMode, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
EReadingMode	readingMode	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetResolutionAsync(EResolution, CancellationToken)

Declaration

```
public Task SetResolutionAsync(EResolution resolution, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
EResolution	resolution	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetWavelengthAsync(Int32, CancellationToken)

Declaration

```
public Task SetWavelengthAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

TurnOnAutoGainRangeModeAsync(CancellationToken)

Declaration

```
public Task TurnOnAutoGainRangeModeAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Implements

IOPMModule
IPowerMeterModule
IModule
IDeviceWithWavelengthRange

Namespace Santec.Hardware.Devices.Modular.Modules.Source

Classes

DiscreteSourceSubModule

Provides a connection to a source submodule of a source module along with properties of the submodule and operations that can be performed using it.

> **NOTE**
>
> This class cannot not be used directly. It can only be used when returned as a source of an ISourceModule.

DiscreteSourceSubModuleInformation

Represents a discrete source submodule information for modular devices.

SimulatedDiscreteSourceSubModule

Provides a simulated discrete source sub module along with properties of the module and operations that can be performed using it.

SimulatedSourceModule

Simulated source module.

SimulatedSourceModuleInformation

Represents the information of a simulated source module.

SourceModule

Provides a connection to a source module of a module controller along with properties of the module and operations that can be performed using it.

> **NOTE**
>
> This class cannot not be used directly. It can only be used when returned as a module of an IModuleController (ex. IXCS).

SourceModuleWithCaseTemperature

Represents a source module with case temperature capabilities.

SourceSubModuleWithTemperatureControl

Represents a source submodule with temperature control capabilities, providing functionality to retrieve the current temperature.

Interfaces

IDiscreteSourceSubModule

Defines the properties and operations of a discrete source submodule that is part of a source module.

ISourceModule

Defines the properties and operations of a source module that is part of a modular device.

ISourceSubModule

Defines the properties and operations of a source submodule that is part of a source module.

ISourceSubModuleInformation

Represents a source submodule information interface for modular devices.

Enums

ESourceModuleType

Represents the type of a source submodule in a system.

Class DiscreteSourceSubModule

Provides a connection to a source submodule of a source module along with properties of the submodule and operations that can be performed using it.

> **NOTE**
>
> This class cannot not be used directly. It can only be used when returned as a source of an ISourceModule.

Inheritance

System.Object
DiscreteSourceSubModule

Implements

IDiscreteSourceSubModule
ISourceSubModule
IModule
IDiscreteSource
ISource
IFiberDevice

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.Modular.Modules.Source

Assembly: Santec.Hardware.dll

Syntax

```
public class DiscreteSourceSubModule : IDiscreteSourceSubModule, ISourceSubModule, IModule, IDiscreteSource,
ISource, IFiberDevice
```

Properties

Fiber

Declaration

```
public FiberInformation Fiber { get; }
```

Property Value

TYPE	DESCRIPTION
FiberInformation	

Index

Declaration

```
public int Index { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

Wavelength

Declaration

```
public int Wavelength { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

Methods

GetAbsolutePowerAsync(CancellationToken)

Declaration

```
public async Task<double> GetAbsolutePowerAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

GetPowerLevelAsync(CancellationToken)

Declaration

```
public async Task<double> GetPowerLevelAsync(CancellationToken cancellationToken)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

GetSourceStateAsync(CancellationToken)

Declaration

```
public async Task<ESourceState> GetSourceStateAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<ESourceState>	

SetAbsolutePowerAsync(Double, CancellationToken)

Declaration

```
public async Task SetAbsolutePowerAsync(double power, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Double	power	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetPowerLevelAsync(Double, CancellationToken)

Declaration

```
public async Task SetPowerLevelAsync(double power, CancellationToken cancellationToken)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Double	power	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

TurnOffSourceAsync(CancellationToken)

Declaration

```
public async Task TurnOffSourceAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

TurnOnSourceAsync(CancellationToken)

Declaration

```
public async Task TurnOnSourceAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Implements

IDiscreteSourceSubModule
ISourceSubModule
IModule
IDiscreteSource
ISource
IFiberDevice

Class DiscreteSourceSubModuleInformation

Represents a discrete source submodule information for modular devices.

Inheritance

System.Object
DiscreteSourceSubModuleInformation

Implements

ISourceSubModuleInformation
IFiberDevice

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.Modular.Modules.Source

Assembly: Santec.Hardware.dll

Syntax

```
public class DiscreteSourceSubModuleInformation : ISourceSubModuleInformation, IFiberDevice
```

Constructors

DiscreteSourceSubModuleInformation(Int32, FiberInformation)

Initializes a new instance of the DiscreteSourceSubModuleInformation class with the specified wavelength and fiber information.

Declaration

```
public DiscreteSourceSubModuleInformation(int wavelength, FiberInformation fiber)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	
FiberInformation	fiber	

Properties

Fiber

Gets or sets the fiber information for the source submodule.

Declaration

```
public FiberInformation Fiber { get; set; }
```

Property Value

TYPE	DESCRIPTION
FiberInformation	

Wavelength

Gets or sets the wavelength of the source submodule in nanometers (nm).

Declaration

```
public int Wavelength { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

Implements

ISourceSubModuleInformation
IFiberDevice

Enum ESourceModuleType

Represents the type of a source submodule in a system.

Namespace: Santec.Hardware.Devices.Modular.Modules.Source

Assembly: Santec.Hardware.dll

Syntax

```
public enum ESourceModuleType
```

Fields

NAME	DESCRIPTION
ELSFP	ELSFP submodule
HighPower	High power laser source submodule.
LowPower	Low power laser source submodule.
MediumPower	Lower power laser source submodule.
Unknown	Represents an unknown or unspecified value.

Interface IDiscreteSourceSubModule

Defines the properties and operations of a discrete source submodule that is part of a source module.

Inherited Members

ISourceSubModule.GetPowerLevelAsync(CancellationToken)
ISourceSubModule.SetPowerLevelAsync(Double, CancellationToken)
ISourceSubModule.GetAbsolutePowerAsync(CancellationToken)
ISourceSubModule.SetAbsolutePowerAsync(Double, CancellationToken)
IModule.Index
IDiscreteSource.Wavelength
ISource.GetSourceStateAsync(CancellationToken)
ISource.TurnOnSourceAsync(CancellationToken)
ISource.TurnOffSourceAsync(CancellationToken)
IFiberDevice.Fiber

Namespace: Santec.Hardware.Devices.Modular.Modules.Source

Assembly: Santec.Hardware.dll

Syntax

```
public interface IDiscreteSourceSubModule : ISourceSubModule, IModule, IDiscreteSource, ISource,
IFiberDevice
```

Interface ISourceModule

Defines the properties and operations of a source module that is part of a modular device.

Inherited Members

IModule.Index

Namespace: Santec.Hardware.Devices.Modular.Modules.Source

Assembly: Santec.Hardware.dll

Syntax

```
public interface ISourceModule : IModule
```

Properties

NumberOfSources

Get the number of sources in the module.

Declaration

```
int NumberOfSources { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	The number of sources in the module.

Sources

Get the sources of the device.

Declaration

```
IReadOnlyList<ISourceSubModule> Sources { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<ISourceSubModule>	The sources of the device.

Type

Gets the type of the submodule represented by this instance.

Declaration

```
ESourceModuleType Type { get; }
```

Property Value

TYPE	DESCRIPTION
ESourceModuleType	

Methods

TurnOffAllSourcesAsync(CancellationToken)

Disable all sub modules

Declaration

```
Task TurnOffAllSourcesAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

TurnOnAllSourcesAsync(CancellationToken)

Enable all sub modules

Declaration

```
Task TurnOnAllSourcesAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Interface ISourceSubModule

Defines the properties and operations of a source submodule that is part of a source module.

Inherited Members

IModule.Index
ISource.GetSourceStateAsync(CancellationToken)
ISource.TurnOnSourceAsync(CancellationToken)
ISource.TurnOffSourceAsync(CancellationToken)
IFiberDevice.Fiber

Namespace: Santec.Hardware.Devices.Modular.Modules.Source

Assembly: Santec.Hardware.dll

Syntax

```
public interface ISourceSubModule : IModule, ISource, IFiberDevice
```

Methods

GetAbsolutePowerAsync(CancellationToken)

Get the absolute power for the channel/submodule.

Declaration

```
Task<double> GetAbsolutePowerAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	Returns absolute power in dbm.

GetPowerLevelAsync(CancellationToken)

Get the power set point for current channel of source.

Declaration

```
Task<double> GetPowerLevelAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	Returns power in percentage

SetAbsolutePowerAsync(Double, CancellationToken)

Set the absolute power for the channel/submodule in dbm.

Declaration

```
Task SetAbsolutePowerAsync(double power, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Double	power	Set absolute power to be set in dbm
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetPowerLevelAsync(Double, CancellationToken)

Sets point for power output of channel of source between 0 to 100%.

Declaration

```
Task SetPowerLevelAsync(double power, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Double	power	Set point power output in percentage
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Interface ISourceSubModuleInformation

Represents a source submodule information interface for modular devices.

Inherited Members

IFiberDevice.Fiber

Namespace: Santec.Hardware.Devices.Modular.Modules.Source

Assembly: Santec.Hardware.dll

Syntax

```
public interface ISourceSubModuleInformation : IFiberDevice
```

Class SimulatedDiscreteSourceSubModule

Provides a simulated discrete source sub module along with properties of the module and operations that can be performed using it.

Inheritance

System.Object
SimulatedDiscreteSourceSubModule

Implements

IDiscreteSourceSubModule
ISourceSubModule
IModule
IDiscreteSource
ISource
IFiberDevice

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.Modular.Modules.Source

Assembly: Santec.Hardware.dll

Syntax

```
public class SimulatedDiscreteSourceSubModule : IDiscreteSourceSubModule, ISourceSubModule, IModule,
IDiscreteSource, ISource, IFiberDevice
```

Properties

Fiber

Declaration

```
public FiberInformation Fiber { get; }
```

Property Value

TYPE	DESCRIPTION
FiberInformation	

Index

Declaration

```
public int Index { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

Wavelength

Declaration

```
public int Wavelength { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

Methods

GetAbsolutePowerAsync(CancellationToken)

Declaration

```
public async Task<double> GetAbsolutePowerAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

GetPowerLevelAsync(CancellationToken)

Declaration

```
public async Task<double> GetPowerLevelAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

GetSourceStateAsync(CancellationToken)

Declaration

```
public async Task<ESourceState> GetSourceStateAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<ESourceState>	

SetAbsolutePowerAsync(Double, CancellationToken)

Declaration

```
public async Task SetAbsolutePowerAsync(double power, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Double	power	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetPowerLevelAsync(Double, CancellationToken)

Declaration

```
public async Task SetPowerLevelAsync(double power, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Double	power	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

TurnOffSourceAsync(CancellationToken)

Declaration

```
public async Task TurnOffSourceAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

TurnOnSourceAsync(CancellationToken)

Declaration

```
public async Task TurnOnSourceAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Implements

IDiscreteSourceSubModule
ISourceSubModule
IModule
IDiscreteSource
ISource
IFiberDevice

Class SimulatedSourceModule

Simulated source module.

Inheritance

System.Object
SimulatedSourceModule

Implements

ISourceModule
ISubModuleController
IModule

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.Modular.Modules.Source

Assembly: Santec.Hardware.dll

Syntax

```
public class SimulatedSourceModule : ISourceModule, ISubModuleController, IModule
```

Properties

Index

Declaration

```
public int Index { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

NumberOfSources

Declaration

```
public int NumberOfSources { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

NumberOfSubModules

Declaration

```
public int NumberOfSubModules { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

Sources

Declaration

```
public IReadOnlyList<ISourceSubModule> Sources { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<ISourceSubModule>	

SubModules

Declaration

```
public IReadOnlyList<IModule> SubModules { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<IModule>	

Type

Declaration

```
public ESourceModuleType Type { get; }
```

Property Value

TYPE	DESCRIPTION
ESourceModuleType	

Methods

AddSource(ISourceSubModule)

Add a source/submodule to the module.

Declaration

```
public void AddSource(ISourceSubModule source)
```

Parameters

TYPE	NAME	DESCRIPTION
ISourceSubModule	source	

TurnOffAllSourcesAsync(CancellationToken)

Turn off all sources/submodules.

Declaration

```
public async Task TurnOffAllSourcesAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

TurnOnAllSourcesAsync(CancellationToken)

Turn on all sources/submodules.

Declaration

```
public async Task TurnOnAllSourcesAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Implements

ISourceModule
ISubModuleController
IModule

Class SimulatedSourceModuleInformation

Represents the information of a simulated source module.

Inheritance

System.Object
SimulatedSourceModuleInformation

Implements

IModuleInformation

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.Modular.Modules.Source

Assembly: Santec.Hardware.dll

Syntax

```
public class SimulatedSourceModuleInformation : IModuleInformation
```

Constructors

SimulatedSourceModuleInformation()

Initializes a new instance of the SimulatedSourceModuleInformation class.

Declaration

```
public SimulatedSourceModuleInformation()
```

SimulatedSourceModuleInformation(List<ISourceSubModuleInformation>)

Initializes a new instance of the SimulatedSourceModuleInformation class with the specified sub-modules.

Declaration

```
public SimulatedSourceModuleInformation(List<ISourceSubModuleInformation> subModules)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Collections.Generic.List<ISourceSubModuleInformation>	subModules	

Properties

ModuleType

Gets or sets the type of the source module.

Declaration

```
public ESourceModuleType ModuleType { get; set; }
```

Property Value

TYPE	DESCRIPTION
ESourceModuleType	

SubModules

Gets or sets submodules

Declaration

```
public List<ISourceSubModuleInformation> SubModules { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.List<ISourceSubModuleInformation>	

Implements

IModuleInformation

Class SourceModule

Provides a connection to a source module of a module controller along with properties of the module and operations that can be performed using it.

> **NOTE**
>
> This class cannot not be used directly. It can only be used when returned as a module of an IModuleController (ex. IXCS).

Inheritance

System.Object
SourceModule

Implements

ISourceModule
ISubModuleController
IModule

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.Modular.Modules.Source

Assembly: Santec.Hardware.dll

Syntax

```
public class SourceModule : ISourceModule, ISubModuleController, IModule
```

Properties

Index

Declaration

```
public int Index { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

NumberOfSources

Declaration

```
public int NumberOfSources { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

NumberOfSubModules

Declaration

```
public int NumberOfSubModules { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

Sources

Declaration

```
public IReadOnlyList<ISourceSubModule> Sources { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<ISourceSubModule>	

SubModules

Declaration

```
public IReadOnlyList<IModule> SubModules { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<IModule>	

Type

Gets the type of submodule held by this source module

Declaration

```
public ESourceModuleType Type { get; }
```

Property Value

TYPE	DESCRIPTION
ESourceModuleType	

Methods

TurnOffAllSourcesAsync(CancellationToken)

Declaration

```
public async Task TurnOffAllSourcesAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

TurnOnAllSourcesAsync(CancellationToken)

Declaration

```
public async Task TurnOnAllSourcesAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Implements

ISourceModule
ISubModuleController
IModule

Class SourceModuleWithCaseTemperature

Represents a source module with case temperature capabilities.

Inheritance

System.Object
SourceModuleWithCaseTemperature

Implements

ISourceModule
IModule

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.Modular.Modules.Source

Assembly: Santec.Hardware.dll

Syntax

```
public class SourceModuleWithCaseTemperature : ISourceModule, IModule
```

Properties

Index

Declaration

```
public int Index { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

NumberOfSources

Declaration

```
public int NumberOfSources { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

NumberOfSubModules

Declaration

```
public int NumberOfSubModules { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

Sources

Declaration

```
public IReadOnlyList<ISourceSubModule> Sources { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<ISourceSubModule>	

SubModules

Declaration

```
public IReadOnlyList<IModule> SubModules { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<IModule>	

Type

Gets the type of submodule held by this source module

Declaration

```
public ESourceModuleType Type { get; }
```

Property Value

TYPE	DESCRIPTION
ESourceModuleType	

Methods

GetCaseTemperatureAsync(CancellationToken)

Asynchronously retrieves the current temperature of the device case in degrees Celsius.

Declaration

```
public async Task<double> GetCaseTemperatureAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	A token to monitor for cancellation requests. The operation will be canceled if the token is triggered.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	Current temperature of the device case in degrees Celsius.

TurnOffAllSourcesAsync(CancellationToken)

Declaration

```
public async Task TurnOffAllSourcesAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

TurnOnAllSourcesAsync(CancellationToken)

Declaration

```
public async Task TurnOnAllSourcesAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Implements

ISourceModule
IModule

Class SourceSubModuleWithTemperatureControl

Represents a source submodule with temperature control capabilities, providing functionality to retrieve the current temperature.

Inheritance

System.Object
SourceSubModuleWithTemperatureControl

Implements

IDiscreteSourceSubModule
ISourceSubModule
IModule
IDiscreteSource
ISource
IFiberDevice

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.Modular.Modules.Source

Assembly: Santec.Hardware.dll

Syntax

```
public class SourceSubModuleWithTemperatureControl : IDiscreteSourceSubModule, ISourceSubModule, IModule,
IDiscreteSource, ISource, IFiberDevice
```

Properties

Fiber

Declaration

```
public FiberInformation Fiber { get; }
```

Property Value

TYPE	DESCRIPTION
FiberInformation	

Index

Declaration

```
public int Index { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

Wavelength

Declaration

```
public int Wavelength { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

Methods

GetAbsolutePowerAsync(CancellationToken)

Declaration

```
public async Task<double> GetAbsolutePowerAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

GetPowerLevelAsync(CancellationToken)

Declaration

```
public async Task<double> GetPowerLevelAsync(CancellationToken cancellationToken)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

GetSourceStateAsync(CancellationToken)

Declaration

```
public async Task<ESourceState> GetSourceStateAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<ESourceState>	

GetTemperatureAsync(CancellationToken)

Asynchronously retrieves the current temperature in degrees Celsius.

Declaration

```
public async Task<double> GetTemperatureAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	A token to monitor for cancellation requests. The operation will be canceled if the token is triggered.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	Current temperature in degrees Celsius.

SetAbsolutePowerAsync(Double, CancellationToken)

Declaration

```
public async Task SetAbsolutePowerAsync(double power, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Double	power	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetPowerLevelAsync(Double, CancellationToken)

Declaration

```
public async Task SetPowerLevelAsync(double power, CancellationToken cancellationToken)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Double	power	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

TurnOffSourceAsync(CancellationToken)

Declaration

```
public async Task TurnOffSourceAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

TurnOnSourceAsync(CancellationToken)

Declaration

```
public async Task TurnOnSourceAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Implements

IDiscreteSourceSubModule
ISourceSubModule
IModule
IDiscreteSource

ISource
IFiberDevice

Namespace Santec.Hardware.Devices.Modular.Modules.Switching

Classes

SimulatedSwitchModule

Provides a simulated switch module.

> **NOTE**
>
> This class is intended for testing/development purposes. It operates much quicker than an actual switch module. It can only be created as part of a simulated switch using a SimulatedDeviceFactory.

SimulatedSwitchModuleInformation

Represents the properties of a simulated switch module.

> **NOTE**
>
> This class is intended for testing/development purposes. It supports the creation of simulated switch modules for simulated switches.

SwitchModule

Provides a connection to a switch module of a module controller along with properties of the module and operations that can be performed using it.

> **NOTE**
>
> This class cannot not be used directly. It can only be used when returned as a module of an IModuleController (ex. IOSX).

Interfaces

ISwitchModule

Defines the properties and operations of a switch module that is part of a modular device.

Enums

ESwitchModuleType

Represents the types of switch modules available.

Enum ESwitchModuleType

Represents the types of switch modules available.

Namespace: Santec.Hardware.Devices.Modular.Modules.Switching

Assembly: Santec.Hardware.dll

Syntax

```
public enum ESwitchModuleType
```

Fields

NAME	DESCRIPTION
MEMS	MEMS Switch Module Type
Optomechnical	Optomechnical Switch Module Type
Relay	Relay Switch Module Type
Unknown	Default value indicating an unknown switch module type.

Interface ISwitchModule

Defines the properties and operations of a switch module that is part of a modular device.

Inherited Members

IModule.Index

IFiberDevice.Fiber

Namespace: Santec.Hardware.Devices.Modular.Modules.Switching

Assembly: Santec.Hardware.dll

Syntax

```
public interface ISwitchModule : IModule, IFiberDevice
```

Properties

NumberOfChannels

Get the number of channels in the module.

Declaration

```
int NumberOfChannels { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	The number of channels in the module.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.

NumberOfCommons

Get the number of commons in the module.

Declaration

```
int NumberOfCommons { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	The number of commons in the module.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.

Type

Gets the type of the switch module.

Declaration

```
ESwitchModuleType Type { get; }
```

Property Value

TYPE	DESCRIPTION
ESwitchModuleType	

Methods

ChangeChannelAsync(Int32, CancellationToken)

Asynchronously change the channel of the module.

Declaration

```
Task ChangeChannelAsync(int channel, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	channel	The channel to switch to.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the specified channel is out of range for the module.
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the switch module channel response does not match the expected response.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

ChangeCommonAsync(Int32, CancellationToken)

Asynchronously change the common channel of the module.

Declaration

```
Task ChangeCommonAsync(int common, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	common	The common channel to switch to.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the specified common channel is out of range for the module.
System.InvalidOperationException	Thrown when the method is called on an uninitialized OSX.

TYPE	CONDITION
UnexpectedDeviceResponseException	Thrown when the OSX common channel response does not match the expected response.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the OSX.

GetChannelAsync(CancellationToken)

Asynchronously get the channel of the module.

Declaration

```
Task<int> GetChannelAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Int32>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the switch module channel response is not a valid channel.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

GetCommonAsync(CancellationToken)

Asynchronously get the common channel of the module.

Declaration

```
Task<int> GetCommonAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Int32>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the switch module common channel response is not a valid channel.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

Class SimulatedSwitchModule

Provides a simulated switch module.

> **NOTE**
>
> This class is intended for testing/development purposes. It operates much quicker than an actual switch module. It can only be created as part of a simulated switch using a SimulatedDeviceFactory.

Inheritance

System.Object
SimulatedSwitchModule

Implements

ISwitchModule
IModule
IFiberDevice

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.Modular.Modules.Switching

Assembly: Santec.Hardware.dll

Syntax

```
public class SimulatedSwitchModule : ISwitchModule, IModule, IFiberDevice
```

Properties

Fiber

Declaration

```
public FiberInformation Fiber { get; }
```

Property Value

TYPE	DESCRIPTION
FiberInformation	

Index

Declaration

```
public int Index { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

NumberOfChannels

Declaration

```
public int NumberOfChannels { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

NumberOfCommons

Declaration

```
public int NumberOfCommons { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

Type

Declaration

```
public ESwitchModuleType Type { get; }
```

Property Value

TYPE	DESCRIPTION
ESwitchModuleType	

Methods

ChangeChannelAsync(Int32, CancellationToken)

Declaration

```
public async Task ChangeChannelAsync(int channel, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	channel	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

ChangeCommonAsync(Int32, CancellationToken)

Declaration

```
public async Task ChangeCommonAsync(int common, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	common	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

GetChannelAsync(CancellationToken)

Declaration

```
public async Task<int> GetChannelAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Int32>	

GetCommonAsync(CancellationToken)

Declaration

```
public async Task<int> GetCommonAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Int32>	

Implements

ISwitchModule
IModule
IFiberDevice

Class SimulatedSwitchModuleInformation

Represents the properties of a simulated switch module.

> **NOTE**
>
> This class is intended for testing/development purposes. It supports the creation of simulated switch modules for simulated switches.

Inheritance

System.Object
SimulatedSwitchModuleInformation

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.Modular.Modules.Switching

Assembly: Santec.Hardware.dll

Syntax

```
public class SimulatedSwitchModuleInformation
```

Constructors

SimulatedSwitchModuleInformation(ESwitchModuleType, Int32, Int32, FiberInformation)

Declaration

```
public SimulatedSwitchModuleInformation(ESwitchModuleType moduleType, int numberOfCommons, int
numberOfChannels, FiberInformation fiber)
```

Parameters

TYPE	NAME	DESCRIPTION
ESwitchModuleType	moduleType	
System.Int32	numberOfCommons	
System.Int32	numberOfChannels	
FiberInformation	fiber	

SimulatedSwitchModuleInformation(Int32, Int32, FiberInformation)

Initialize a new record of simulated switch module information.

Declaration

```
public SimulatedSwitchModuleInformation(int numberOfCommons, int numberOfChannels, FiberInformation fiber)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	numberOfCommons	The number of commons in the simulated switch module.
System.Int32	numberOfChannels	The number of channels in the simulated switch module.
FiberInformation	fiber	The fiber information of the simulated switch module.

Properties

Fiber

Get the fiber information of the simulated switch module.

Declaration

```
public FiberInformation Fiber { get; }
```

Property Value

TYPE	DESCRIPTION
FiberInformation	The fiber information of the simulated switch module.

ModuleType

Gets or sets the type of the simulated switch module.

Declaration

```
public ESwitchModuleType ModuleType { get; set; }
```

Property Value

TYPE	DESCRIPTION
ESwitchModuleType	

NumberOfChannels

Get the number of channels in the simulated switch module.

Declaration

```
public int NumberOfChannels { get; }
```

Property Value

TYPE	DESCRIPTION

TYPE	DESCRIPTION
System.Int32	The number of channels in the simulated switch module.

NumberOfCommons

Get the number of commons in the simulated switch module.

Declaration

```
public int NumberOfCommons { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	The number of commons in the simulated switch module.

Class SwitchModule

Provides a connection to a switch module of a module controller along with properties of the module and operations that can be performed using it.

> **NOTE**
>
> This class cannot not be used directly. It can only be used when returned as a module of an IModuleController (ex. IOSX).

Inheritance

System.Object
SwitchModule

Implements

ISwitchModule
IModule
IFiberDevice

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.Modular.Modules.Switching

Assembly: Santec.Hardware.dll

Syntax

```
public class SwitchModule : ISwitchModule, IModule, IFiberDevice
```

Properties

Fiber

Declaration

```
public FiberInformation Fiber { get; }
```

Property Value

TYPE	DESCRIPTION
FiberInformation	

Index

Declaration

```
public int Index { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

NumberOfChannels

Declaration

```
public int NumberOfChannels { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

NumberOfCommons

Declaration

```
public int NumberOfCommons { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

Type

Declaration

```
public ESwitchModuleType Type { get; }
```

Property Value

TYPE	DESCRIPTION
ESwitchModuleType	

Methods

ChangeChannelAsync(Int32, CancellationToken)

Declaration

```
public async Task ChangeChannelAsync(int channel, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	channel	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

ChangeCommonAsync(Int32, CancellationToken)

Declaration

```
public async Task ChangeCommonAsync(int common, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	common	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

GetChannelAsync(CancellationToken)

Declaration

```
public async Task<int> GetChannelAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Int32>	

GetCommonAsync(CancellationToken)

Declaration

```
public async Task<int> GetCommonAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Int32>	

Implements

ISwitchModule
IModule
IFiberDevice

Namespace Santec.Hardware.Devices.Polarity

Classes

CableInformation

Represents cable information about a cable in a multi-cable polarity mapping operation.

CameraRecord

Represents an RD-P/RD-SP device camera.

> **NOTE**
>
> This class cannot be used directly. It can only be used when returned as part of a list of cameras by a GetCameras() call.

LegacyCustomCameraSettings

Represents custom camera settings, including gain and exposure, for an RD-P/RD-SP.

> **NOTE**
>
> This class cannot not be used directly. It can only be used when returned by an RD-P/RD-SP.

PTM

Provides a connection to an PTM along with properties of the device and operations that can be performed using it.

> **NOTE**
>
> This class cannot not be used directly. It can only be used when returned:
>
> 1. As part of a list of devices by a DetectHardwareAsync(EConnectionTypesToDetect) or DetectHardwareAsync<T>(EConnectionTypesToDetect) call.
> 2. As a device by a TryDetectIPDeviceAsync(IPAddress) or TryDetectIPDeviceAsync<T>(IPAddress) call.

RDP

Provides a connection to an RD-P along with properties of the device and operations that can be performed using it.

> **NOTE**
>
> This class cannot not be used directly. It can only be used when returned:
>
> 1. As a device by a CreateRDP(IRLM) call to a RDPFactory.

> **WARNING**
>
> An RD-P is a composite device and does not support all operations of a PTM/Santec Canada family device. Unsupported operations will throw a System.NotSupportedException.

RDPFactory

Provides a factory for creating RD-P devices.

RDSP

Provides a connection to an RD-SP along with properties of the device and operations that can be performed using it.

NOTE

This class cannot not be used directly. It can only be used when returned:

1. As a device by a CreateRDP(IRLM) call to a RDPFactory.

WARNING

An RD-SP is a composite device and does not support all operations of a PTM/Santec Canada family device. Unsupported operations will throw a System.NotSupportedException.

RDSPCameraValidator

Provides system-level validation for RD-SPs.

SimulatedPTM

Provides a simulated PTM with configurable polarity mappings.

NOTE

This class is intended for testing/development purposes. It operates much quicker than an actual PTM. It can only be created using a SimulatedDeviceFactory. It should only be used when returned as part of a list of devices by a DetectHardwareAsync(EConnectionTypesToDetect) or DetectHardwareAsync<T>(EConnectionTypesToDetect) call to a SimulatedHardwareDetectionService. A default detected A polarity mapping is set but the detected mapping can be configured via the ConfigureMapping(IEnumerable<Nullable<Int32>>) method. Additionally, the chance for the polarity mapping to fail and return a different detected mapping can also be configured with the ConfigureFailureChance(Int32) method. The default failure chance is 0%.

SimulatedRDP

Provides a simulated RD-P with configurable polarity mappings.

NOTE

This class is intended for testing/development purposes. It operates much quicker than an actual RD-P. It can only be created using a SimulatedRDPFactory. It should only be used when returned as part of a list of devices by a DetectHardwareAsync(EConnectionTypesToDetect) or DetectHardwareAsync<T>(EConnectionTypesToDetect) call to a SimulatedHardwareDetectionService. A default detected A polarity mapping is set but the detected mapping can be configured via the ConfigureMapping(IEnumerable<Nullable<Int32>>) method. Additionally, the chance for the polarity mapping to fail and return a different detected mapping can also be configured with the ConfigureFailureChance(Int32) method. The default failure chance is 0%.

SimulatedRDPFactory

Provides a factory for creating simulated RD-P devices.

NOTE

This class is intended for testing/development purposes.

SimulatedRDSP

Provides a simulated RD-P with configurable polarity mappings.

NOTE

This class is intended for testing/development purposes. It operates much quicker than an actual RD-P. It can only be created using a SimulatedRDPFactory. It should only be used when returned as part of a list of devices by a DetectHardwareAsync(EConnectionTypesToDetect) or DetectHardwareAsync<T>(EConnectionTypesToDetect) call to a SimulatedHardwareDetectionService. A default detected A polarity mapping is set but the detected mapping can be configured via the ConfigureMapping(IEnumerable<Nullable<Int32>>) method. Additionally, the chance for the polarity mapping to fail and return a different detected mapping can also be configured with the ConfigureFailureChance(Int32) method. The default failure chance is 0%.

Interfaces

IPTM

Defines the properties of a PTM and operations that can be performed with a connection to the device.

IPTM<TModelType>

Defines the properties of a PTM, including the model type, and operations that can be performed with a connection to the device.

IRDP

Defines the properties of an RD-P and operations that can be performed with a connection to the device.

IRDP<TModelType>

Defines the properties of an RD-P, including the model type, and operations that can be performed with a connection to the device.

IRDPFactory

Defines the operations of a factory for creating RD-P device connections.

IRDSP

Defines the properties of an RD-SP and operations that can be performed with a connection to the device.

Enums

ECameraMode

Specifies the camera settings used by the RD-P.

ELossCompensationConfiguration

Represents a loss compensation configuration to use with an RD-P camera when using an external source.

EMappingMode

Specifies the mapping mode of an RD-P.

This is used by an RD-P to determine whether to map detectors left to right (MPO) or right to left (Duplex).

EMappingSensitivity

Specifies the mapping sensitivity of an RD-P.

This corresponds to the strictness of the threshold used when differentiating rows/columns during the mapping data processing.

EPTMModel

Specifies the model of a PTM.

ERDPModel

Specifies the model of an RD-P.

ERDPRequirementFailureReason

Specifies the reasons for failure to meet the hardware requirements for an RD-P/RD-SP device.

ERDSPModel

Specifies the model of an RD-SP.

ERDSPSystemHardwareRequirementsFailureReason

Specifies the reasons for failure to meet the system-level hardware requirements for RD-SP devices.

Class CableInformation

Represents cable information about a cable in a multi-cable polarity mapping operation.

Inheritance

System.Object
CableInformation

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.Polarity

Assembly: Santec.Hardware.dll

Syntax

```
public class CableInformation
```

Constructors

CableInformation(Int32, Int32)

Initializes a new instance of the CableInformation class.

Declaration

```
public CableInformation(int numberOfFibers, int offset)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	numberOfFibers	The number of fibers in the cable.
System.Int32	offset	The mapping offset of the cable.

Properties

NumberOfFibers

Get the number of fibers in the cable.

Declaration

```
public int NumberOfFibers { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	The number of fibers in the cable.

Offset

Get the mapping offset of the cable.

> **NOTE**
>
> Offset + 1 will be used as the first input location of the cable when mapping the cable.

Declaration

```
public int Offset { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	The mapping offset of the cable.

Class CameraRecord

Represents an RD-P/RD-SP device camera.

> **NOTE**
>
> This class cannot be used directly. It can only be used when returned as part of a list of cameras by a GetCameras() call.

Inheritance

System.Object
CameraRecord

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.Polarity

Assembly: Santec.Hardware.dll

Syntax

```
public class CameraRecord
```

Properties

CameraName

Get the camera name.

Declaration

```
public string CameraName { get; }
```

Property Value

TYPE	DESCRIPTION
System.String	The name of the camera.

CameraSerialNumber

Get the camera serial number.

Declaration

```
public string CameraSerialNumber { get; }
```

Property Value

TYPE	DESCRIPTION
System.String	The camera serial number.

Enum ECameraMode

Specifies the camera settings used by the RD-P.

Namespace: Santec.Hardware.Devices.Polarity

Assembly: Santec.Hardware.dll

Syntax

```
public enum ECameraMode
```

Fields

NAME	DESCRIPTION
Custom	The camera is using custom settings.
LossCompensation	The camera is using settings according to the loss compensation value.

Enum ELossCompensationConfiguration

Represents a loss compensation configuration to use with an RD-P camera when using an external source.

Namespace: Santec.Hardware.Devices.Polarity

Assembly: Santec.Hardware.dll

Syntax

```
public enum ELossCompensationConfiguration
```

Fields

NAME	DESCRIPTION
PTM	Use the PTM loss compensation settings.
RLM	Use the RLM loss compensation settings.

Enum EMappingMode

Specifies the mapping mode of an RD-P.

This is used by an RD-P to determine whether to map detectors left to right (MPO) or right to left (Duplex).

Namespace: Santec.Hardware.Devices.Polarity

Assembly: Santec.Hardware.dll

Syntax

```
public enum EMappingMode
```

Fields

NAME	DESCRIPTION
Duplex	Duplex mapping mode. Looking at the end faces of the connectors, the first fiber is on the right.
MPO	MPO mapping mode. Looking at the end face of the MPO, the first fiber is on the left.

Enum EMappingSensitivity

Specifies the mapping sensitivity of an RD-P.

This corresponds to the strictness of the threshold used when differentiating rows/columns during the mapping data processing.

Namespace: Santec.Hardware.Devices.Polarity

Assembly: Santec.Hardware.dll

Syntax

```
public enum EMappingSensitivity
```

Fields

NAME	DESCRIPTION
BareFiber	Bare fiber sensitivity. Looser threshold for differentiating rows/columns.
Connector	Connectorized fiber sensitivity. Tighter threshold for differentiating rows/columns.

Enum EPTMModel

Specifies the model of a PTM.

Namespace: Santec.Hardware.Devices.Polarity

Assembly: Santec.Hardware.dll

Syntax

```
public enum EPTMModel
```

Fields

NAME	DESCRIPTION
PTM100	The PTM-100 model.

Enum ERDPModel

Specifies the model of an RD-P.

Namespace: Santec.Hardware.Devices.Polarity

Assembly: Santec.Hardware.dll

Syntax

```
public enum ERDPModel
```

Fields

NAME	DESCRIPTION
RDP100	The RD-P model.

Enum ERDPRequirementFailureReason

Specifies the reasons for failure to meet the hardware requirements for an RD-P/RD-SP device.

Namespace: Santec.Hardware.Devices.Polarity

Assembly: Santec.Hardware.dll

Syntax

```
public enum ERDPRequirementFailureReason
```

Fields

NAME	DESCRIPTION
CameraCountMismatchWithRDSPDetectors	Indicates that the number of cameras found does not match the number of RD-SP detectors.
InsufficientChannels	Indicates the device does not have sufficient channels for an RD-P/RD-SP.
MissingRequiredWavelength	Indicates the device does not support any of the required wavelengths for an RD-P/RD-SP.
MultipleCamerasNotSupported	Indicates that the device does not support multiple cameras, but multiple cameras were found.
RDSPDetectorsMissingCameras	Indicates that one or more RD-SP detectors are missing their corresponding cameras.
Unspecified	No reason specified.

Enum ERDSPModel

Specifies the model of an RD-SP.

Namespace: Santec.Hardware.Devices.Polarity

Assembly: Santec.Hardware.dll

Syntax

```
public enum ERDSPModel
```

Fields

NAME	DESCRIPTION
RDSP100	The RD-SP-100 model.

Enum ERDSPSystemHardwareRequirementsFailureReason

Specifies the reasons for failure to meet the system-level hardware requirements for RD-SP devices.

Namespace: Santec.Hardware.Devices.Polarity

Assembly: Santec.Hardware.dll

Syntax

```
public enum ERDSPSystemHardwareRequirementsFailureReason
```

Fields

NAME	DESCRIPTION
CamerasMissingRDSPDetectors	Indicates that one or more cameras are not associated with any RD-SP detector.
Unspecified	No reason specified.

Interface IPTM

Defines the properties of a PTM and operations that can be performed with a connection to the device.

Inherited Members

ISantecDevice.GetIPAddressAsync(CancellationToken)
ISantecDevice.SetIPAddressAsync(IPAddress, CancellationToken)
ISantecDevice.EnableDHCPAsync(CancellationToken)
ISantecDevice.GetGatewayAsync(CancellationToken)
ISantecDevice.SetGatewayAsync(IPAddress, CancellationToken)
ISantecDevice.GetSubnetPrefixLengthAsync(CancellationToken)
ISantecDevice.SetSubnetPrefixLengthAsync(Int32, CancellationToken)
ISantecDevice.GetHostnameAsync(CancellationToken)
ISantecDevice.SetHostnameAsync(String, CancellationToken)
ISantecDevice.GetInteractionModeAsync(CancellationToken)
ISantecDevice.SetInteractionModeAsync(EInteractionMode, CancellationToken)
IDevice.Name
IDevice.SerialNumber
IDevice.FirmwareVersion
IDevice.Address
IDevice.IsInitialized
IDevice.InitializeAsync()
IDevice.Uninitialize()
IDevice.GetTimeout()
IDevice.SetTimeout(ETimeout)
IDevice.ResetAsync()
IDevice.RefreshAsync()
IDevice.PingAsync()
IDevice.QueryAsync(String, CancellationToken)
IDevice.WriteAsync(String, CancellationToken)
IDevice.ReadAsync(CancellationToken)
System.IDisposable.Dispose()

Namespace: Santec.Hardware.Devices.Polarity

Assembly: Santec.Hardware.dll

Syntax

```
public interface IPTM : ISantecDevice, IDevice, IDisposable
```

Properties

NumberOfChannels

Get the number of channels on the device.

Declaration

```
int NumberOfChannels { get; }
```

Property Value

TYPE	DESCRIPTION

TYPE	DESCRIPTION
System.Int32	The number of channels on the device.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the property is accessed before the device has been initialized.

Methods

MapOutputAsync(Int32, CancellationToken)

Asynchronously map the detector of a specified output channel.

Declaration

```
Task<int?> MapOutputAsync(int channel, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	channel	The output channel to map the detector for.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Nullable<System.Int32>>	The task representing the asynchronous operation.

Remarks

The output channel cannot always be mapped to a detector by a device. This method will return `null` in these cases.

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.

TYPE	CONDITION
UnexpectedDeviceResponseException	Thrown when the device map detector response is not a valid response.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.
System.NotSupportedException	Thrown when the operation is not supported by the device.

MapPolarityAsync(Int32, CancellationToken)

Asynchronously map the polarity for a set of output channels.

Declaration

```
Task<IReadOnlyList<int?>> MapPolarityAsync(int numberOfChannels, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	numberOfChannels	The number of output channels to map.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Collections.Generic.IReadOnlyList<System.Nullable<System.Int32>>>	The task representing the asynchronous operation.

Remarks

Output channels cannot always be mapped to detectors by a device. This method will return a value of `null` for these output channels.

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the specified number of channels is out of range for the device.

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device map polarity response does not match the expected response format.
UnexpectedDeviceResponseException	Thrown when any of the mapping values are not valid response values.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.
PolarityMappingFailedException	Thrown when the device cannot determine a mapping of outputs to inputs.

TurnOffLaserAsync(CancellationToken)

Asynchronously turn off the active laser.

Declaration

```
Task TurnOffLaserAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.

TYPE	CONDITION
UnexpectedDeviceResponseException	Thrown when the device laser response does not match the expected response.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.
System.NotSupportedException	Thrown when the operation is not supported by the device.

TurnOnFlashingVFLModeAsync(Int32, CancellationToken)

Asynchronously enter flashing VFL mode and turn on the specified output laser.

Declaration

```
Task TurnOnFlashingVFLModeAsync(int channel, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	channel	The output channel to turn on the laser for.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the specified number of channels is out of range for the device.
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.

TYPE	CONDITION
UnexpectedDeviceResponseException	Thrown when the device flashing VFL response does not match the expected response.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.
System.NotSupportedException	Thrown when the operation is not supported by the device.

TurnOnVFLModeAsync(Int32, CancellationToken)

Asynchronously enter VFL mode and turn on the specified output laser.

Declaration

```
Task TurnOnVFLModeAsync(int channel, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	channel	The output channel to turn on the laser for.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the specified number of channels is out of range for the device.
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.

TYPE	CONDITION
UnexpectedDeviceResponseException	Thrown when the device VFL response does not match the expected response.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.
System.NotSupportedException	Thrown when the operation is not supported by the device.

Interface IPTM<TModelType>

Defines the properties of a PTM, including the model type, and operations that can be performed with a connection to the device.

Inherited Members

IPTM.NumberOfChannels
IPTM.MapPolarityAsync(Int32, CancellationToken)
IPTM.TurnOnVFLModeAsync(Int32, CancellationToken)
IPTM.TurnOnFlashingVFLModeAsync(Int32, CancellationToken)
IPTM.TurnOffLaserAsync(CancellationToken)
IPTM.MapOutputAsync(Int32, CancellationToken)
ISantecDevice.GetIPAddressAsync(CancellationToken)
ISantecDevice.SetIPAddressAsync(IPAddress, CancellationToken)
ISantecDevice.EnableDHCPAsync(CancellationToken)
ISantecDevice.GetGatewayAsync(CancellationToken)
ISantecDevice.SetGatewayAsync(IPAddress, CancellationToken)
ISantecDevice.GetSubnetPrefixLengthAsync(CancellationToken)
ISantecDevice.SetSubnetPrefixLengthAsync(Int32, CancellationToken)
ISantecDevice.GetHostnameAsync(CancellationToken)
ISantecDevice.SetHostnameAsync(String, CancellationToken)
ISantecDevice.GetInteractionModeAsync(CancellationToken)
ISantecDevice.SetInteractionModeAsync(EInteractionMode, CancellationToken)
IDevice<TModelType>.Model
IDevice.Name
IDevice.SerialNumber
IDevice.FirmwareVersion
IDevice.Address
IDevice.IsInitialized
IDevice.InitializeAsync()
IDevice.Uninitialize()
IDevice.GetTimeout()
IDevice.SetTimeout(ETimeout)
IDevice.ResetAsync()
IDevice.RefreshAsync()
IDevice.PingAsync()
IDevice.QueryAsync(String, CancellationToken)
IDevice.WriteAsync(String, CancellationToken)
IDevice.ReadAsync(CancellationToken)
System.IDisposable.Dispose()

Namespace: Santec.Hardware.Devices.Polarity

Assembly: Santec.Hardware.dll

Syntax

```
public interface IPTM<out TModelType> : IPTM, ISantecDevice, IDevice<TModelType>, IDevice, IDisposable where
TModelType : Enum
```

Type Parameters

NAME	DESCRIPTION
TModelType	

Interface IRDP

Defines the properties of an RD-P and operations that can be performed with a connection to the device.

Inherited Members

IPTM.NumberOfChannels
IPTM.MapPolarityAsync(Int32, CancellationToken)
IPTM.TurnOnVFLModeAsync(Int32, CancellationToken)
IPTM.TurnOnFlashingVFLModeAsync(Int32, CancellationToken)
IPTM.TurnOffLaserAsync(CancellationToken)
IPTM.MapOutputAsync(Int32, CancellationToken)
ISantecDevice.GetIPAddressAsync(CancellationToken)
ISantecDevice.SetIPAddressAsync(IPAddress, CancellationToken)
ISantecDevice.EnableDHCPAsync(CancellationToken)
ISantecDevice.GetGatewayAsync(CancellationToken)
ISantecDevice.SetGatewayAsync(IPAddress, CancellationToken)
ISantecDevice.GetSubnetPrefixLengthAsync(CancellationToken)
ISantecDevice.SetSubnetPrefixLengthAsync(Int32, CancellationToken)
ISantecDevice.GetHostnameAsync(CancellationToken)
ISantecDevice.SetHostnameAsync(String, CancellationToken)
ISantecDevice.GetInteractionModeAsync(CancellationToken)
ISantecDevice.SetInteractionModeAsync(EInteractionMode, CancellationToken)
ICompositeDevice.Subdevices
ICompositeDevice.Dispose(Boolean)
ICompositeDevice.Uninitialize(Boolean)
IDevice.Name
IDevice.SerialNumber
IDevice.FirmwareVersion
IDevice.Address
IDevice.IsInitialized
IDevice.InitializeAsync()
IDevice.Uninitialize()
IDevice.GetTimeout()
IDevice.SetTimeout(ETimeout)
IDevice.ResetAsync()
IDevice.RefreshAsync()
IDevice.PingAsync()
IDevice.QueryAsync(String, CancellationToken)
IDevice.WriteAsync(String, CancellationToken)
IDevice.ReadAsync(CancellationToken)
System.IDisposable.Dispose()

Namespace: Santec.Hardware.Devices.Polarity

Assembly: Santec.Hardware.dll

Syntax

```
public interface IRDP : IPTM, ISantecDevice, ICompositeDevice, IDevice, IDisposable
```

Properties

Wavelength

Gets the wavelength of the RD-P.

Declaration

```
int Wavelength { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	The RD-P wavelength.

Methods

CollectOutputMappingDataAsync(Int32, Int32, CancellationToken)

Asynchronously collect the output mapping data of a specified fiber for a specified cable during a multi-cable segmented polarity mapping operation.

Declaration

```
Task<bool> CollectOutputMappingDataAsync(int cable, int fiber, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	cable	The cable to collect the mapping data for.
System.Int32	fiber	The fiber of the cable to collect the mapping data for.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Boolean>	The task representing the asynchronous operation. Task result returns `true` if a fiber has been detected; otherwise, `false`.

Remarks

Task result returns `true` if a fiber has been detected; otherwise, `false`.

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the specified cable is out of range for the polarity mapping operation.
System.ArgumentException	Thrown when the specified fiber is out of range for the specified cable.
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
System.InvalidOperationException	Thrown when a polarity mapping operation is not in progress.
System.InvalidOperationException	Thrown when a segmented polarity mapping operation is not in progress.

CollectOutputMappingDataAsync(Int32, CancellationToken)

Asynchronously collect the output mapping data of a specified channel for a segmented polarity mapping operation.

Declaration

```
Task<bool> CollectOutputMappingDataAsync(int channel, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	channel	The output channel to collect the mapping data for.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Boolean>	The task representing the asynchronous operation. Task result returns `true` if a fiber has been detected; otherwise, `false`.

Remarks

Task result returns `true` if a fiber has been detected; otherwise, `false`.

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the specified channel is out of range for the polarity mapping operation.
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
System.InvalidOperationException	Thrown when a polarity mapping operation is not in progress.
System.InvalidOperationException	Thrown when a segmented polarity mapping operation is not in progress.

EndMappingPolarityAsync()

Asynchronously end a segmented polarity mapping operation.

Declaration

```
Task EndMappingPolarityAsync()
```

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
System.InvalidOperationException	Thrown when a polarity mapping operation is not in progress.
System.InvalidOperationException	Thrown when a segmented polarity mapping operation is not in progress.

GetCameraMode()

Get the camera mode of the RD-P.

Declaration

```
ECameraMode GetCameraMode()
```

Returns

TYPE	DESCRIPTION
ECameraMode	The current camera mode.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.

GetCustomCameraSettings()

Get the custom camera settings of the RD-P.

Declaration

```
LegacyCustomCameraSettings GetCustomCameraSettings()
```

Returns

TYPE	DESCRIPTION
LegacyCustomCameraSettings	The custom camera settings.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.

GetLossCompensation()

Get the loss compensation of the RD-P.

Declaration

```
int GetLossCompensation()
```

Returns

TYPE	DESCRIPTION
System.Int32	The loss compensation input power in dBm.

Exceptions

TYPE	CONDITION

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.

GetMappingInProgress()

Get whether a polarity mapping operation is in progress.

Declaration

```
bool GetMappingInProgress()
```

Returns

TYPE	DESCRIPTION
System.Boolean	`true` if a polarity mapping operation is in progress; otherwise, `false`.

GetMappingMode()

Get the mapping mode of the RD-P.

Declaration

```
EMappingMode GetMappingMode()
```

Returns

TYPE	DESCRIPTION
EMappingMode	The mapping mode of the RD-P.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.

GetMappingSensitivity()

Get the mapping sensitivity of the RD-P.

Declaration

```
EMappingSensitivity GetMappingSensitivity()
```

Returns

TYPE	DESCRIPTION
EMappingSensitivity	The mapping sensitivity of the RD-P.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.

GetOutputMappingDataCollected(Int32)

Get whether the output mapping data of a specified channel has been collected for a segmented polarity mapping operation.

Declaration

```
bool GetOutputMappingDataCollected(int channel)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	channel	The output channel for which to get whether the mapping data has been collected.

Returns

TYPE	DESCRIPTION
System.Boolean	`true` if the output mapping data has been collected for the channel; otherwise, `false`.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the specified channel is out of range for the polarity mapping operation.
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
System.InvalidOperationException	Thrown when a polarity mapping operation is not in progress.
System.InvalidOperationException	Thrown when a segmented polarity mapping operation is not in progress.

GetOutputMappingDataCollected(Int32, Int32)

Get whether the output mapping data of a specified fiber for a specified cable has been collected during a multi-cable segmented polarity mapping operation.

Declaration

```
bool GetOutputMappingDataCollected(int cable, int fiber)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	cable	The cable that the fiber is part of.
System.Int32	fiber	The fiber of the cable for which to get whether the mapping data has been collected.

Returns

TYPE	DESCRIPTION
System.Boolean	`true` if the output mapping data has been collected for the specificed fiber; otherwise, `false`.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the specified cable is out of range for the polarity mapping operation.
System.ArgumentException	Thrown when the specified fiber is out of range for the specified cable.
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
System.InvalidOperationException	Thrown when a polarity mapping operation is not in progress.
System.InvalidOperationException	Thrown when a segmented polarity mapping operation is not in progress.

GetOutputMappingImages(Int32)

Get the output mapping images for the specified channel for a segmented polarity mapping operation.

Declaration

```
IReadOnlyList<Mat> GetOutputMappingImages(int channel)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	channel	The output channel for which to get the mapping images.

Returns

TYPE	DESCRIPTION

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<OpenCvSharp.Mat>	A list of the mapping images for each detector for the specified channel.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the specified channel is out of range for the polarity mapping operation.
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
System.InvalidOperationException	Thrown when a polarity mapping operation is not in progress.
System.InvalidOperationException	Thrown when a segmented polarity mapping operation is not in progress.
System.InvalidOperationException	Thrown when mapping data has not been collected for the specified channel.

GetOutputMappingImages(Int32, Int32)

Get the output mapping images for the specified fiber of a specified cable for a segmented polarity mapping operation.

Declaration

```
IReadOnlyList<Mat> GetOutputMappingImages(int cable, int fiber)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	cable	The cable that the fiber is part of.
System.Int32	fiber	The fiber of the cable for which to get the mapping images.

Returns

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<OpenCvSharp.Mat>	A list of the mapping images for each detector for the specified fiber.

Exceptions

TYPE	CONDITION

TYPE	CONDITION
System.ArgumentException	Thrown when the specified cable is out of range for the polarity mapping operation.
System.ArgumentException	Thrown when the specified fiber is out of range for the specified cable.
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
System.InvalidOperationException	Thrown when a polarity mapping operation is not in progress.
System.InvalidOperationException	Thrown when a segmented polarity mapping operation is not in progress.
System.InvalidOperationException	Thrown when mapping data has not been collected for the specified fiber.

MapPolarityAsync(Int32, IEnumerable<Nullable<Int32>>, CancellationToken)

Asynchronously map the polarity for a set of output channels. This method uses an expected mapping to assist in analysis and provide results in scenarios it otherwise could not determine a mapping for.

Declaration

```
Task<IReadOnlyList<int?>> MapPolarityAsync(int numberOfChannels, IEnumerable<int?> expectedMapping,
CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	numberOfChannels	The number of output channels to map.
System.Collections.Generic.IEnumerable<System.Nullable<System.Int32>>	expectedMapping	The expected mapping result.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Collections.Generic.IReadOnlyList<System.Nullable<System.Int32>>>	The task representing the asynchronous operation.

Remarks

Output channels cannot always be mapped to detectors by a device. This method will return a value of `null` for these output channels.

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the specified number of channels is out of range for the device.
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device map polarity response does not match the expected response format.
UnexpectedDeviceResponseException	Thrown when any of the mapping values are not valid response values.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.
PolarityMappingFailedException	Thrown when the device cannot determine a mapping of outputs to inputs.

ProcessMappingData(IEnumerable<Nullable<Int32>>)

Process the collected output mapping data and generate a polarity mapping. This method can use an expected mapping to assist in analysis and provide results in scenarios it otherwise could not determine a mapping for.

Declaration

```
IReadOnlyList<int?> ProcessMappingData(IEnumerable<int?> expectedMapping = null)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Collections.Generic.IEnumerable<System.Nullable<System.Int32>>	expectedMapping	Optional: The expected mapping results.

Returns

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<System.Nullable<System.Int32>>	A read-only list of the polarity mappings for each output channel.

Remarks

Output channels cannot always be mapped to detectors by a device. This method will throw a PolarityMappingFailedException in this case.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
System.InvalidOperationException	Thrown when a polarity mapping operation is not in progress.
System.InvalidOperationException	Thrown when a segmented polarity mapping operation is not in progress.
System.InvalidOperationException	Thrown when data is missing for one or more outputs.
PolarityMappingFailedException	Thrown when the device cannot determine a mapping of outputs to inputs.

ProcessMappingData(Int32, IEnumerable<Nullable<Int32>>)

Process the collected output mapping data and generate a polarity mapping for a single cable of a multi-cable polarity mapping operation. This method can use an expected mapping to assist in analysis and provide results in scenarios it otherwise could not determine a mapping for.

Declaration

```
IReadOnlyList<int?> ProcessMappingData(int cable, IEnumerable<int?> expectedMapping = null)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	cable	The 1-based index of the cable to process the mapping data for.
System.Collections.Generic.IEnumerable<System.Nullable<System.Int32>>	expectedMapping	Optional: The expected mapping results.

Returns

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<System.Nullable<System.Int32>>	A read-only list of the polarity mappings for each output channel.

Remarks

Output channels cannot always be mapped to detectors by a device. This method will throw a PolarityMappingFailedException in

this case.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the specified cable index is out of range for the polarity mapping operation.
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
System.InvalidOperationException	Thrown when a polarity mapping operation is not in progress.
System.InvalidOperationException	Thrown when a segmented polarity mapping operation is not in progress.
System.InvalidOperationException	Thrown when a multi-cable polarity mapping operation is not in progress.
System.InvalidOperationException	Thrown when data is missing for one or more outputs.
PolarityMappingFailedException	Thrown when the device cannot determine a mapping of outputs to inputs.

SetCameraModeAsync(ECameraMode)

Asynchronously set the camera mode of the RD-P.

Declaration

```
Task SetCameraModeAsync(ECameraMode cameraMode)
```

Parameters

TYPE	NAME	DESCRIPTION
ECameraMode	cameraMode	The camera mode to set.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.

SetCustomCameraSettingsAsync(Int32, Int32)

Asynchronously set the custom camera settings of the RD-P.

Declaration

```
Task SetCustomCameraSettingsAsync(int gain, int exposure)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	gain	The camera gain.
System.Int32	exposure	The camera exposure.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

NOTE

The gain is represented as an integer value where the value is the gain in dB multiplied by 10 (ie. 100 => 10.0dB).

NOTE

The exposure is represented as an integer value where the value is the exponent for the equation: exposure = 2^x, in seconds (ie. -1 => 2^-1 = 0.5s)

NOTE

Setting the custom camera settings will automatically toggle the RD-P to custom settings camera mode.

Exceptions

TYPE	CONDITION

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
System.ArgumentException	Thrown when the specified gain is out of range for the device.
System.ArgumentException	Thrown when the specified exposure is out of range for the device.

SetLossCompensationAsync(Int32)

Asynchronously set the loss compensation of the RD-P.

> **NOTE**
>
> This value must be between 0dBm and -22dBm. The loss compensation adjusts the exposure and gain settings of the camera to compensate for the loss in the system. Loss compensation uses a binned approach. From 0dBm to -10dBm of input power, the standard camera settings are used. From -10dBm to -14dBm, the camera gain is increased. From -14dBm to -18dBm, both the camera gain and exposure settings are increased. Acquiring images will take longer. Lastly, from -18dBm to -22dBm, the exposure setting is increased. Acquiring images will take even longer.

Declaration

```
Task SetLossCompensationAsync(int inputPower)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	inputPower	The input power into the camera.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

> **NOTE**
>
> Setting the custom camera settings will automatically toggle the RD-P to custom settings camera mode.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
System.ArgumentException	Thrown when the specified loss compensation is out of range for the device.

SetMappingMode(EMappingMode)

Set the mapping mode of the RD-P.

Declaration

```
void SetMappingMode(EMappingMode mappingMode)
```

Parameters

TYPE	NAME	DESCRIPTION
EMappingMode	mappingMode	The mapping mode to change the RD-P to.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.

SetMappingSensitivity(EMappingSensitivity)

Set the mapping sensitivity of the RD-P.

Declaration

```
void SetMappingSensitivity(EMappingSensitivity mappingSensitivity)
```

Parameters

TYPE	NAME	DESCRIPTION
EMappingSensitivity	mappingSensitivity	The mapping sensitivity to change the RD-P to.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.

StartMappingPolarityAsync(IEnumerable<CableInformation>)

Asynchronously start a multi-cable segmented polarity mapping operation for a set of cables.

Declaration

```
Task StartMappingPolarityAsync(IEnumerable<CableInformation> cables)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Collections.Generic.IEnumerable<CableInformation>	cables	The cable information of the cables.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the specified number of fibers for a cable is out of range for the device.
System.ArgumentException	Thrown when the specified cable offset for a cable is less then 0.
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
System.InvalidOperationException	Thrown when a polarity mapping operation is already in progress.

StartMappingPolarityAsync(Int32)

Asynchronously start a segmented polarity mapping operation for a set of output channels.

Declaration

```
Task StartMappingPolarityAsync(int numberOfChannels)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	numberOfChannels	The number of output channels to map.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the specified number of channels is out of range for the device.
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
System.InvalidOperationException	Thrown when a polarity mapping operation is already in progress.

Interface IRDP<TModelType>

Defines the properties of an RD-P, including the model type, and operations that can be performed with a connection to the device.

Inherited Members

IRDP.Wavelength
IRDP.GetMappingMode()
IRDP.SetMappingMode(EMappingMode)
IRDP.GetMappingSensitivity()
IRDP.SetMappingSensitivity(EMappingSensitivity)
IRDP.GetCameraMode()
IRDP.SetCameraModeAsync(ECameraMode)
IRDP.GetCustomCameraSettings()
IRDP.SetCustomCameraSettingsAsync(Int32, Int32)
IRDP.GetLossCompensation()
IRDP.SetLossCompensationAsync(Int32)
IRDP.GetMappingInProgress()
IRDP.StartMappingPolarityAsync(Int32)
IRDP.StartMappingPolarityAsync(IEnumerable<CableInformation>)
IRDP.EndMappingPolarityAsync()
IRDP.GetOutputMappingDataCollected(Int32)
IRDP.GetOutputMappingDataCollected(Int32, Int32)
IRDP.GetOutputMappingImages(Int32)
IRDP.GetOutputMappingImages(Int32, Int32)
IRDP.CollectOutputMappingDataAsync(Int32, CancellationToken)
IRDP.CollectOutputMappingDataAsync(Int32, Int32, CancellationToken)
IRDP.ProcessMappingData(IEnumerable<Nullable<Int32>>)
IRDP.ProcessMappingData(Int32, IEnumerable<Nullable<Int32>>)
IRDP.MapPolarityAsync(Int32, IEnumerable<Nullable<Int32>>, CancellationToken)
ICompositeDevice.Subdevices
ICompositeDevice.Dispose(Boolean)
ICompositeDevice.Uninitialize(Boolean)
IPTM.NumberOfChannels
IPTM.MapPolarityAsync(Int32, CancellationToken)
IPTM.TurnOnVFLModeAsync(Int32, CancellationToken)
IPTM.TurnOnFlashingVFLModeAsync(Int32, CancellationToken)
IPTM.TurnOffLaserAsync(CancellationToken)
IPTM.MapOutputAsync(Int32, CancellationToken)
ISantecDevice.GetIPAddressAsync(CancellationToken)
ISantecDevice.SetIPAddressAsync(IPAddress, CancellationToken)
ISantecDevice.EnableDHCPAsync(CancellationToken)
ISantecDevice.GetGatewayAsync(CancellationToken)
ISantecDevice.SetGatewayAsync(IPAddress, CancellationToken)
ISantecDevice.GetSubnetPrefixLengthAsync(CancellationToken)
ISantecDevice.SetSubnetPrefixLengthAsync(Int32, CancellationToken)
ISantecDevice.GetHostnameAsync(CancellationToken)
ISantecDevice.SetHostnameAsync(String, CancellationToken)
ISantecDevice.GetInteractionModeAsync(CancellationToken)
ISantecDevice.SetInteractionModeAsync(EInteractionMode, CancellationToken)
IDevice<TModelType>.Model
IDevice.Name

IDevice.SerialNumber
IDevice.FirmwareVersion
IDevice.Address
IDevice.IsInitialized
IDevice.InitializeAsync()
IDevice.Uninitialize()
IDevice.GetTimeout()
IDevice.SetTimeout(ETimeout)
IDevice.ResetAsync()
IDevice.RefreshAsync()
IDevice.PingAsync()
IDevice.QueryAsync(String, CancellationToken)
IDevice.WriteAsync(String, CancellationToken)
IDevice.ReadAsync(CancellationToken)
System.IDisposable.Dispose()

Namespace: Santec.Hardware.Devices.Polarity

Assembly: Santec.Hardware.dll

Syntax

```
public interface IRDP<out TModelType> : IRDP, ICompositeDevice, IPTM<TModelType>, IPTM, ISantecDevice,
IDevice<TModelType>, IDevice, IDisposable where TModelType : Enum
```

Type Parameters

NAME	DESCRIPTION
TModelType	

Interface IRDPFactory

Defines the operations of a factory for creating RD-P device connections.

Namespace: Santec.Hardware.Devices.Polarity

Assembly: Santec.Hardware.dll

Syntax

```
public interface IRDPFactory
```

Methods

CheckForCamera()

Check for an RD-P camera on the system.

Declaration

```
bool CheckForCamera()
```

Returns

TYPE	DESCRIPTION
System.Boolean	`true` if an RD-P camera is found; otherwise, `false`.

CheckIfRLMMeetsHardwareRequirements(IRLM)

Check if an RLM meets the RD-P hardware requirements.

> **NOTE**
>
> An RLM must have an OSX connected to meet the hardware requirements.

> **NOTE**
>
> This check omits camera-related requirements.

Declaration

```
bool CheckIfRLMMeetsHardwareRequirements(IRLM rlm)
```

Parameters

TYPE	NAME	DESCRIPTION
IRLM	rlm	The RLM to check.

Returns

TYPE	DESCRIPTION

TYPE	DESCRIPTION
System.Boolean	`true` if the RLM meets the hardware requirements; otherwise, `false`.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the specified RLM is uninitialized.

CheckIfRLMMeetsHardwareRequirements(IRLM, IReadOnlyList<CameraRecord>)

Check if an RLM meets the RD-P hardware requirements.

> **NOTE**
>
> An RLM must have an OSX connected to meet the hardware requirements.

> **NOTE**
>
> This will also evaluate whether the RLM and attached cameras meet the requirements for RD-P/RD-SP creation with the CreateRDP(IRLM) method.

Declaration

```
bool CheckIfRLMMeetsHardwareRequirements(IRLM rlm, IReadOnlyList<CameraRecord> cameras)
```

Parameters

TYPE	NAME	DESCRIPTION
IRLM	rlm	The RLM to check.
System.Collections.Generic.IReadOnlyList<CameraRecord>	cameras	The RD-P/RD-SP cameras on the system.

Returns

TYPE	DESCRIPTION
System.Boolean	`true` if the RLM meets the hardware requirements; otherwise, `false`.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the specified RLM is uninitialized.

TYPE	CONDITION
System.ArgumentException	Thrown when the camera list is empty.

CheckIfRLMMeetsHardwareRequirementsWithDetails(IRLM)

Check if an RLM meets the RD-P hardware requirements.

> **NOTE**
>
> An RLM must have an OSX connected to meet the hardware requirements.

> **NOTE**
>
> This check omits camera-related requirements.

Declaration

```
HardwareRequirementsEvaluationResult<ERDPRequirementFailureReason>
CheckIfRLMMeetsHardwareRequirementsWithDetails(IRLM rlm)
```

Parameters

TYPE	NAME	DESCRIPTION
IRLM	rlm	The RLM to check.

Returns

TYPE	DESCRIPTION
HardwareRequirementsEvaluationResult<ERDPRequirementFailureReason>	`true` if the RLM meets the hardware requirements; otherwise, `false`.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the specified RLM is uninitialized.

CheckIfRLMMeetsHardwareRequirementsWithDetails(IRLM, IReadOnlyList<CameraRecord>)

Check if an RLM meets the RD-P hardware requirements.

> **NOTE**
>
> An RLM must have an OSX connected to meet the hardware requirements.

> **NOTE**
>
> This will also evaluate whether the RLM and attached cameras meet the requirements for RD-P/RD-SP creation with the CreateRDP(IRLM)

method.

Declaration

```
HardwareRequirementsEvaluationResult<ERDPRequirementFailureReason>
CheckIfRLMMeetsHardwareRequirementsWithDetails(IRLM rlm, IReadOnlyList<CameraRecord> cameras)
```

Parameters

TYPE	NAME	DESCRIPTION
IRLM	rlm	The RLM to check.
System.Collections.Generic.IReadOnlyList<CameraRecord>	cameras	The RD-P/RD-SP cameras on the system.

Returns

TYPE	DESCRIPTION
HardwareRequirementsEvaluationResult<ERDPRequirementFailureReason>	`true` if the RLM meets the hardware requirements; otherwise, `false`.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the specified RLM is uninitialized.
System.ArgumentException	Thrown when the camera list is empty.

CreateRDP(IPTM)

Creates a new RD-P with the specified PTM.

NOTE

PTM must be initialized and have a camera connected.

Declaration

```
IRDP CreateRDP(IPTM ptm)
```

Parameters

TYPE	NAME	DESCRIPTION
IPTM	ptm	The PTM component of the RD-P.

Returns

TYPE	DESCRIPTION
IRDP	The newly created RD-P.

Exceptions

TYPE	CONDITION
CameraNotFoundException	Thrown when the RD-P camera cannot be found on the system.
System.InvalidOperationException	Thrown when multiple RD-P cameras are found on the system.
System.InvalidOperationException	Thrown when the specified PTM is uninitialized.
System.ArgumentException	Thrown when the specified PTM is an RD-P.

CreateRDP(IPTM, IReadOnlyList<CameraRecord>)

Creates a new RD-P or RD-SP with the specified PTM and specified cameras.

> **NOTE**
>
> The PTM must be initialized.

Declaration

```
IRDP CreateRDP(IPTM ptm, IReadOnlyList<CameraRecord> cameras)
```

Parameters

TYPE	NAME	DESCRIPTION
IPTM	ptm	The PTM component of the RD-P.
System.Collections.Generic.IReadOnlyList<CameraRecord>	cameras	The cameras to use with the RD-P.

Returns

TYPE	DESCRIPTION
IRDP	The newly created RD-P.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the specified PTM is uninitialized.
System.ArgumentException	Thrown when the camera list is empty.
System.ArgumentException	Thrown when there are multiple cameras in the camera list.
System.ArgumentException	Thrown when the specified PTM is an RD-P.

CreateRDP(IRLM)

Creates a new RD-P with the specified RLM.

> **NOTE**
>
> The RLM must be initialized and have an OSX connected to create a RD-P.

Declaration

```
IRDP CreateRDP(IRLM rlm)
```

Parameters

TYPE	NAME	DESCRIPTION
IRLM	rlm	The RLM component of the RD-P.

Returns

TYPE	DESCRIPTION
IRDP	The newly created RD-P.

Exceptions

TYPE	CONDITION
CameraNotFoundException	Thrown when the RD-P camera cannot be found on the system.
System.InvalidOperationException	Thrown when multiple RD-P cameras are found on the system and the system does not support multiple cameras.

TYPE	CONDITION
System.InvalidOperationException	Thrown when the number of RD-SP cameras found on a system does not match the number of RD-SP detectors for an RD-SP.
System.InvalidOperationException	Thrown when there are multiple RD-SP cameras on the system but an RD-SP detector is missing its corresponding camera.
System.InvalidOperationException	Thrown when the specified RLM is uninitialized.
System.InvalidOperationException	Thrown when the specified RLM does not meet the RD-P hardware requirements.

CreateRDP(IRLM, IReadOnlyList<CameraRecord>)

Creates a new RD-P or RD-SP with the specified RLM and specified cameras.

> **NOTE**
>
> The RLM must be initialized and have an OSX connected to create a RD-P.

Declaration

```
IRDP CreateRDP(IRLM rlm, IReadOnlyList<CameraRecord> cameras)
```

Parameters

TYPE	NAME	DESCRIPTION
IRLM	rlm	The RLM component of the RD-P.
System.Collections.Generic.IReadOnlyList<CameraRecord>	cameras	The cameras to use with the RD-P.

Returns

TYPE	DESCRIPTION
IRDP	The newly created RD-P or RD-SP.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the camera list is empty.

TYPE	CONDITION
System.InvalidOperationException	Thrown when multiple RD-P cameras are found on the system and the system does not support multiple cameras.
System.InvalidOperationException	Thrown when the number of RD-SP cameras found on a system does not match the number of RD-SP detectors for an RD-SP.
System.InvalidOperationException	Thrown when there are multiple RD-SP cameras on the system but an RD-SP detector is missing its corresponding camera.
System.InvalidOperationException	Thrown when the specified RLM is uninitialized.
System.InvalidOperationException	Thrown when the specified RLM does not meet the RD-P hardware requirements.

CreateRDPWithExternalSource(Int32, Int32, ELossCompensationConfiguration)

Creates a new RD-P using an external source.

Declaration

```
IRDP CreateRDPWithExternalSource(int wavelength, int numberOfChannels, ELossCompensationConfiguration
lossCompensationConfiguration = ELossCompensationConfiguration.RLM)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	The wavelength of the external source.
System.Int32	numberOfChannels	The number of channels for the RD-P.
ELossCompensationConfiguration	lossCompensationConfiguration	The loss compensation configuration to use when in loss compensation mode.

Returns

TYPE	DESCRIPTION
IRDP	The newly created RD-P.

Exceptions

TYPE	CONDITION
CameraNotFoundException	Thrown when the RD-P camera cannot be found on the system.
System.InvalidOperationException	Thrown when multiple RD-P cameras are found on the system.

CreateRDPWithExternalSource(Int32, Int32, IReadOnlyList<CameraRecord>, ELossCompensationConfiguration)

Creates a new RD-P using an external source.

Declaration

```
IRDP CreateRDPWithExternalSource(int wavelength, int numberOfChannels, IReadOnlyList<CameraRecord> cameras,
ELossCompensationConfiguration lossCompensationConfiguration = ELossCompensationConfiguration.RLM)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	The wavelength of the external source.
System.Int32	numberOfChannels	The number of channels for the RD-P.
System.Collections.Generic.IReadOnlyList<CameraRecord>	cameras	The cameras to use with the RD-P.
ELossCompensationConfiguration	lossCompensationConfiguration	The loss compensation configuration to use when in loss compensation mode.

Returns

TYPE	DESCRIPTION
IRDP	The newly created RD-P or RD-SP.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the camera list is empty.
System.ArgumentException	Thrown when there are multiple cameras in the camera list.

GetCameras()

Get the available RD-P/RD-SP cameras on the system.

Declaration

```
IReadOnlyList<CameraRecord> GetCameras()
```

Returns

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<CameraRecord>	A list of the available RD-P/RD-SP cameras on the system.

Interface IRDSP

Defines the properties of an RD-SP and operations that can be performed with a connection to the device.

Inherited Members

IRDP.Wavelength
IRDP.GetMappingMode()
IRDP.SetMappingMode(EMappingMode)
IRDP.GetMappingSensitivity()
IRDP.SetMappingSensitivity(EMappingSensitivity)
IRDP.GetCameraMode()
IRDP.SetCameraModeAsync(ECameraMode)
IRDP.GetCustomCameraSettings()
IRDP.SetCustomCameraSettingsAsync(Int32, Int32)
IRDP.GetLossCompensation()
IRDP.SetLossCompensationAsync(Int32)
IRDP.GetMappingInProgress()
IRDP.StartMappingPolarityAsync(Int32)
IRDP.StartMappingPolarityAsync(IEnumerable<CableInformation>)
IRDP.EndMappingPolarityAsync()
IRDP.GetOutputMappingDataCollected(Int32)
IRDP.GetOutputMappingDataCollected(Int32, Int32)
IRDP.GetOutputMappingImages(Int32)
IRDP.GetOutputMappingImages(Int32, Int32)
IRDP.CollectOutputMappingDataAsync(Int32, CancellationToken)
IRDP.CollectOutputMappingDataAsync(Int32, Int32, CancellationToken)
IRDP.ProcessMappingData(IEnumerable<Nullable<Int32>>)
IRDP.ProcessMappingData(Int32, IEnumerable<Nullable<Int32>>)
IRDP.MapPolarityAsync(Int32, IEnumerable<Nullable<Int32>>, CancellationToken)
ICompositeDevice.Subdevices
ICompositeDevice.Dispose(Boolean)
ICompositeDevice.Uninitialize(Boolean)
IPTM.NumberOfChannels
IPTM.MapPolarityAsync(Int32, CancellationToken)
IPTM.TurnOnVFLModeAsync(Int32, CancellationToken)
IPTM.TurnOnFlashingVFLModeAsync(Int32, CancellationToken)
IPTM.TurnOffLaserAsync(CancellationToken)
IPTM.MapOutputAsync(Int32, CancellationToken)
IDevice<ERDSPModel>.Model
IRLM.Detectors
IRLM.SupportsILResolution
IRLM.SupportsMaximumResolutionRL
IRLM.SupportedRLModes
IRLM.SupportedLengthModes
IRLM.GetRLModeAsync(CancellationToken)
IRLM.SetRLModeAsync(ERLMode, CancellationToken)
IRLM.GetPositionBModeAsync(CancellationToken)
IRLM.SetPositionBModeAsync(EPositionBMode, CancellationToken)
IRLM.GetGainModeAsync(CancellationToken)
IRLM.SetGainModeAsync(EGainMode, CancellationToken)
IRLM.GetLengthModeAsync(CancellationToken)
IRLM.SetLengthModeAsync(ELengthMode, CancellationToken)

IRLM.GetDUTILAsync(Int32, CancellationToken)
IRLM.SetDUTILAsync(Int32, Double, CancellationToken)
IRLM.GetRLReferenceAsync(CancellationToken)
IRLM.SetRLReferenceAsync(Double, CancellationToken)
IRLM.GetILResolutionAsync(CancellationToken)
IRLM.SetILResolutionAsync(EResolution, CancellationToken)
IRLM.ReferenceRLAsync(CancellationToken)
IRLM.ReadRLAsync(Int32, CancellationToken)
IRLM.ReadRLAsync(Int32, Double, CancellationToken)
IRLM.ReadRLAsync(Int32, Double, Double, CancellationToken)
IRLM.GetTraceDiagnosticsAsync(CancellationToken)
ISantecDevice.GetIPAddressAsync(CancellationToken)
ISantecDevice.SetIPAddressAsync(IPAddress, CancellationToken)
ISantecDevice.EnableDHCPAsync(CancellationToken)
ISantecDevice.GetGatewayAsync(CancellationToken)
ISantecDevice.SetGatewayAsync(IPAddress, CancellationToken)
ISantecDevice.GetSubnetPrefixLengthAsync(CancellationToken)
ISantecDevice.SetSubnetPrefixLengthAsync(Int32, CancellationToken)
ISantecDevice.GetHostnameAsync(CancellationToken)
ISantecDevice.SetHostnameAsync(String, CancellationToken)
ISantecDevice.GetInteractionModeAsync(CancellationToken)
ISantecDevice.SetInteractionModeAsync(EInteractionMode, CancellationToken)
IDiscretePowerMeter.ReadMultiplePowersAsync(IEnumerable<Int32>, Int32, CancellationToken)
IDiscretePowerMeter.ReadMultipleILsAsync(IEnumerable<Int32>, Int32, CancellationToken)
ILaserSourceWithMonitor.ReadMonitorPowerAsync(Int32, CancellationToken)
ILaserSourceWithMonitor.GetFactoryPowerAsync(Int32, Int32, CancellationToken)
ILaserSource.NumberOfOutputs
ILaserSource.SupportsTurningOffLaser
ILaserSource.GetActiveLaserAsync(CancellationToken)
ILaserSource.TurnOnLaserAsync(Int32, CancellationToken)
ILaserSource.GetOutputChannelAsync(CancellationToken)
ILaserSource.ChangeOutputChannelAsync(Int32, CancellationToken)
IWavelengthDevice.Wavelengths
ISwitchController.Switches
ISwitchController.GetSwitchChannelAsync(Int32, CancellationToken)
ISwitchController.ChangeSwitchChannelAsync(Int32, Int32, CancellationToken)
ISwitchController.ChangeMultipleSwitchesChannelsAsync(IEnumerable<DeviceSwitchAndChannelPair>, CancellationToken)
IFiberDevice.Fiber
IPowerMeterWithReferencePersistence.RequiresILReferenceSave
IPowerMeterWithReferencePersistence.SaveILReferencesAsync(CancellationToken)
IPowerMeter.ReadPowerAsync(Int32, Int32, CancellationToken)
IInsertionLossMeter.GetILReferenceAsync(Int32, Int32, CancellationToken)
IInsertionLossMeter.SetILReferenceAsync(Int32, Int32, Double, CancellationToken)
IInsertionLossMeter.ReferenceILAsync(Int32, Int32, CancellationToken)
IInsertionLossMeter.ReferenceILAsync(Int32, Int32, Boolean, CancellationToken)
IInsertionLossMeter.ReadILAsync(Int32, Int32, CancellationToken)
IDetectorDevice.NumberOfDetectors
IDevice.Name
IDevice.SerialNumber
IDevice.FirmwareVersion
IDevice.Address
IDevice.IsInitialized

IDevice.InitializeAsync()
IDevice.Uninitialize()
IDevice.GetTimeout()
IDevice.SetTimeout(ETimeout)
IDevice.ResetAsync()
IDevice.RefreshAsync()
IDevice.PingAsync()
IDevice.QueryAsync(String, CancellationToken)
IDevice.WriteAsync(String, CancellationToken)
IDevice.ReadAsync(CancellationToken)
System.IDisposable.Dispose()

Namespace: Santec.Hardware.Devices.Polarity

Assembly: Santec.Hardware.dll

Syntax

```
public interface IRDSP : IRDP<ERDSPModel>, IRDP, ICompositeDevice, IPTM<ERDSPModel>, IPTM,
IDevice<ERDSPModel>, IRLM, ISantecDevice, IDiscretePowerMeter, ILaserSourceWithMonitor, ILaserSource,
IWavelengthDevice, ISwitchController, IFiberDevice, IPowerMeterWithReferencePersistence, IPowerMeter,
IInsertionLossMeter, IDetectorDevice, IDevice<ERLMModel>, IDevice, IDisposable
```

Extension Methods

RLMExtensions.IsMultimodeRLM100(IRLM)

Class LegacyCustomCameraSettings

Represents custom camera settings, including gain and exposure, for an RD-P/RD-SP.

> **NOTE**
>
> This class cannot not be used directly. It can only be used when returned by an RD-P/RD-SP.

Inheritance

System.Object
LegacyCustomCameraSettings

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.Polarity

Assembly: Santec.Hardware.dll

Syntax

```
public class LegacyCustomCameraSettings
```

Properties

Exposure

The camera exposure.

> **NOTE**
>
> The exposure is represented as an integer value where the value is an exponent for the equation: exposure = 2^x, in seconds (ie. -1 => 2^-1 = 0.5s)

Declaration

```
public int Exposure { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	The camera exposure.

Gain

The camera gain.

> **NOTE**

The gain is represented as an integer value where the value is the gain in dB multiplied by 10 (ie. 100 => 10.0dB).

Declaration

```
public int Gain { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	The camera gain.

Class PTM

Provides a connection to an PTM along with properties of the device and operations that can be performed using it.

> **NOTE**
>
> This class cannot not be used directly. It can only be used when returned:
>
> 1. As part of a list of devices by a DetectHardwareAsync(EConnectionTypesToDetect) or DetectHardwareAsync<T>(EConnectionTypesToDetect) call.
> 2. As a device by a TryDetectIPDeviceAsync(IPAddress) or TryDetectIPDeviceAsync<T>(IPAddress) call.

Inheritance

System.Object
PhysicalDevice
SantecDevice
PTM

Implements

IPTM<EPTMModel>
IPTM
ISantecDevice
IDevice<EPTMModel>
IDevice
System.IDisposable

Inherited Members

SantecDevice.HandleIDNResponse(String[])
SantecDevice.GetIPAddressAsync(CancellationToken)
SantecDevice.SetIPAddressAsync(IPAddress, CancellationToken)
SantecDevice.EnableDHCPAsync(CancellationToken)
SantecDevice.GetGatewayAsync(CancellationToken)
SantecDevice.SetGatewayAsync(IPAddress, CancellationToken)
SantecDevice.GetSubnetPrefixLengthAsync(CancellationToken)
SantecDevice.SetSubnetPrefixLengthAsync(Int32, CancellationToken)
SantecDevice.GetHostnameAsync(CancellationToken)
SantecDevice.SetHostnameAsync(String, CancellationToken)
SantecDevice.GetInteractionModeAsync(CancellationToken)
SantecDevice.SetInteractionModeAsync(EInteractionMode, CancellationToken)
PhysicalDevice.Address
PhysicalDevice.SerialNumber
PhysicalDevice.FirmwareVersion
PhysicalDevice.IsInitialized
PhysicalDevice.IsInitializing
PhysicalDevice.Uninitialize()
PhysicalDevice.SetTimeout(ETimeout)
PhysicalDevice.GetTimeout()
PhysicalDevice.Dispose()
PhysicalDevice.Dispose(Boolean)
PhysicalDevice.PingAsync()
PhysicalDevice.ResetAsync()
PhysicalDevice.QueryAsync(String, CancellationToken)
PhysicalDevice.WriteAsync(String, CancellationToken)

PhysicalDevice.ReadAsync(CancellationToken)
System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.Polarity

Assembly: Santec.Hardware.dll

Syntax

```
public class PTM : SantecDevice, IPTM<EPTMModel>, IPTM, ISantecDevice, IDevice<EPTMModel>, IDevice,
IDisposable
```

Properties

Model

Declaration

```
public EPTMModel Model { get; }
```

Property Value

TYPE	DESCRIPTION
EPTMModel	

Name

Declaration

```
public override string Name { get; }
```

Property Value

TYPE	DESCRIPTION
System.String	

Overrides

PhysicalDevice.Name

Remarks

> **NOTE**
>
> This property will always return "PTM".

NumberOfChannels

Declaration

```
public int NumberOfChannels { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

Methods

InitializeAsync()

Declaration

```
public override async Task InitializeAsync()
```

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Overrides

SantecDevice.InitializeAsync()

Exceptions

TYPE	CONDITION
UnexpectedDeviceResponseException	Thrown when the PTM channels response is not a valid number.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the PTM.

MapOutputAsync(Int32, CancellationToken)

Declaration

```
public async Task<int?> MapOutputAsync(int channel, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	channel	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Nullable<System.Int32>>	

MapPolarityAsync(Int32, CancellationToken)

Declaration

```
public async Task<IReadOnlyList<int?>> MapPolarityAsync(int numberOfChannels, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	numberOfChannels	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Collections.Generic.IReadOnlyList<System.Nullable<System.Int32>>>	

RefreshAsync()

Declaration

```
public override async Task RefreshAsync()
```

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Overrides

PhysicalDevice.RefreshAsync()

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized PTM.
UnexpectedDeviceResponseException	Thrown when the PTM channels response is not a valid number.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the PTM.

TurnOffLaserAsync(CancellationToken)

Declaration

```
public async Task TurnOffLaserAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

TurnOnFlashingVFLModeAsync(Int32, CancellationToken)

Declaration

```
public async Task TurnOnFlashingVFLModeAsync(int channel, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	channel	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

TurnOnVFLModeAsync(Int32, CancellationToken)

Declaration

```
public async Task TurnOnVFLModeAsync(int channel, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	channel	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Implements

IPTM<TModelType>
IPTM
ISantecDevice
IDevice<TModelType>
IDevice
System.IDisposable

Class RDP

Provides a connection to an RD-P along with properties of the device and operations that can be performed using it.

> **NOTE**
>
> This class cannot not be used directly. It can only be used when returned:
>
> 1. As a device by a CreateRDP(IRLM) call to a RDPFactory.

> **WARNING**
>
> An RD-P is a composite device and does not support all operations of a PTM/Santec Canada family device. Unsupported operations will throw a System.NotSupportedException.

Inheritance

System.Object
CompositeSantecDevice
RDP

Implements

IRDP<ERDPModel>
IRDP
ICompositeDevice
IPTM<ERDPModel>
IPTM
ISantecDevice
IDevice<ERDPModel>
IDevice
System.IDisposable

Inherited Members

CompositeSantecDevice.SerialNumber
CompositeSantecDevice.FirmwareVersion
CompositeSantecDevice.Address
CompositeSantecDevice.IsInitialized
CompositeSantecDevice.IsInitializing
CompositeSantecDevice.Subdevices
CompositeSantecDevice.Dispose()
CompositeSantecDevice.Dispose(Boolean)
CompositeSantecDevice.SetTimeout(ETimeout)
CompositeSantecDevice.GetTimeout()
CompositeSantecDevice.EnableDHCPAsync(CancellationToken)
CompositeSantecDevice.GetGatewayAsync(CancellationToken)
CompositeSantecDevice.GetHostnameAsync(CancellationToken)
CompositeSantecDevice.GetInteractionModeAsync(CancellationToken)
CompositeSantecDevice.GetIPAddressAsync(CancellationToken)
CompositeSantecDevice.GetSubnetPrefixLengthAsync(CancellationToken)
CompositeSantecDevice.PingAsync()
CompositeSantecDevice.ResetAsync()
CompositeSantecDevice.SetGatewayAsync(IPAddress, CancellationToken)
CompositeSantecDevice.SetHostnameAsync(String, CancellationToken)
CompositeSantecDevice.SetInteractionModeAsync(EInteractionMode, CancellationToken)

CompositeSantecDevice.SetIPAddressAsync(IPAddress, CancellationToken)
CompositeSantecDevice.SetSubnetPrefixLengthAsync(Int32, CancellationToken)
CompositeSantecDevice.QueryAsync(String, CancellationToken)
CompositeSantecDevice.WriteAsync(String, CancellationToken)
CompositeSantecDevice.ReadAsync(CancellationToken)
CompositeSantecDevice.Uninitialize()
System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.Polarity

Assembly: Santec.Hardware.dll

Syntax

```
public class RDP : CompositeSantecDevice, IRDP<ERDPModel>, IRDP, ICompositeDevice, IPTM<ERDPModel>, IPTM,
ISantecDevice, IDevice<ERDPModel>, IDevice, IDisposable
```

Properties

Model

Declaration

```
public ERDPModel Model { get; }
```

Property Value

TYPE	DESCRIPTION
ERDPModel	

Name

Declaration

```
public override string Name { get; }
```

Property Value

TYPE	DESCRIPTION
System.String	

Overrides

CompositeSantecDevice.Name

Remarks

> **NOTE**
>
> This property will always return "RD-P".

NumberOfChannels

Declaration

```
public int NumberOfChannels { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

Wavelength

Declaration

```
public int Wavelength { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

Methods

CollectOutputMappingDataAsync(Int32, Int32, CancellationToken)

Declaration

```
public Task<bool> CollectOutputMappingDataAsync(int cable, int fiber, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	cable	
System.Int32	fiber	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Boolean>	

CollectOutputMappingDataAsync(Int32, CancellationToken)

Declaration

```
public Task<bool> CollectOutputMappingDataAsync(int channel, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	channel	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Boolean>	

Dispose(Boolean, Boolean)

Declaration

```
protected override void Dispose(bool disposing, bool disposeSubdevices)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Boolean	disposing	
System.Boolean	disposeSubdevices	

Overrides

CompositeSantecDevice.Dispose(Boolean, Boolean)

EndMappingPolarityAsync()

Declaration

```
public Task EndMappingPolarityAsync()
```

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

GetCameraMode()

Declaration

```
public ECameraMode GetCameraMode()
```

Returns

TYPE	DESCRIPTION
ECameraMode	

GetCustomCameraSettings()

Declaration

```
public LegacyCustomCameraSettings GetCustomCameraSettings()
```

Returns

TYPE	DESCRIPTION
LegacyCustomCameraSettings	

GetLossCompensation()

Declaration

```
public int GetLossCompensation()
```

Returns

TYPE	DESCRIPTION
System.Int32	

GetMappingInProgress()

Declaration

```
public bool GetMappingInProgress()
```

Returns

TYPE	DESCRIPTION
System.Boolean	

GetMappingMode()

Declaration

```
public EMappingMode GetMappingMode()
```

Returns

TYPE	DESCRIPTION
EMappingMode	

GetMappingSensitivity()

Declaration

```
public EMappingSensitivity GetMappingSensitivity()
```

Returns

TYPE	DESCRIPTION
EMappingSensitivity	

GetOutputMappingDataCollected(Int32)

Declaration

```
public bool GetOutputMappingDataCollected(int channel)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	channel	

Returns

TYPE	DESCRIPTION
System.Boolean	

GetOutputMappingDataCollected(Int32, Int32)

Declaration

```
public bool GetOutputMappingDataCollected(int cable, int fiber)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	cable	
System.Int32	fiber	

Returns

TYPE	DESCRIPTION
System.Boolean	

GetOutputMappingImages(Int32)

Declaration

```
public IReadOnlyList<Mat> GetOutputMappingImages(int channel)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	channel	

Returns

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<OpenCvSharp.Mat>	

GetOutputMappingImages(Int32, Int32)

Declaration

```
public IReadOnlyList<Mat> GetOutputMappingImages(int cable, int fiber)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	cable	
System.Int32	fiber	

Returns

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<OpenCvSharp.Mat>	

InitializeAsync()

Declaration

```
public override async Task InitializeAsync()
```

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Overrides

CompositeSantecDevice.InitializeAsync()

Exceptions

TYPE	CONDITION
CompositeDeviceHardwareException	Thrown when the component RLM no longer meets the RD-P hardware requirements.

MapOutputAsync(Int32, CancellationToken)

⚠ CAUTION

This operation is not supported for RD-P devices. This operation will always throw a System.NotSupportedException.

Declaration

```
public Task<int?> MapOutputAsync(int channel, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	channel	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Nullable<System.Int32>>	

MapPolarityAsync(Int32, IEnumerable<Nullable<Int32>>, CancellationToken)

Declaration

```
public Task<IReadOnlyList<int?>> MapPolarityAsync(int numberOfChannels, IEnumerable<int?> expectedMapping,
CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	numberOfChannels	
System.Collections.Generic.IEnumerable<System.Nullable<System.Int32>>	expectedMapping	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Collections.Generic.IReadOnlyList<System.Nullable<System.Int32>>>	

MapPolarityAsync(Int32, CancellationToken)

Declaration

```
public Task<IReadOnlyList<int?>> MapPolarityAsync(int numberOfChannels, CancellationToken cancellationToken
= default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	numberOfChannels	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Collections.Generic.IReadOnlyList<System.Nullable<System.Int32>>>	

ProcessMappingData(IEnumerable<Nullable<Int32>>)

Declaration

```
public IReadOnlyList<int?> ProcessMappingData(IEnumerable<int?> expectedMapping = null)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Collections.Generic.IEnumerable<System.Nullable<System.Int32>>	expectedMapping	

Returns

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<System.Nullable<System.Int32>>	

ProcessMappingData(Int32, IEnumerable<Nullable<Int32>>)

Declaration

```
public IReadOnlyList<int?> ProcessMappingData(int cableIndex, IEnumerable<int?> expectedMapping = null)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	cableIndex	
System.Collections.Generic.IEnumerable<System.Nullable<System.Int32>>	expectedMapping	

Returns

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<System.Nullable<System.Int32>>	

RefreshAsync()

Declaration

```
public override async Task RefreshAsync()
```

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Overrides

CompositeSantecDevice.RefreshAsync()

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized RD-P.
CompositeDeviceHardwareException	Thrown when the component RLM no longer meets the RD-P hardware requirements.

TYPE	CONDITION
System.InvalidOperationException	Thrown when the RLM subdevice is uninitialized when the method is called.
UnexpectedDeviceResponseException	Thrown when the RLM subdevice gives an unexpected response during its refresh operation.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the RLM subdevice during its refresh operation.

SetCameraModeAsync(ECameraMode)

Declaration

```
public Task SetCameraModeAsync(ECameraMode cameraMode)
```

Parameters

TYPE	NAME	DESCRIPTION
ECameraMode	cameraMode	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetCustomCameraSettingsAsync(Int32, Int32)

Declaration

```
public Task SetCustomCameraSettingsAsync(int gain, int exposure)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	gain	
System.Int32	exposure	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetLossCompensationAsync(Int32)

Declaration

```
public Task SetLossCompensationAsync(int inputPower)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	inputPower	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetMappingMode(EMappingMode)

Declaration

```
public void SetMappingMode(EMappingMode mappingMode)
```

Parameters

TYPE	NAME	DESCRIPTION
EMappingMode	mappingMode	

SetMappingSensitivity(EMappingSensitivity)

Declaration

```
public void SetMappingSensitivity(EMappingSensitivity mappingSensitivity)
```

Parameters

TYPE	NAME	DESCRIPTION
EMappingSensitivity	mappingSensitivity	

StartMappingPolarityAsync(IEnumerable<CableInformation>)

Declaration

```
public Task StartMappingPolarityAsync(IEnumerable<CableInformation> cables)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Collections.Generic.IEnumerable<CableInformation>	cables	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

StartMappingPolarityAsync(Int32)

Declaration

```
public Task StartMappingPolarityAsync(int numberOfChannels)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	numberOfChannels	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

TurnOffLaserAsync(CancellationToken)

> **⚠ CAUTION**
>
> This operation is not supported for RD-SP devices. This operation will always throw a System.NotSupportedException.

Declaration

```
public Task TurnOffLaserAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

TurnOnFlashingVFLModeAsync(Int32, CancellationToken)

> **⚠ CAUTION**
>
> This operation is not supported for RD-P devices. This operation will always throw a System.NotSupportedException.

Declaration

```
public Task TurnOnFlashingVFLModeAsync(int channel, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	channel	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

TurnOnVFLModeAsync(Int32, CancellationToken)

> **⚠ CAUTION**
>
> This operation is not supported for RD-P devices. This operation will always throw a System.NotSupportedException.

Declaration

```
public Task TurnOnVFLModeAsync(int channel, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	channel	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Uninitialize(Boolean)

Declaration

```
public override void Uninitialize(bool uninitializeSubdevices)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Boolean	uninitializeSubdevices	

Overrides

CompositeSantecDevice.Uninitialize(Boolean)

Implements

IRDP<TModelType>
IRDP
ICompositeDevice
IPTM<TModelType>
IPTM
ISantecDevice
IDevice<TModelType>
IDevice
System.IDisposable

Class RDPFactory

Provides a factory for creating RD-P devices.

Inheritance

System.Object
RDPFactory

Implements

IRDPFactory

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.Polarity

Assembly: Santec.Hardware.dll

Syntax

```
public class RDPFactory : IRDPFactory
```

Constructors

RDPFactory()

Initializes a new RD=P factory.

Declaration

```
public RDPFactory()
```

RDPFactory(ICultureService)

Initializes a new device factory.

Declaration

```
public RDPFactory(ICultureService cultureService)
```

Parameters

TYPE	NAME	DESCRIPTION
Santec.Core.Globalization.ICultureService	cultureService	The culture service for localization.

Methods

CheckForCamera()

Declaration

```
public bool CheckForCamera()
```

Returns

TYPE	DESCRIPTION
System.Boolean	

CheckIfRLMMeetsHardwareRequirements(IRLM)

Declaration

```
public bool CheckIfRLMMeetsHardwareRequirements(IRLM rlm)
```

Parameters

TYPE	NAME	DESCRIPTION
IRLM	rlm	

Returns

TYPE	DESCRIPTION
System.Boolean	

CheckIfRLMMeetsHardwareRequirements(IRLM, IReadOnlyList<CameraRecord>)

Declaration

```
public bool CheckIfRLMMeetsHardwareRequirements(IRLM rlm, IReadOnlyList<CameraRecord> cameras)
```

Parameters

TYPE	NAME	DESCRIPTION
IRLM	rlm	
System.Collections.Generic.IReadOnlyList<CameraRecord>	cameras	

Returns

TYPE	DESCRIPTION
System.Boolean	

CheckIfRLMMeetsHardwareRequirementsWithDetails(IRLM)

Declaration

```
public HardwareRequirementsEvaluationResult<ERDPRequirementFailureReason>
CheckIfRLMMeetsHardwareRequirementsWithDetails(IRLM rlm)
```

Parameters

TYPE	NAME	DESCRIPTION
IRLM	rlm	

Returns

TYPE	DESCRIPTION
HardwareRequirementsEvaluationResult<ERDPRequirementFailureReason>	

CheckIfRLMMeetsHardwareRequirementsWithDetails(IRLM, IReadOnlyList<CameraRecord>)

Declaration

```
public HardwareRequirementsEvaluationResult<ERDPRequirementFailureReason>
CheckIfRLMMeetsHardwareRequirementsWithDetails(IRLM rlm, IReadOnlyList<CameraRecord> cameras)
```

Parameters

TYPE	NAME	DESCRIPTION
IRLM	rlm	
System.Collections.Generic.IReadOnlyList<CameraRecord>	cameras	

Returns

TYPE	DESCRIPTION
HardwareRequirementsEvaluationResult<ERDPRequirementFailureReason>	

CreateRDP(IPTM)

Declaration

```
public IRDP CreateRDP(IPTM ptm)
```

Parameters

TYPE	NAME	DESCRIPTION
IPTM	ptm	

Returns

TYPE	DESCRIPTION
IRDP	

CreateRDP(IPTM, IReadOnlyList<CameraRecord>)

Declaration

```
public IRDP CreateRDP(IPTM ptm, IReadOnlyList<CameraRecord> cameras)
```

Parameters

TYPE	NAME	DESCRIPTION
IPTM	ptm	
System.Collections.Generic.IReadOnlyList<CameraRecord>	cameras	

Returns

TYPE	DESCRIPTION
IRDP	

CreateRDP(IRLM)

Declaration

```
public IRDP CreateRDP(IRLM rlm)
```

Parameters

TYPE	NAME	DESCRIPTION
IRLM	rlm	

Returns

TYPE	DESCRIPTION
IRDP	

CreateRDP(IRLM, IReadOnlyList<CameraRecord>)

Declaration

```
public IRDP CreateRDP(IRLM rlm, IReadOnlyList<CameraRecord> cameras)
```

Parameters

TYPE	NAME	DESCRIPTION
IRLM	rlm	
System.Collections.Generic.IReadOnlyList<CameraRecord>	cameras	

Returns

TYPE	DESCRIPTION
IRDP	

CreateRDPWithExternalSource(Int32, Int32, ELossCompensationConfiguration)

Declaration

```
public IRDP CreateRDPWithExternalSource(int wavelength, int numberOfChannels, ELossCompensationConfiguration
lossCompensationConfiguration = ELossCompensationConfiguration.RLM)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	
System.Int32	numberOfChannels	

TYPE	NAME	DESCRIPTION
ELossCompensationConfiguration	lossCompensationConfiguration	

Returns

TYPE	DESCRIPTION
IRDP	

CreateRDPWithExternalSource(Int32, Int32, IReadOnlyList<CameraRecord>, ELossCompensationConfiguration)

Declaration

```
public IRDP CreateRDPWithExternalSource(int wavelength, int numberOfChannels, IReadOnlyList<CameraRecord>
cameras, ELossCompensationConfiguration lossCompensationConfiguration = ELossCompensationConfiguration.RLM)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	
System.Int32	numberOfChannels	
System.Collections.Generic.IReadOnlyList<CameraRecord>	cameras	
ELossCompensationConfiguration	lossCompensationConfiguration	

Returns

TYPE	DESCRIPTION
IRDP	

GetCameras()

Declaration

```
public IReadOnlyList<CameraRecord> GetCameras()
```

Returns

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<CameraRecord>	

Implements

IRDPFactory

Class RDSP

Provides a connection to an RD-SP along with properties of the device and operations that can be performed using it.

NOTE

This class cannot not be used directly. It can only be used when returned:

1. As a device by a CreateRDP(IRLM) call to a RDPFactory.

WARNING

An RD-SP is a composite device and does not support all operations of a PTM/Santec Canada family device. Unsupported operations will throw a System.NotSupportedException.

Inheritance

System.Object
CompositeSantecDevice
RDSP

Implements

IRDSP
IRDP<ERDSPModel>
IRDP
ICompositeDevice
IPTM<ERDSPModel>
IPTM
IDevice<ERDSPModel>
IRLM
ISantecDevice
IDiscretePowerMeter
ILaserSourceWithMonitor
ILaserSource
IWavelengthDevice
ISwitchController
IFiberDevice
IPowerMeterWithReferencePersistence
IPowerMeter
IInsertionLossMeter
IDetectorDevice
IDevice<ERLMModel>
IDevice
System.IDisposable

Inherited Members

CompositeSantecDevice.SerialNumber
CompositeSantecDevice.FirmwareVersion
CompositeSantecDevice.Address
CompositeSantecDevice.IsInitialized
CompositeSantecDevice.IsInitializing
CompositeSantecDevice.Subdevices
CompositeSantecDevice.Dispose()
CompositeSantecDevice.Dispose(Boolean)

CompositeSantecDevice.SetTimeout(ETimeout)
CompositeSantecDevice.GetTimeout()
CompositeSantecDevice.PingAsync()
CompositeSantecDevice.ResetAsync()
CompositeSantecDevice.Uninitialize()
System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.Polarity

Assembly: Santec.Hardware.dll

Syntax

```
public class RDSP : CompositeSantecDevice, IRDSP, IRDP<ERDSPModel>, IRDP, ICompositeDevice,
IPTM<ERDSPModel>, IPTM, IDevice<ERDSPModel>, IRLM, ISantecDevice, IDiscretePowerMeter,
ILaserSourceWithMonitor, ILaserSource, IWavelengthDevice, ISwitchController, IFiberDevice,
IPowerMeterWithReferencePersistence, IPowerMeter, IInsertionLossMeter, IDetectorDevice, IDevice<ERLMModel>,
IDevice, IDisposable
```

Properties

Detectors

Declaration

```
public IReadOnlyList<DeviceDetector> Detectors { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<DeviceDetector>	

Fiber

Declaration

```
public FiberInformation Fiber { get; }
```

Property Value

TYPE	DESCRIPTION
FiberInformation	

Model

Declaration

```
public ERDSPModel Model { get; }
```

Property Value

TYPE	DESCRIPTION
ERDSPModel	

Name

Declaration

```
public override string Name { get; }
```

Property Value

TYPE	DESCRIPTION
System.String	

Overrides

CompositeSantecDevice.Name

Remarks

> **NOTE**
>
> This property will always return "RD-P".

NumberOfChannels

Declaration

```
public int NumberOfChannels { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

NumberOfDetectors

Declaration

```
public int NumberOfDetectors { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

NumberOfOutputs

Declaration

```
public int NumberOfOutputs { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

RequiresILReferenceSave

Declaration

```
public bool RequiresILReferenceSave { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	

SupportedLengthModes

Declaration

```
public IReadOnlyList<ELengthMode> SupportedLengthModes { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<ELengthMode>	

SupportedRLModes

Declaration

```
public IReadOnlyList<ERLMode> SupportedRLModes { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<ERLMode>	

SupportsILResolution

Declaration

```
public bool SupportsILResolution { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	

SupportsMaximumResolutionRL

Declaration

```
public bool SupportsMaximumResolutionRL { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	

SupportsTurningOffLaser

Declaration

```
public bool SupportsTurningOffLaser { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	

Switches

Declaration

```
public List<DeviceSwitch> Switches { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.List<DeviceSwitch>	

Wavelength

Declaration

```
public int Wavelength { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

Wavelengths

Declaration

```
public List<int> Wavelengths { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.List<System.Int32>	

Methods

ChangeMultipleSwitchesChannelsAsync(IEnumerable<DeviceSwitchAndChannelPair>, CancellationToken)

Declaration

```
public Task ChangeMultipleSwitchesChannelsAsync(IEnumerable<DeviceSwitchAndChannelPair>
deviceSwitchesAndChannels, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Collections.Generic.IEnumerable<DeviceSwitchAndChannelPair>	deviceSwitchesAndChannels	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

ChangeOutputChannelAsync(Int32, CancellationToken)

Declaration

```
public Task ChangeOutputChannelAsync(int output, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	output	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

ChangeSwitchChannelAsync(Int32, Int32, CancellationToken)

Declaration

```
public Task ChangeSwitchChannelAsync(int switchIndex, int channel, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	switchIndex	
System.Int32	channel	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

CollectOutputMappingDataAsync(Int32, Int32, CancellationToken)

Declaration

```
public Task<bool> CollectOutputMappingDataAsync(int cable, int fiber, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	cable	
System.Int32	fiber	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Boolean>	

CollectOutputMappingDataAsync(Int32, CancellationToken)

Declaration

```
public Task<bool> CollectOutputMappingDataAsync(int channel, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	channel	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Boolean>	

Dispose(Boolean, Boolean)

Declaration

```
protected override void Dispose(bool disposing, bool disposeSubdevices)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Boolean	disposing	
System.Boolean	disposeSubdevices	

Overrides

CompositeSantecDevice.Dispose(Boolean, Boolean)

EnableDHCPAsync(CancellationToken)

Declaration

```
public override Task EnableDHCPAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Overrides

CompositeSantecDevice.EnableDHCPAsync(CancellationToken)

EndMappingPolarityAsync()

Declaration

```
public Task EndMappingPolarityAsync()
```

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

GetActiveLaserAsync(CancellationToken)

Declaration

```
public Task<int> GetActiveLaserAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Int32>	

GetCameraMode()

Declaration

```
public ECameraMode GetCameraMode()
```

Returns

TYPE	DESCRIPTION
ECameraMode	

GetCustomCameraSettings()

Declaration

```
public LegacyCustomCameraSettings GetCustomCameraSettings()
```

Returns

TYPE	DESCRIPTION
LegacyCustomCameraSettings	

GetDUTILAsync(Int32, CancellationToken)

Declaration

```
public Task<double> GetDUTILAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

GetFactoryPowerAsync(Int32, Int32, CancellationToken)

Declaration

```
public Task<FactoryPowerReading> GetFactoryPowerAsync(int output, int wavelength, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	output	
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<FactoryPowerReading>	

GetGainModeAsync(CancellationToken)

Declaration

```
public Task<EGainMode> GetGainModeAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<EGainMode>	

GetGatewayAsync(CancellationToken)

Declaration

```
public override Task<IPAddress> GetGatewayAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Net.IPAddress>	

Overrides

CompositeSantecDevice.GetGatewayAsync(CancellationToken)

GetHostnameAsync(CancellationToken)

Declaration

```
public override Task<string> GetHostnameAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.String>	

Overrides

CompositeSantecDevice.GetHostnameAsync(CancellationToken)

GetILReferenceAsync(Int32, Int32, CancellationToken)

Declaration

```
public Task<double> GetILReferenceAsync(int detector, int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

GetILResolutionAsync(CancellationToken)

Declaration

```
public Task<EResolution> GetILResolutionAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<EResolution>	

GetInteractionModeAsync(CancellationToken)

Declaration

```
public override Task<EInteractionMode> GetInteractionModeAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<EInteractionMode>	

Overrides

CompositeSantecDevice.GetInteractionModeAsync(CancellationToken)

GetIPAddressAsync(CancellationToken)

Declaration

```
public override Task<IPAddress> GetIPAddressAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Net.IPAddress>	

Overrides

CompositeSantecDevice.GetIPAddressAsync(CancellationToken)

GetLengthModeAsync(CancellationToken)

Declaration

```
public Task<ELengthMode> GetLengthModeAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<ELengthMode>	

GetLossCompensation()

Declaration

```
public int GetLossCompensation()
```

Returns

TYPE	DESCRIPTION
System.Int32	

GetMappingInProgress()

Declaration

```
public bool GetMappingInProgress()
```

Returns

TYPE	DESCRIPTION
System.Boolean	

GetMappingMode()

Declaration

```
public EMappingMode GetMappingMode()
```

Returns

TYPE	DESCRIPTION
EMappingMode	

GetMappingSensitivity()

Declaration

```
public EMappingSensitivity GetMappingSensitivity()
```

Returns

TYPE	DESCRIPTION
EMappingSensitivity	

GetOutputChannelAsync(CancellationToken)

Declaration

```
public Task<int> GetOutputChannelAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Int32>	

GetOutputMappingDataCollected(Int32)

Declaration

```
public bool GetOutputMappingDataCollected(int channel)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	channel	

Returns

TYPE	DESCRIPTION
System.Boolean	

GetOutputMappingDataCollected(Int32, Int32)

Declaration

```
public bool GetOutputMappingDataCollected(int cable, int fiber)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	cable	
System.Int32	fiber	

Returns

TYPE	DESCRIPTION
System.Boolean	

GetOutputMappingImages(Int32)

Declaration

```
public IReadOnlyList<Mat> GetOutputMappingImages(int channel)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	channel	

Returns

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<OpenCvSharp.Mat>	

GetOutputMappingImages(Int32, Int32)

Declaration

```
public IReadOnlyList<Mat> GetOutputMappingImages(int cable, int fiber)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	cable	
System.Int32	fiber	

Returns

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<OpenCvSharp.Mat>	

GetPositionBModeAsync(CancellationToken)

Declaration

```
public Task<EPositionBMode> GetPositionBModeAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<EPositionBMode>	

GetRLModeAsync(CancellationToken)

Declaration

```
public Task<ERLMode> GetRLModeAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<ERLMode>	

GetRLReferenceAsync(CancellationToken)

Declaration

```
public Task<double> GetRLReferenceAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

GetSubnetPrefixLengthAsync(CancellationToken)

Declaration

```
public override Task<int> GetSubnetPrefixLengthAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Int32>	

Overrides

CompositeSantecDevice.GetSubnetPrefixLengthAsync(CancellationToken)

GetSwitchChannelAsync(Int32, CancellationToken)

Declaration

```
public Task<int> GetSwitchChannelAsync(int switchIndex, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	switchIndex	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Int32>	

GetTraceDiagnosticsAsync(CancellationToken)

Declaration

```
public Task<TraceDiagnostics> GetTraceDiagnosticsAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<TraceDiagnostics>	

InitializeAsync()

Declaration

```
public override async Task InitializeAsync()
```

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Overrides

CompositeSantecDevice.InitializeAsync()

Exceptions

TYPE	CONDITION
CompositeDeviceHardwareException	Thrown when the component RLM no longer meets the RD-SP hardware requirements.

MapOutputAsync(Int32, CancellationToken)

> **⚠ CAUTION**
>
> This operation is not supported for RD-SP devices. This operation will always throw a System.NotSupportedException.

Declaration

```
public Task<int?> MapOutputAsync(int channel, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	channel	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Nullable<System.Int32>>	

MapPolarityAsync(Int32, IEnumerable<Nullable<Int32>>, CancellationToken)

Declaration

```
public Task<IReadOnlyList<int?>> MapPolarityAsync(int numberOfChannels, IEnumerable<int?> expectedMapping,
CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	numberOfChannels	
System.Collections.Generic.IEnumerable<System.Nullable<System.Int32>>	expectedMapping	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Collections.Generic.IReadOnlyList<System.Nullable<System.Int32>>>	

MapPolarityAsync(Int32, CancellationToken)

Declaration

```
public Task<IReadOnlyList<int?>> MapPolarityAsync(int numberOfChannels, CancellationToken cancellationToken
= default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	numberOfChannels	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Collections.Generic.IReadOnlyList<System.Nullable<System.Int32>>>	

ProcessMappingData(IEnumerable<Nullable<Int32>>)

Declaration

```
public IReadOnlyList<int?> ProcessMappingData(IEnumerable<int?> expectedMapping = null)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Collections.Generic.IEnumerable<System.Nullable<System.Int32>>	expectedMapping	

Returns

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<System.Nullable<System.Int32>>	

ProcessMappingData(Int32, IEnumerable<Nullable<Int32>>)

Declaration

```
public IReadOnlyList<int?> ProcessMappingData(int cable, IEnumerable<int?> expectedMapping = null)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	cable	
System.Collections.Generic.IEnumerable<System.Nullable<System.Int32>>	expectedMapping	

Returns

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<System.Nullable<System.Int32>>	

QueryAsync(String, CancellationToken)

Declaration

```
public override Task<string> QueryAsync(string command, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	command	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.String>	

Overrides

CompositeSantecDevice.QueryAsync(String, CancellationToken)

ReadAsync(CancellationToken)

Declaration

```
public override Task<string> ReadAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.String>	

Overrides

CompositeSantecDevice.ReadAsync(CancellationToken)

ReadILAsync(Int32, Int32, CancellationToken)

Declaration

```
public Task<double> ReadILAsync(int detector, int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

ReadMonitorPowerAsync(Int32, CancellationToken)

Declaration

```
public Task<double> ReadMonitorPowerAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

ReadMultipleILsAsync(IEnumerable<Int32>, Int32, CancellationToken)

Declaration

```
public Task<List<double>> ReadMultipleILsAsync(IEnumerable<int> detectors, int wavelength, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Collections.Generic.IEnumerable<System.Int32>	detectors	
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Collections.Generic.List<System.Double>>	

ReadMultiplePowersAsync(IEnumerable<Int32>, Int32, CancellationToken)

Declaration

```
public Task<List<double>> ReadMultiplePowersAsync(IEnumerable<int> detectors, int wavelength,
CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Collections.Generic.IEnumerable<System.Int32>	detectors	
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Collections.Generic.List<System.Double>>	

ReadPowerAsync(Int32, Int32, CancellationToken)

Declaration

```
public Task<double> ReadPowerAsync(int detector, int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

ReadRLAsync(Int32, Double, Double, CancellationToken)

Declaration

```
public Task<RLReading> ReadRLAsync(int wavelength, double lengthReferenceA, double lengthReferenceB,
CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	
System.Double	lengthReferenceA	
System.Double	lengthReferenceB	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<RLReading>	

ReadRLAsync(Int32, Double, CancellationToken)

Declaration

```
public Task<RLReading> ReadRLAsync(int wavelength, double lengthReference, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	
System.Double	lengthReference	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<RLReading>	

ReadRLAsync(Int32, CancellationToken)

Declaration

```
public Task<RLReading> ReadRLAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<RLReading>	

ReferenceILAsync(Int32, Int32, Boolean, CancellationToken)

Declaration

```
public Task<double> ReferenceILAsync(int detector, int wavelength, bool applyReferenceToAllDetectors,
CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Int32	wavelength	
System.Boolean	applyReferenceToAllDetectors	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

ReferenceILAsync(Int32, Int32, CancellationToken)

Declaration

```
public Task<double> ReferenceILAsync(int detector, int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

ReferenceRLAsync(CancellationToken)

Declaration

```
public Task<double> ReferenceRLAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

RefreshAsync()

Declaration

```
public override async Task RefreshAsync()
```

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Overrides

CompositeSantecDevice.RefreshAsync()

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized RD-SP.
CompositeDeviceHardwareException	Thrown when the component RLM no longer meets the RD-SP hardware requirements.
System.InvalidOperationException	Thrown when the RLM subdevice is uninitialized when the method is called.
UnexpectedDeviceResponseException	Thrown when the RLM subdevice gives an unexpected response during its refresh operation.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the RLM subdevice during its refresh operation.

SaveILReferencesAsync(CancellationToken)

Declaration

```
public Task SaveILReferencesAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetCameraModeAsync(ECameraMode)

Declaration

```
public Task SetCameraModeAsync(ECameraMode cameraMode)
```

Parameters

TYPE	NAME	DESCRIPTION
ECameraMode	cameraMode	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetCustomCameraSettingsAsync(Int32, Int32)

Declaration

```
public Task SetCustomCameraSettingsAsync(int gain, int exposure)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	gain	
System.Int32	exposure	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetDUTILAsync(Int32, Double, CancellationToken)

Declaration

```
public Task SetDUTILAsync(int wavelength, double il, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	
System.Double	il	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetGainModeAsync(EGainMode, CancellationToken)

Declaration

```
public Task SetGainModeAsync(EGainMode gainMode, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
EGainMode	gainMode	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetGatewayAsync(IPAddress, CancellationToken)

Declaration

```
public override Task SetGatewayAsync(IPAddress address, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Net.IPAddress	address	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Overrides

CompositeSantecDevice.SetGatewayAsync(IPAddress, CancellationToken)

SetHostnameAsync(String, CancellationToken)

Declaration

```
public override Task SetHostnameAsync(string hostname, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	hostname	

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Overrides

CompositeSantecDevice.SetHostnameAsync(String, CancellationToken)

SetILReferenceAsync(Int32, Int32, Double, CancellationToken)

Declaration

```
public Task SetILReferenceAsync(int detector, int wavelength, double reference, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Int32	wavelength	
System.Double	reference	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetILResolutionAsync(EResolution, CancellationToken)

Declaration

```
public Task SetILResolutionAsync(EResolution resolution, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
EResolution	resolution	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetInteractionModeAsync(EInteractionMode, CancellationToken)

Declaration

```
public override Task SetInteractionModeAsync(EInteractionMode interactionMode, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
EInteractionMode	interactionMode	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Overrides

CompositeSantecDevice.SetInteractionModeAsync(EInteractionMode, CancellationToken)

SetIPAddressAsync(IPAddress, CancellationToken)

Declaration

```
public override Task SetIPAddressAsync(IPAddress address, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Net.IPAddress	address	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Overrides

CompositeSantecDevice.SetIPAddressAsync(IPAddress, CancellationToken)

SetLengthModeAsync(ELengthMode, CancellationToken)

Declaration

```
public Task SetLengthModeAsync(ELengthMode lengthMode, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
ELengthMode	lengthMode	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetLossCompensationAsync(Int32)

Declaration

```
public Task SetLossCompensationAsync(int inputPower)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	inputPower	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetMappingMode(EMappingMode)

Declaration

```
public void SetMappingMode(EMappingMode mappingMode)
```

Parameters

TYPE	NAME	DESCRIPTION
EMappingMode	mappingMode	

SetMappingSensitivity(EMappingSensitivity)

Declaration

```
public void SetMappingSensitivity(EMappingSensitivity mappingSensitivity)
```

Parameters

TYPE	NAME	DESCRIPTION
EMappingSensitivity	mappingSensitivity	

SetPositionBModeAsync(EPositionBMode, CancellationToken)

Declaration

```
public Task SetPositionBModeAsync(EPositionBMode positionBMode, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
EPositionBMode	positionBMode	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetRLModeAsync(ERLMode, CancellationToken)

Declaration

```
public Task SetRLModeAsync(ERLMode rlMode, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
ERLMode	rlMode	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetRLReferenceAsync(Double, CancellationToken)

Declaration

```
public Task SetRLReferenceAsync(double reference, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Double	reference	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetSubnetPrefixLengthAsync(Int32, CancellationToken)

Declaration

```
public override Task SetSubnetPrefixLengthAsync(int prefixLength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	prefixLength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Overrides

CompositeSantecDevice.SetSubnetPrefixLengthAsync(Int32, CancellationToken)

StartMappingPolarityAsync(IEnumerable<CableInformation>)

Declaration

```
public Task StartMappingPolarityAsync(IEnumerable<CableInformation> cables)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Collections.Generic.IEnumerable<CableInformation>	cables	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

StartMappingPolarityAsync(Int32)

Declaration

```
public Task StartMappingPolarityAsync(int numberOfChannels)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	numberOfChannels	

Returns

TYPE	DESCRIPTION

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

TurnOffLaserAsync(CancellationToken)

Declaration

```
public Task TurnOffLaserAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

TurnOnFlashingVFLModeAsync(Int32, CancellationToken)

> **⚠ CAUTION**
>
> This operation is not supported for RD-SP devices. This operation will always throw a System.NotSupportedException.

Declaration

```
public Task TurnOnFlashingVFLModeAsync(int channel, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	channel	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

TurnOnLaserAsync(Int32, CancellationToken)

Declaration

```
public Task TurnOnLaserAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

TurnOnVFLModeAsync(Int32, CancellationToken)

> **⚠ CAUTION**
>
> This operation is not supported for RD-P devices. This operation will always throw a System.NotSupportedException.

Declaration

```
public Task TurnOnVFLModeAsync(int channel, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	channel	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Uninitialize(Boolean)

Declaration

```
public override void Uninitialize(bool uninitializeSubdevices)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Boolean	uninitializeSubdevices	

Overrides

CompositeSantecDevice.Uninitialize(Boolean)

WriteAsync(String, CancellationToken)

Declaration

```
public override Task WriteAsync(string command, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	command	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Overrides

CompositeSantecDevice.WriteAsync(String, CancellationToken)

Explicit Interface Implementations

IDetectorDevice.Detectors

Declaration

```
IReadOnlyList<IModule> IDetectorDevice.Detectors { get; }
```

Returns

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<IModule>	

IDevice<ERLMModel>.Model

Declaration

```
ERLMModel IDevice<ERLMModel>.Model { get; }
```

Returns

TYPE	DESCRIPTION
ERLMModel	

Implements

IRDSP
IRDP<TModelType>
IRDP
ICompositeDevice
IPTM<TModelType>
IPTM
IDevice<TModelType>
IRLM
ISantecDevice

IDiscretePowerMeter
ILaserSourceWithMonitor
ILaserSource
IWavelengthDevice
ISwitchController
IFiberDevice
IPowerMeterWithReferencePersistence
IPowerMeter
IInsertionLossMeter
IDetectorDevice
IDevice<TModelType>
IDevice
System.IDisposable

Extension Methods

RLMExtensions.IsMultimodeRLM100(IRLM)

Class RDSPCameraValidator

Provides system-level validation for RD-SPs.

Inheritance

System.Object
RDSPCameraValidator

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.Polarity

Assembly: Santec.Hardware.dll

Syntax

```
public class RDSPCameraValidator
```

Constructors

RDSPCameraValidator(ICatalogProvider)

Initializes a new instance of the RDSPCameraValidator class.

Declaration

```
public RDSPCameraValidator(ICatalogProvider catalogProvider)
```

Parameters

TYPE	NAME	DESCRIPTION
Santec.Core.Localization.ICatalogProvider	catalogProvider	The catalog provider for localized messages.

Methods

CheckIfAllCamerasAssociatedWithRDSPDetectors(IEnumerable<IRLM>, IReadOnlyList<CameraRecord>)

Checks whether all cameras are associated with an RD-SP detector across all provided RLMs.

Declaration

```
public SystemHardwareRequirementsEvaluationResult<ERDSPSystemHardwareRequirementsFailureReason>
CheckIfAllCamerasAssociatedWithRDSPDetectors(IEnumerable<IRLM> rlms, IReadOnlyList<CameraRecord> cameras)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Collections.Generic.IEnumerable<IRLM>	rlms	The RLMs to check.
System.Collections.Generic.IReadOnlyList<CameraRecord>	cameras	The cameras to validate.

Returns

TYPE	DESCRIPTION
SystemHardwareRequirementsEvaluationResult<ERDSPSystemHardwareRequirementsFailureReason>	A SystemHardwareRequirementsEvaluationResult<T> indicating whether all cameras are associated with an RD-SP detector.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when any of the specified RLMs are uninitialized.
System.ArgumentNullException	Thrown when the RLMs list or cameras list are null.
System.ArgumentException	Thrown when the camera list is empty.

Class SimulatedPTM

Provides a simulated PTM with configurable polarity mappings.

> **NOTE**
>
> This class is intended for testing/development purposes. It operates much quicker than an actual PTM. It can only be created using a SimulatedDeviceFactory. It should only be used when returned as part of a list of devices by a DetectHardwareAsync(EConnectionTypesToDetect) or DetectHardwareAsync<T>(EConnectionTypesToDetect) call to a SimulatedHardwareDetectionService. A default detected A polarity mapping is set but the detected mapping can be configured via the ConfigureMapping(IEnumerable<Nullable<Int32>>) method. Additionally, the chance for the polarity mapping to fail and return a different detected mapping can also be configured with the ConfigureFailureChance(Int32) method. The default failure chance is 0%.

Inheritance

System.Object
SimulatedSantecDevice
SimulatedPTM

Implements

ISimulatedDevice
IPTM<EPTMModel>
IPTM
ISantecDevice
IDevice<EPTMModel>
IDevice
System.IDisposable

Inherited Members

SimulatedSantecDevice.creatingDevice
SimulatedSantecDevice.Dispose()
SimulatedSantecDevice.SerialNumber
SimulatedSantecDevice.FirmwareVersion
SimulatedSantecDevice.Address
SimulatedSantecDevice.IsInitialized
SimulatedSantecDevice.QueryResponses
SimulatedSantecDevice.ConfigureQueryResponses(IEnumerable<QueryResponsePair>)
SimulatedSantecDevice.GetTimeout()
SimulatedSantecDevice.SetTimeout(ETimeout)
SimulatedSantecDevice.GetInteractionModeAsync(CancellationToken)
SimulatedSantecDevice.GetIPAddressAsync(CancellationToken)
SimulatedSantecDevice.SetIPAddressAsync(IPAddress, CancellationToken)
SimulatedSantecDevice.EnableDHCPAsync(CancellationToken)
SimulatedSantecDevice.GetGatewayAsync(CancellationToken)
SimulatedSantecDevice.SetGatewayAsync(IPAddress, CancellationToken)
SimulatedSantecDevice.GetSubnetPrefixLengthAsync(CancellationToken)
SimulatedSantecDevice.SetSubnetPrefixLengthAsync(Int32, CancellationToken)
SimulatedSantecDevice.GetHostnameAsync(CancellationToken)
SimulatedSantecDevice.SetHostnameAsync(String, CancellationToken)
SimulatedSantecDevice.SetInteractionModeAsync(EInteractionMode, CancellationToken)
SimulatedSantecDevice.InitializeAsync()
SimulatedSantecDevice.PingAsync()
SimulatedSantecDevice.Uninitialize()
SimulatedSantecDevice.ResetAsync()

SimulatedSantecDevice.RefreshAsync()
SimulatedSantecDevice.QueryAsync(String, CancellationToken)
SimulatedSantecDevice.WriteAsync(String, CancellationToken)
SimulatedSantecDevice.ReadAsync(CancellationToken)
System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.Polarity

Assembly: Santec.Hardware.dll

Syntax

```
public class SimulatedPTM : SimulatedSantecDevice, ISimulatedDevice, IPTM<EPTMModel>, IPTM, ISantecDevice,
IDevice<EPTMModel>, IDevice, IDisposable
```

Properties

FailureChance

Declaration

```
public int FailureChance { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

Model

Declaration

```
public EPTMModel Model { get; }
```

Property Value

TYPE	DESCRIPTION
EPTMModel	

Name

Declaration

```
public override string Name { get; }
```

Property Value

TYPE	DESCRIPTION
System.String	

Overrides

SimulatedSantecDevice.Name

NumberOfChannels

Declaration

```
public int NumberOfChannels { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

PolarityMapping

Declaration

```
public IReadOnlyList<int?> PolarityMapping { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<System.Nullable<System.Int32>>	

Methods

ConfigureFailureChance(Int32)

Declaration

```
public void ConfigureFailureChance(int failureChance)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	failureChance	

ConfigureMapping(IEnumerable<Nullable<Int32>>)

Declaration

```
public void ConfigureMapping(IEnumerable<int?> mapping)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Collections.Generic.IEnumerable<System.Nullable<System.Int32>>	mapping	

MapOutputAsync(Int32, CancellationToken)

Declaration

```
public async Task<int?> MapOutputAsync(int channel, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	channel	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Nullable<System.Int32>>	

MapPolarityAsync(Int32, CancellationToken)

Declaration

```
public async Task<IReadOnlyList<int?>> MapPolarityAsync(int numberOfChannels, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	numberOfChannels	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Collections.Generic.IReadOnlyList<System.Nullable<System.Int32>>>	

TurnOffLaserAsync(CancellationToken)

Declaration

```
public async Task TurnOffLaserAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

TurnOnFlashingVFLModeAsync(Int32, CancellationToken)

Declaration

```
public async Task TurnOnFlashingVFLModeAsync(int channel, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	channel	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

TurnOnVFLModeAsync(Int32, CancellationToken)

Declaration

```
public async Task TurnOnVFLModeAsync(int channel, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	channel	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Implements

ISimulatedDevice
IPTM<TModelType>
IPTM
ISantecDevice
IDevice<TModelType>
IDevice
System.IDisposable

Class SimulatedRDP

Provides a simulated RD-P with configurable polarity mappings.

> **NOTE**
>
> This class is intended for testing/development purposes. It operates much quicker than an actual RD-P. It can only be created using a SimulatedRDPFactory. It should only be used when returned as part of a list of devices by a DetectHardwareAsync(EConnectionTypesToDetect) or DetectHardwareAsync<T>(EConnectionTypesToDetect) call to a SimulatedHardwareDetectionService. A default detected A polarity mapping is set but the detected mapping can be configured via the ConfigureMapping(IEnumerable<Nullable<Int32>>) method. Additionally, the chance for the polarity mapping to fail and return a different detected mapping can also be configured with the ConfigureFailureChance(Int32) method. The default failure chance is 0%.

Inheritance

System.Object
CompositeSantecDevice
SimulatedRDP

Implements

IRDP<ERDPModel>
IRDP
ICompositeDevice
IPTM<ERDPModel>
IPTM
ISantecDevice
IDevice<ERDPModel>
IDevice
System.IDisposable

Inherited Members

CompositeSantecDevice.SerialNumber
CompositeSantecDevice.FirmwareVersion
CompositeSantecDevice.Address
CompositeSantecDevice.IsInitialized
CompositeSantecDevice.IsInitializing
CompositeSantecDevice.Subdevices
CompositeSantecDevice.Dispose()
CompositeSantecDevice.Dispose(Boolean)
CompositeSantecDevice.Dispose(Boolean, Boolean)
CompositeSantecDevice.SetTimeout(ETimeout)
CompositeSantecDevice.GetTimeout()
CompositeSantecDevice.EnableDHCPAsync(CancellationToken)
CompositeSantecDevice.GetGatewayAsync(CancellationToken)
CompositeSantecDevice.GetHostnameAsync(CancellationToken)
CompositeSantecDevice.GetInteractionModeAsync(CancellationToken)
CompositeSantecDevice.GetIPAddressAsync(CancellationToken)
CompositeSantecDevice.GetSubnetPrefixLengthAsync(CancellationToken)
CompositeSantecDevice.PingAsync()
CompositeSantecDevice.ResetAsync()
CompositeSantecDevice.SetGatewayAsync(IPAddress, CancellationToken)
CompositeSantecDevice.SetHostnameAsync(String, CancellationToken)
CompositeSantecDevice.SetInteractionModeAsync(EInteractionMode, CancellationToken)
CompositeSantecDevice.SetIPAddressAsync(IPAddress, CancellationToken)
CompositeSantecDevice.SetSubnetPrefixLengthAsync(Int32, CancellationToken)
CompositeSantecDevice.QueryAsync(String, CancellationToken)
CompositeSantecDevice.WriteAsync(String, CancellationToken)

CompositeSantecDevice.ReadAsync(CancellationToken)
CompositeSantecDevice.Uninitialize()
CompositeSantecDevice.Uninitialize(Boolean)
System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.Polarity

Assembly: Santec.Hardware.dll

Syntax

```
public class SimulatedRDP : CompositeSantecDevice, IRDP<ERDPModel>, IRDP, ICompositeDevice, IPTM<ERDPModel>, IPTM,
ISantecDevice, IDevice<ERDPModel>, IDevice, IDisposable
```

Properties

FailureChance

Declaration

```
public int FailureChance { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

Model

Declaration

```
public ERDPModel Model { get; }
```

Property Value

TYPE	DESCRIPTION
ERDPModel	

Name

Declaration

```
public override string Name { get; }
```

Property Value

TYPE	DESCRIPTION
System.String	

Overrides

CompositeSantecDevice.Name

NumberOfChannels

Declaration

```
public int NumberOfChannels { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

PolarityMapping

Declaration

```
public IReadOnlyList<int?> PolarityMapping { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<System.Nullable<System.Int32>>	

PolarityMappings

Declaration

```
public IReadOnlyList<IReadOnlyList<int?>> PolarityMappings { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<System.Collections.Generic.IReadOnlyList<System.Nullable<System.Int32>>>	

Wavelength

Declaration

```
public int Wavelength { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

Methods

CollectOutputMappingDataAsync(Int32, Int32, CancellationToken)

Declaration

```
public Task<bool> CollectOutputMappingDataAsync(int cable, int fiber, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	cable	
System.Int32	fiber	

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Boolean>	

CollectOutputMappingDataAsync(Int32, CancellationToken)

Declaration

```
public Task<bool> CollectOutputMappingDataAsync(int channel, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	channel	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Boolean>	

ConfigureFailureChance(Int32)

Declaration

```
public void ConfigureFailureChance(int failureChance)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	failureChance	

ConfigureMapping(IEnumerable<Nullable<Int32>>)

Declaration

```
public void ConfigureMapping(IEnumerable<int?> mapping)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Collections.Generic.IEnumerable<System.Nullable<System.Int32>>	mapping	

ConfigureMappings(IEnumerable<IEnumerable<Nullable<Int32>>>)

Declaration

```
public void ConfigureMappings(IEnumerable<IEnumerable<int?>> mappings)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Collections.Generic.IEnumerable<System.Collections.Generic.IEnumerable<System.Nullable<System.Int32>>>	mappings	

EndMappingPolarityAsync()

Declaration

```
public Task EndMappingPolarityAsync()
```

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

GetCameraMode()

Declaration

```
public ECameraMode GetCameraMode()
```

Returns

TYPE	DESCRIPTION
ECameraMode	

GetCustomCameraSettings()

Declaration

```
public LegacyCustomCameraSettings GetCustomCameraSettings()
```

Returns

TYPE	DESCRIPTION
LegacyCustomCameraSettings	

GetLossCompensation()

Declaration

```
public int GetLossCompensation()
```

Returns

TYPE	DESCRIPTION
System.Int32	

GetMappingInProgress()

Declaration

```
public bool GetMappingInProgress()
```

Returns

TYPE	DESCRIPTION
System.Boolean	

GetMappingMode()

Declaration

```
public EMappingMode GetMappingMode()
```

Returns

TYPE	DESCRIPTION
EMappingMode	

GetMappingSensitivity()

Declaration

```
public EMappingSensitivity GetMappingSensitivity()
```

Returns

TYPE	DESCRIPTION
EMappingSensitivity	

GetOutputMappingDataCollected(Int32)

Declaration

```
public bool GetOutputMappingDataCollected(int channel)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	channel	

Returns

TYPE	DESCRIPTION
System.Boolean	

GetOutputMappingDataCollected(Int32, Int32)

Declaration

```
public bool GetOutputMappingDataCollected(int cable, int fiber)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	cable	
System.Int32	fiber	

Returns

TYPE	DESCRIPTION
System.Boolean	

GetOutputMappingImages(Int32)

Declaration

```
public IReadOnlyList<Mat> GetOutputMappingImages(int channel)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	channel	

Returns

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<OpenCvSharp.Mat>	

GetOutputMappingImages(Int32, Int32)

Declaration

```
public IReadOnlyList<Mat> GetOutputMappingImages(int cable, int fiber)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	cable	
System.Int32	fiber	

Returns

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<OpenCvSharp.Mat>	

InitializeAsync()

Declaration

```
public override async Task InitializeAsync()
```

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Overrides

CompositeSantecDevice.InitializeAsync()

Exceptions

TYPE	CONDITION
CompositeDeviceHardwareException	Thrown when the component RLM no longer meets the RD-P hardware requirements.

MapOutputAsync(Int32, CancellationToken)

> **⚠ CAUTION**
>
> This operation is not supported for RD-P devices. This operation will always throw a System.NotSupportedException.

Declaration

```
public Task<int?> MapOutputAsync(int channel, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	channel	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Nullable<System.Int32>>	

MapPolarityAsync(Int32, IEnumerable<Nullable<Int32>>, CancellationToken)

Declaration

```
public Task<IReadOnlyList<int?>> MapPolarityAsync(int numberOfChannels, IEnumerable<int?> expectedMapping,
CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	numberOfChannels	
System.Collections.Generic.IEnumerable<System.Nullable<System.Int32>>	expectedMapping	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Collections.Generic.IReadOnlyList<System.Nullable<System.Int32>>>	

MapPolarityAsync(Int32, CancellationToken)

Declaration

```
public Task<IReadOnlyList<int?>> MapPolarityAsync(int numberOfChannels, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	numberOfChannels	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Collections.Generic.IReadOnlyList<System.Nullable<System.Int32>>>	

ProcessMappingData(IEnumerable<Nullable<Int32>>)

Declaration

```
public IReadOnlyList<int?> ProcessMappingData(IEnumerable<int?> expectedMapping)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Collections.Generic.IEnumerable<System.Nullable<System.Int32>>	expectedMapping	

Returns

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<System.Nullable<System.Int32>>	

ProcessMappingData(Int32, IEnumerable<Nullable<Int32>>)

Declaration

```
public IReadOnlyList<int?> ProcessMappingData(int cable, IEnumerable<int?> expectedMapping)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	cable	
System.Collections.Generic.IEnumerable<System.Nullable<System.Int32>>	expectedMapping	

Returns

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<System.Nullable<System.Int32>>	

RefreshAsync()

Declaration

```
public override async Task RefreshAsync()
```

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

TYPE	DESCRIPTION

Overrides

CompositeSantecDevice.RefreshAsync()

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized RD-P.
CompositeDeviceHardwareException	Thrown when the subdevice no longer meets the RD-P hardware requirements.
System.InvalidOperationException	Thrown when the subdevice is uninitialized when the method is called.
UnexpectedDeviceResponseException	Thrown when the subdevice gives an unexpected response during its refresh operation.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the subdevice during its refresh operation.

SetCameraModeAsync(ECameraMode)

Declaration

```
public Task SetCameraModeAsync(ECameraMode cameraMode)
```

Parameters

TYPE	NAME	DESCRIPTION
ECameraMode	cameraMode	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetCustomCameraSettingsAsync(Int32, Int32)

Declaration

```
public Task SetCustomCameraSettingsAsync(int exposure, int gain)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	exposure	
System.Int32	gain	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetLossCompensationAsync(Int32)

Declaration

```
public Task SetLossCompensationAsync(int inputPower)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	inputPower	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetMappingMode(EMappingMode)

Declaration

```
public void SetMappingMode(EMappingMode mappingMode)
```

Parameters

TYPE	NAME	DESCRIPTION
EMappingMode	mappingMode	

SetMappingSensitivity(EMappingSensitivity)

Declaration

```
public void SetMappingSensitivity(EMappingSensitivity mappingSensitivity)
```

Parameters

TYPE	NAME	DESCRIPTION
EMappingSensitivity	mappingSensitivity	

StartMappingPolarityAsync(IEnumerable<CableInformation>)

Declaration

```
public Task StartMappingPolarityAsync(IEnumerable<CableInformation> cables)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Collections.Generic.IEnumerable<CableInformation>	cables	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

StartMappingPolarityAsync(Int32)

Declaration

```
public Task StartMappingPolarityAsync(int numberOfChannels)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	numberOfChannels	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

TurnOffLaserAsync(CancellationToken)

> **⚠ CAUTION**
>
> This operation is not supported for RDP devices. This operation will always throw a System.NotSupportedException.

Declaration

```
public Task TurnOffLaserAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

TurnOnFlashingVFLModeAsync(Int32, CancellationToken)

> **⚠ CAUTION**
>
> This operation is not supported for RDP devices. This operation will always throw a System.NotSupportedException.

Declaration

```
public Task TurnOnFlashingVFLModeAsync(int channel, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	channel	

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

TurnOnVFLModeAsync(Int32, CancellationToken)

> **⚠ CAUTION**
>
> This operation is not supported for RDP devices. This operation will always throw a System.NotSupportedException.

Declaration

```
public Task TurnOnVFLModeAsync(int channel, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	channel	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Implements

IRDP<TModelType>
IRDP
ICompositeDevice
IPTM<TModelType>
IPTM
ISantecDevice
IDevice<TModelType>
IDevice
System.IDisposable

Class SimulatedRDPFactory

Provides a factory for creating simulated RD-P devices.

> **NOTE**
>
> This class is intended for testing/development purposes.

Inheritance

System.Object
SimulatedRDPFactory

Implements

IRDPFactory

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.Polarity

Assembly: Santec.Hardware.dll

Syntax

```
public class SimulatedRDPFactory : IRDPFactory
```

Constructors

SimulatedRDPFactory()

Initialize a new simulated RD-P factory.

Declaration

```
public SimulatedRDPFactory()
```

SimulatedRDPFactory(ICultureService)

Initialize a new simulated RD-P factory.

Declaration

```
public SimulatedRDPFactory(ICultureService cultureService)
```

Parameters

TYPE	NAME	DESCRIPTION
Santec.Core.Globalization.ICultureService	cultureService	The culture service for localization.

Properties

AlwaysCreateRDSP

Get whether the simulated RD-P factory will always create an RD-SP from an RLM.

Declaration

```
public bool AlwaysCreateRDSP { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	`true` if the RD-P factory will always create RD-SPs; otherwise, `false`.

Remarks

Use ConfigureRDSPCreation(Boolean) to set this value.

Methods

CheckForCamera()

Declaration

```
public bool CheckForCamera()
```

Returns

TYPE	DESCRIPTION
System.Boolean	`true` if a RD-P camera is found; otherwise, `false`. The default value is `true`.

Remarks

Use ConfigureCameras(IReadOnlyList<CameraRecord>) to set whether a camera is detected.

CheckIfRLMMeetsHardwareRequirements(IRLM)

Declaration

```
public bool CheckIfRLMMeetsHardwareRequirements(IRLM RLM)
```

Parameters

TYPE	NAME	DESCRIPTION
IRLM	RLM	

Returns

TYPE	DESCRIPTION
System.Boolean	

CheckIfRLMMeetsHardwareRequirements(IRLM, IReadOnlyList<CameraRecord>)

Declaration

```
public bool CheckIfRLMMeetsHardwareRequirements(IRLM rlm, IReadOnlyList<CameraRecord> cameras)
```

Parameters

TYPE	NAME	DESCRIPTION
IRLM	rlm	
System.Collections.Generic.IReadOnlyList<CameraRecord>	cameras	

Returns

TYPE	DESCRIPTION
System.Boolean	

CheckIfRLMMeetsHardwareRequirementsWithDetails(IRLM)

Declaration

```
public HardwareRequirementsEvaluationResult<ERDPRequirementFailureReason>
CheckIfRLMMeetsHardwareRequirementsWithDetails(IRLM rlm)
```

Parameters

TYPE	NAME	DESCRIPTION
IRLM	rlm	

Returns

TYPE	DESCRIPTION
HardwareRequirementsEvaluationResult<ERDPRequirementFailureReason>	

CheckIfRLMMeetsHardwareRequirementsWithDetails(IRLM, IReadOnlyList<CameraRecord>)

Declaration

```
public HardwareRequirementsEvaluationResult<ERDPRequirementFailureReason>
CheckIfRLMMeetsHardwareRequirementsWithDetails(IRLM rlm, IReadOnlyList<CameraRecord> cameras)
```

Parameters

TYPE	NAME	DESCRIPTION
IRLM	rlm	
System.Collections.Generic.IReadOnlyList<CameraRecord>	cameras	

Returns

TYPE	DESCRIPTION
HardwareRequirementsEvaluationResult<ERDPRequirementFailureReason>	

ConfigureCameras(IReadOnlyList<CameraRecord>)

Configure the cameras returned by GetCameras()

Declaration

```
public void ConfigureCameras(IReadOnlyList<CameraRecord> cameras)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Collections.Generic.IReadOnlyList<CameraRecord>	cameras	

ConfigureFailureChance(Int32)

Configure value returned by Santec.Hardware.Devices.Polarity.ISimulatedPolarityMeter.FailureChance of the created RD-P/RD-SP.

Declaration

```
public void ConfigureFailureChance(int failureChance)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	failureChance	The configured failure chance of the created RD-P/RD-SP.

ConfigureMapping(IEnumerable<Nullable<Int32>>)

Configure value returned by Santec.Hardware.Devices.Polarity.ISimulatedPolarityMeter.PolarityMapping of the created RD-P/RD-SP.

Declaration

```
public void ConfigureMapping(IEnumerable<int?> mapping)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Collections.Generic.IEnumerable<System.Nullable<System.Int32>>	mapping	The configured mapping of the created RD-P/RD-SP.

ConfigureMappings(List<IEnumerable<Nullable<Int32>>>)

Configure value returned by Santec.Hardware.Devices.Polarity.ISimulatedRDP.PolarityMappings of the created RD-P/RD-SP.

Declaration

```
public void ConfigureMappings(List<IEnumerable<int?>> mappings)
```

Parameters

TYPE	NAME	DESCRIPTION

TYPE	NAME	DESCRIPTION
System.Collections.Generic.List<System.Collections.Generic.IEnumerable<System.Nullable<System.Int32>>>	mappings	The configured mappings of the created RD-P/RD-SP.

ConfigureRDSPCreation(Boolean)

Configure value returned by AlwaysCreateRDSP.

Declaration

```
public void ConfigureRDSPCreation(bool alwaysCreateRDSP)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Boolean	alwaysCreateRDSP	Whether the factory will always create RD-SPs from RLMs.

CreateRDP(IPTM)

Declaration

```
public IRDP CreateRDP(IPTM ptm)
```

Parameters

TYPE	NAME	DESCRIPTION
IPTM	ptm	

Returns

TYPE	DESCRIPTION
IRDP	

CreateRDP(IPTM, IReadOnlyList<CameraRecord>)

Declaration

```
public IRDP CreateRDP(IPTM ptm, IReadOnlyList<CameraRecord> cameras)
```

Parameters

TYPE	NAME	DESCRIPTION
IPTM	ptm	
System.Collections.Generic.IReadOnlyList<CameraRecord>	cameras	

Returns

TYPE	DESCRIPTION
IRDP	

CreateRDP(IRLM)

Declaration

```
public IRDP CreateRDP(IRLM rlm)
```

Parameters

TYPE	NAME	DESCRIPTION
IRLM	rlm	

Returns

TYPE	DESCRIPTION
IRDP	

CreateRDP(IRLM, IReadOnlyList<CameraRecord>)

Declaration

```
public IRDP CreateRDP(IRLM rlm, IReadOnlyList<CameraRecord> cameras)
```

Parameters

TYPE	NAME	DESCRIPTION
IRLM	rlm	
System.Collections.Generic.IReadOnlyList<CameraRecord>	cameras	

Returns

TYPE	DESCRIPTION
IRDP	

CreateRDPWithExternalSource(Int32, Int32, ELossCompensationConfiguration)

Declaration

```
public IRDP CreateRDPWithExternalSource(int wavelength, int numberOfChannels, ELossCompensationConfiguration
lossCompensationConfiguration = ELossCompensationConfiguration.RLM)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	
System.Int32	numberOfChannels	

TYPE	NAME	DESCRIPTION
ELossCompensationConfiguration	lossCompensationConfiguration	

Returns

TYPE	DESCRIPTION
IRDP	

CreateRDPWithExternalSource(Int32, Int32, IReadOnlyList<CameraRecord>, ELossCompensationConfiguration)

Declaration

```
public IRDP CreateRDPWithExternalSource(int wavelength, int numberOfChannels, IReadOnlyList<CameraRecord>
cameras, ELossCompensationConfiguration lossCompensationConfiguration = ELossCompensationConfiguration.RLM)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	
System.Int32	numberOfChannels	
System.Collections.Generic.IReadOnlyList<CameraRecord>	cameras	
ELossCompensationConfiguration	lossCompensationConfiguration	

Returns

TYPE	DESCRIPTION
IRDP	

GetCameras()

Get the available RD-P/RD-SP cameras on the system.

Declaration

```
public IReadOnlyList<CameraRecord> GetCameras()
```

Returns

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<CameraRecord>	A list of the available RD-P/RD-SP cameras on the system.

Remarks

Use ConfigureCameras(IReadOnlyList<CameraRecord>) to set this value.

Implements

IRDPFactory

Class SimulatedRDSP

Provides a simulated RD-P with configurable polarity mappings.

> **NOTE**
>
> This class is intended for testing/development purposes. It operates much quicker than an actual RD-P. It can only be created using a SimulatedRDPFactory. It should only be used when returned as part of a list of devices by a DetectHardwareAsync(EConnectionTypesToDetect) or DetectHardwareAsync<T>(EConnectionTypesToDetect) call to a SimulatedHardwareDetectionService. A default detected A polarity mapping is set but the detected mapping can be configured via the ConfigureMapping(IEnumerable<Nullable<Int32>>) method. Additionally, the chance for the polarity mapping to fail and return a different detected mapping can also be configured with the ConfigureFailureChance(Int32) method. The default failure chance is 0%.

Inheritance

System.Object
CompositeSantecDevice
SimulatedRDSP

Implements

IRDSP
IRLM
IDiscretePowerMeter
ILaserSourceWithMonitor
ILaserSource
IWavelengthDevice
ISwitchController
IFiberDevice
IPowerMeterWithReferencePersistence
IPowerMeter
IInsertionLossMeter
IDetectorDevice
IDevice<ERLMModel>
IRDP<ERDSPModel>
IRDP
ICompositeDevice
IPTM<ERDSPModel>
IPTM
ISantecDevice
IDevice<ERDSPModel>
IDevice
System.IDisposable

Inherited Members

CompositeSantecDevice.SerialNumber
CompositeSantecDevice.FirmwareVersion
CompositeSantecDevice.Address
CompositeSantecDevice.IsInitialized
CompositeSantecDevice.IsInitializing
CompositeSantecDevice.Subdevices
CompositeSantecDevice.Dispose()
CompositeSantecDevice.Dispose(Boolean)
CompositeSantecDevice.Dispose(Boolean, Boolean)
CompositeSantecDevice.SetTimeout(ETimeout)
CompositeSantecDevice.GetTimeout()
CompositeSantecDevice.EnableDHCPAsync(CancellationToken)
CompositeSantecDevice.GetGatewayAsync(CancellationToken)

CompositeSantecDevice.GetHostnameAsync(CancellationToken)
CompositeSantecDevice.GetIPAddressAsync(CancellationToken)
CompositeSantecDevice.GetSubnetPrefixLengthAsync(CancellationToken)
CompositeSantecDevice.PingAsync()
CompositeSantecDevice.ResetAsync()
CompositeSantecDevice.SetGatewayAsync(IPAddress, CancellationToken)
CompositeSantecDevice.SetHostnameAsync(String, CancellationToken)
CompositeSantecDevice.SetIPAddressAsync(IPAddress, CancellationToken)
CompositeSantecDevice.SetSubnetPrefixLengthAsync(Int32, CancellationToken)
CompositeSantecDevice.Uninitialize()
CompositeSantecDevice.Uninitialize(Boolean)
System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.Polarity

Assembly: Santec.Hardware.dll

Syntax

```
public class SimulatedRDSP : CompositeSantecDevice, IRDSP, IRLM, IDiscretePowerMeter, ILaserSourceWithMonitor,
ILaserSource, IWavelengthDevice, ISwitchController, IFiberDevice, IPowerMeterWithReferencePersistence, IPowerMeter,
IInsertionLossMeter, IDetectorDevice, IDevice<ERLMModel>, IRDP<ERDSPModel>, IRDP, ICompositeDevice,
IPTM<ERDSPModel>, IPTM, ISantecDevice, IDevice<ERDSPModel>, IDevice, IDisposable
```

Properties

Detectors

Declaration

```
public IReadOnlyList<DeviceDetector> Detectors { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<DeviceDetector>	

FailureChance

Declaration

```
public int FailureChance { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

Fiber

Declaration

```
public FiberInformation Fiber { get; }
```

Property Value

TYPE	DESCRIPTION
FiberInformation	

Model

Declaration

```
public ERDSPModel Model { get; }
```

Property Value

TYPE	DESCRIPTION
ERDSPModel	

MultiCableMappingInProgress

Declaration

```
public bool MultiCableMappingInProgress { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	

Name

Declaration

```
public override string Name { get; }
```

Property Value

TYPE	DESCRIPTION
System.String	

Overrides

CompositeSantecDevice.Name

NumberOfChannels

Declaration

```
public int NumberOfChannels { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

NumberOfDetectors

Declaration

```
public int NumberOfDetectors { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

NumberOfOutputs

Declaration

```
public int NumberOfOutputs { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

PolarityMapping

Declaration

```
public IReadOnlyList<int?> PolarityMapping { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<System.Nullable<System.Int32>>	

PolarityMappings

Declaration

```
public IReadOnlyList<IReadOnlyList<int?>> PolarityMappings { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<System.Collections.Generic.IReadOnlyList<System.Nullable<System.Int32>>>	

RequiresILReferenceSave

Declaration

```
public bool RequiresILReferenceSave { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	

SegmentedMappingInProgress

Declaration

```
public bool SegmentedMappingInProgress { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	

SupportedLengthModes

Declaration

```
public IReadOnlyList<ELengthMode> SupportedLengthModes { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<ELengthMode>	

SupportedRLModes

Declaration

```
public IReadOnlyList<ERLMode> SupportedRLModes { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<ERLMode>	

SupportsILResolution

Declaration

```
public bool SupportsILResolution { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	

SupportsMaximumResolutionRL

Declaration

```
public bool SupportsMaximumResolutionRL { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	

SupportsTurningOffLaser

Declaration

```
public bool SupportsTurningOffLaser { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	

Switches

Declaration

```
public List<DeviceSwitch> Switches { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.List<DeviceSwitch>	

Wavelength

Declaration

```
public int Wavelength { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

Wavelengths

Declaration

```
public List<int> Wavelengths { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.List<System.Int32>	

Methods

ChangeMultipleSwitchesChannelsAsync(IEnumerable<DeviceSwitchAndChannelPair>, CancellationToken)

Declaration

```
public Task ChangeMultipleSwitchesChannelsAsync(IEnumerable<DeviceSwitchAndChannelPair> deviceSwitchesAndChannels,
CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Collections.Generic.IEnumerable<DeviceSwitchAndChannelPair>	deviceSwitchesAndChannels	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

ChangeOutputChannelAsync(Int32, CancellationToken)

Declaration

```
public Task ChangeOutputChannelAsync(int output, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	output	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

ChangeSwitchChannelAsync(Int32, Int32, CancellationToken)

Declaration

```
public Task ChangeSwitchChannelAsync(int switchIndex, int channel, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	switchIndex	
System.Int32	channel	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

CollectOutputMappingDataAsync(Int32, Int32, CancellationToken)

Declaration

```
public Task<bool> CollectOutputMappingDataAsync(int cable, int fiber, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	cable	

TYPE	NAME	DESCRIPTION
System.Int32	fiber	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Boolean>	

CollectOutputMappingDataAsync(Int32, CancellationToken)

Declaration

```
public Task<bool> CollectOutputMappingDataAsync(int channel, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	channel	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Boolean>	

ConfigureFailureChance(Int32)

Declaration

```
public void ConfigureFailureChance(int failureChance)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	failureChance	

ConfigureMapping(IEnumerable<Nullable<Int32>>)

Declaration

```
public void ConfigureMapping(IEnumerable<int?> mapping)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Collections.Generic.IEnumerable<System.Nullable<System.Int32>>	mapping	

ConfigureMappings(IEnumerable<IEnumerable<Nullable<Int32>>>)

Declaration

```
public void ConfigureMappings(IEnumerable<IEnumerable<int?>> mappings)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Collections.Generic.IEnumerable<System.Collections.Generic.IEnumerable<System.Nullable<System.Int32>>>	mappings	

EndMappingPolarityAsync()

Declaration

```
public Task EndMappingPolarityAsync()
```

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

GetActiveLaserAsync(CancellationToken)

Declaration

```
public Task<int> GetActiveLaserAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Int32>	

GetCameraMode()

Declaration

```
public ECameraMode GetCameraMode()
```

Returns

TYPE	DESCRIPTION
ECameraMode	

GetCustomCameraSettings()

Declaration

```
public LegacyCustomCameraSettings GetCustomCameraSettings()
```

Returns

TYPE	DESCRIPTION
LegacyCustomCameraSettings	

GetDUTILAsync(Int32, CancellationToken)

Declaration

```
public Task<double> GetDUTILAsync(int wavelength, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

GetFactoryPowerAsync(Int32, Int32, CancellationToken)

Declaration

```
public Task<FactoryPowerReading> GetFactoryPowerAsync(int output, int wavelength, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	output	
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<FactoryPowerReading>	

GetGainModeAsync(CancellationToken)

Declaration

```
public Task<EGainMode> GetGainModeAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<EGainMode>	

GetILReferenceAsync(Int32, Int32, CancellationToken)

Declaration

```
public Task<double> GetILReferenceAsync(int detector, int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

GetILResolutionAsync(CancellationToken)

Declaration

```
public Task<EResolution> GetILResolutionAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<EResolution>	

GetInteractionModeAsync(CancellationToken)

Declaration

```
public override Task<EInteractionMode> GetInteractionModeAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<EInteractionMode>	

Overrides

CompositeSantecDevice.GetInteractionModeAsync(CancellationToken)

GetLengthModeAsync(CancellationToken)

Declaration

```
public Task<ELengthMode> GetLengthModeAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<ELengthMode>	

GetLossCompensation()

Declaration

```
public int GetLossCompensation()
```

Returns

TYPE	DESCRIPTION
System.Int32	

GetMappingInProgress()

Declaration

```
public bool GetMappingInProgress()
```

Returns

TYPE	DESCRIPTION
System.Boolean	

GetMappingMode()

Declaration

```
public EMappingMode GetMappingMode()
```

Returns

TYPE	DESCRIPTION
EMappingMode	

GetMappingSensitivity()

Declaration

```
public EMappingSensitivity GetMappingSensitivity()
```

Returns

TYPE	DESCRIPTION
EMappingSensitivity	

GetOutputChannelAsync(CancellationToken)

Declaration

```
public Task<int> GetOutputChannelAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Int32>	

GetOutputMappingDataCollected(Int32)

Declaration

```
public bool GetOutputMappingDataCollected(int channel)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	channel	

Returns

TYPE	DESCRIPTION
System.Boolean	

GetOutputMappingDataCollected(Int32, Int32)

Declaration

```
public bool GetOutputMappingDataCollected(int cable, int fiber)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	cable	
System.Int32	fiber	

Returns

TYPE	DESCRIPTION
System.Boolean	

GetOutputMappingImages(Int32)

Declaration

```
public IReadOnlyList<Mat> GetOutputMappingImages(int channel)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	channel	

Returns

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<OpenCvSharp.Mat>	

GetOutputMappingImages(Int32, Int32)

Declaration

```
public IReadOnlyList<Mat> GetOutputMappingImages(int cable, int fiber)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	cable	
System.Int32	fiber	

Returns

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<OpenCvSharp.Mat>	

GetPositionBModeAsync(CancellationToken)

Declaration

```
public Task<EPositionBMode> GetPositionBModeAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<EPositionBMode>	

GetRLModeAsync(CancellationToken)

Declaration

```
public Task<ERLMode> GetRLModeAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<ERLMode>	

GetRLReferenceAsync(CancellationToken)

Declaration

```
public Task<double> GetRLReferenceAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

GetSwitchChannelAsync(Int32, CancellationToken)

Declaration

```
public Task<int> GetSwitchChannelAsync(int switchIndex, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	switchIndex	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Int32>	

GetTraceDiagnosticsAsync(CancellationToken)

Declaration

```
public Task<TraceDiagnostics> GetTraceDiagnosticsAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<TraceDiagnostics>	

InitializeAsync()

Declaration

```
public override async Task InitializeAsync()
```

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Overrides

CompositeSantecDevice.InitializeAsync()

Exceptions

TYPE	CONDITION
CompositeDeviceHardwareException	Thrown when the component RLM no longer meets the RD-P hardware requirements.

MapOutputAsync(Int32, CancellationToken)

> **⚠ CAUTION**
>
> This operation is not supported for RD-P devices. This operation will always throw a System.NotSupportedException.

Declaration

```
public Task<int?> MapOutputAsync(int channel, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	channel	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Nullable<System.Int32>>	

MapPolarityAsync(Int32, IEnumerable<Nullable<Int32>>, CancellationToken)

Declaration

```
public Task<IReadOnlyList<int?>> MapPolarityAsync(int numberOfChannels, IEnumerable<int?> expectedMapping,
CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	numberOfChannels	
System.Collections.Generic.IEnumerable<System.Nullable<System.Int32>>	expectedMapping	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Collections.Generic.IReadOnlyList<System.Nullable<System.Int32>>>	

MapPolarityAsync(Int32, CancellationToken)

Declaration

```
public Task<IReadOnlyList<int?>> MapPolarityAsync(int numberOfChannels, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	numberOfChannels	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Collections.Generic.IReadOnlyList<System.Nullable<System.Int32>>>	

ProcessMappingData(IEnumerable<Nullable<Int32>>)

Declaration

```
public IReadOnlyList<int?> ProcessMappingData(IEnumerable<int?> expectedMapping = null)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Collections.Generic.IEnumerable<System.Nullable<System.Int32>>	expectedMapping	

Returns

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<System.Nullable<System.Int32>>	

ProcessMappingData(Int32, IEnumerable<Nullable<Int32>>)

Declaration

```
public IReadOnlyList<int?> ProcessMappingData(int cable, IEnumerable<int?> expectedMapping = null)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	cable	
System.Collections.Generic.IEnumerable<System.Nullable<System.Int32>>	expectedMapping	

Returns

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<System.Nullable<System.Int32>>	

QueryAsync(String, CancellationToken)

Declaration

```
public override Task<string> QueryAsync(string command, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	command	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.String>	

Overrides

CompositeSantecDevice.QueryAsync(String, CancellationToken)

ReadAsync(CancellationToken)

Declaration

```
public override Task<string> ReadAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.String>	

Overrides

CompositeSantecDevice.ReadAsync(CancellationToken)

ReadILAsync(Int32, Int32, CancellationToken)

Declaration

```
public Task<double> ReadILAsync(int detector, int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

ReadMonitorPowerAsync(Int32, CancellationToken)

Declaration

```
public Task<double> ReadMonitorPowerAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

ReadMultipleILsAsync(IEnumerable<Int32>, Int32, CancellationToken)

Declaration

```
public Task<List<double>> ReadMultipleILsAsync(IEnumerable<int> detectors, int wavelength, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Collections.Generic.IEnumerable<System.Int32>	detectors	
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Collections.Generic.List<System.Double>>	

ReadMultiplePowersAsync(IEnumerable<Int32>, Int32, CancellationToken)

Declaration

```
public Task<List<double>> ReadMultiplePowersAsync(IEnumerable<int> detectors, int wavelength, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Collections.Generic.IEnumerable<System.Int32>	detectors	
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Collections.Generic.List<System.Double>>	

ReadPowerAsync(Int32, Int32, CancellationToken)

Declaration

```
public Task<double> ReadPowerAsync(int detector, int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

ReadRLAsync(Int32, Double, Double, CancellationToken)

Declaration

```
public Task<RLReading> ReadRLAsync(int wavelength, double lengthReferenceA, double lengthReferenceB,
CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	
System.Double	lengthReferenceA	
System.Double	lengthReferenceB	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<RLReading>	

ReadRLAsync(Int32, Double, CancellationToken)

Declaration

```
public Task<RLReading> ReadRLAsync(int wavelength, double lengthReference, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	
System.Double	lengthReference	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<RLReading>	

ReadRLAsync(Int32, CancellationToken)

Declaration

```
public Task<RLReading> ReadRLAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<RLReading>	

ReferenceILAsync(Int32, Int32, Boolean, CancellationToken)

Declaration

```
public Task<double> ReferenceILAsync(int detector, int wavelength, bool applyReferenceToAllDetectors,
CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Int32	wavelength	
System.Boolean	applyReferenceToAllDetectors	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

ReferenceILAsync(Int32, Int32, CancellationToken)

Declaration

```
public Task<double> ReferenceILAsync(int detector, int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

ReferenceRLAsync(CancellationToken)

Declaration

```
public Task<double> ReferenceRLAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

RefreshAsync()

Declaration

```
public override async Task RefreshAsync()
```

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Overrides

CompositeSantecDevice.RefreshAsync()

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized RD-P.
CompositeDeviceHardwareException	Thrown when the component RLM no longer meets the RD-P hardware requirements.
System.InvalidOperationException	Thrown when the RLM subdevice is uninitialized when the method is called.
UnexpectedDeviceResponseException	Thrown when the RLM subdevice gives an unexpected response during its refresh operation.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the RLM subdevice during its refresh operation.

SaveILReferencesAsync(CancellationToken)

Declaration

```
public Task SaveILReferencesAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetCameraModeAsync(ECameraMode)

Declaration

```
public Task SetCameraModeAsync(ECameraMode cameraMode)
```

Parameters

TYPE	NAME	DESCRIPTION
ECameraMode	cameraMode	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetCustomCameraSettingsAsync(Int32, Int32)

Declaration

```
public Task SetCustomCameraSettingsAsync(int exposure, int gain)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	exposure	
System.Int32	gain	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetDUTILAsync(Int32, Double, CancellationToken)

Declaration

```
public Task SetDUTILAsync(int wavelength, double il, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	
System.Double	il	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetGainModeAsync(EGainMode, CancellationToken)

Declaration

```
public Task SetGainModeAsync(EGainMode gainMode, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
EGainMode	gainMode	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetILReferenceAsync(Int32, Int32, Double, CancellationToken)

Declaration

```
public Task SetILReferenceAsync(int detector, int wavelength, double reference, CancellationToken cancellationToken
= default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Int32	wavelength	
System.Double	reference	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetILResolutionAsync(EResolution, CancellationToken)

Declaration

```
public Task SetILResolutionAsync(EResolution resolution, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
EResolution	resolution	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetInteractionModeAsync(EInteractionMode, CancellationToken)

Declaration

```
public override Task SetInteractionModeAsync(EInteractionMode interactionMode, CancellationToken cancellationToken
= default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
EInteractionMode	interactionMode	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Overrides

CompositeSantecDevice.SetInteractionModeAsync(EInteractionMode, CancellationToken)

SetLengthModeAsync(ELengthMode, CancellationToken)

Declaration

```
public Task SetLengthModeAsync(ELengthMode lengthMode, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
ELengthMode	lengthMode	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetLossCompensationAsync(Int32)

Declaration

```
public Task SetLossCompensationAsync(int inputPower)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	inputPower	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetMappingMode(EMappingMode)

Declaration

```
public void SetMappingMode(EMappingMode mappingMode)
```

Parameters

TYPE	NAME	DESCRIPTION
EMappingMode	mappingMode	

SetMappingSensitivity(EMappingSensitivity)

Declaration

```
public void SetMappingSensitivity(EMappingSensitivity mappingSensitivity)
```

Parameters

TYPE	NAME	DESCRIPTION
EMappingSensitivity	mappingSensitivity	

SetPositionBModeAsync(EPositionBMode, CancellationToken)

Declaration

```
public Task SetPositionBModeAsync(EPositionBMode positionBMode, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
EPositionBMode	positionBMode	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetRLModeAsync(ERLMode, CancellationToken)

Declaration

```
public Task SetRLModeAsync(ERLMode rlMode, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
ERLMode	rlMode	

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetRLReferenceAsync(Double, CancellationToken)

Declaration

```
public Task SetRLReferenceAsync(double reference, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Double	reference	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

StartMappingPolarityAsync(IEnumerable<CableInformation>)

Declaration

```
public Task StartMappingPolarityAsync(IEnumerable<CableInformation> cables)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Collections.Generic.IEnumerable<CableInformation>	cables	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

StartMappingPolarityAsync(Int32)

Declaration

```
public Task StartMappingPolarityAsync(int numberOfChannels)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	numberOfChannels	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

TurnOffLaserAsync(CancellationToken)

> **⚠ CAUTION**
>
> This operation is not supported for RDP devices. This operation will always throw a System.NotSupportedException.

Declaration

```
public Task TurnOffLaserAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

TurnOnFlashingVFLModeAsync(Int32, CancellationToken)

> **⚠ CAUTION**
>
> This operation is not supported for RDP devices. This operation will always throw a System.NotSupportedException.

Declaration

```
public Task TurnOnFlashingVFLModeAsync(int channel, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	channel	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

TurnOnLaserAsync(Int32, CancellationToken)

Declaration

```
public Task TurnOnLaserAsync(int wavelength, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

TurnOnVFLModeAsync(Int32, CancellationToken)

> **⚠ CAUTION**
>
> This operation is not supported for RDP devices. This operation will always throw a System.NotSupportedException.

Declaration

```
public Task TurnOnVFLModeAsync(int channel, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	channel	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

WriteAsync(String, CancellationToken)

Declaration

```
public override Task WriteAsync(string command, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	command	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Overrides

CompositeSantecDevice.WriteAsync(String, CancellationToken)

Explicit Interface Implementations

IDetectorDevice.Detectors

Declaration

```
IReadOnlyList<IModule> IDetectorDevice.Detectors { get; }
```

Returns

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<IModule>	

IDevice<ERLMModel>.Model

Declaration

```
ERLMModel IDevice<ERLMModel>.Model { get; }
```

Returns

TYPE	DESCRIPTION
ERLMModel	

Implements

IRDSP
IRLM
IDiscretePowerMeter
ILaserSourceWithMonitor
ILaserSource
IWavelengthDevice
ISwitchController
IFiberDevice
IPowerMeterWithReferencePersistence
IPowerMeter
IInsertionLossMeter
IDetectorDevice
IDevice<TModelType>
IRDP<TModelType>
IRDP
ICompositeDevice
IPTM<TModelType>
IPTM
ISantecDevice
IDevice<TModelType>
IDevice
System.IDisposable

Extension Methods

RLMExtensions.IsMultimodeRLM100(IRLM)

Namespace Santec.Hardware.Devices.Polarization

Classes

PCU

Provides a connection to a PCU along with properties of the device and operations that can be performed using it.

> **NOTE**
>
> This class cannot be used directly. It can only be used when returned:
>
> 1. As part of a list of devices by a DetectHardwareAsync(EConnectionTypesToDetect) or DetectHardwareAsync<T>(EConnectionTypesToDetect) call.
> 2. As a device by a TryDetectIPDeviceAsync(IPAddress) or TryDetectIPDeviceAsync<T>(IPAddress) call.

PDLAndBRReading

Represents the results of a PDL and BR reading.

> **NOTE**
>
> This class should not be used directly. It should only be used when returned by a PDL and BR measurement.

PDLReading

Represents the results of a PDL reading.

> **NOTE**
>
> This class should not be used directly. It should only be used when returned by a PDL measurement.

PEM

Provides a connection to a PEM along with properties of the device and operations that can be performed using it.

> **NOTE**
>
> This class cannot not be used directly. It can only be used when returned:
>
> 1. As part of a list of devices by a DetectHardwareAsync(EConnectionTypesToDetect) or DetectHardwareAsync<T>(EConnectionTypesToDetect) call.
> 2. As a device by a TryDetectIPDeviceAsync(IPAddress) or TryDetectIPDeviceAsync<T>(IPAddress) call.

PERReading

Represents the results of a PER reading.

> **NOTE**
>
> This class should not be used directly. It should only be used when returned by a PER measurement.

PLM

Provides a connection to a PLM along with properties of the device and operations that can be performed using it.

NOTE

This class cannot not be used directly. It can only be used when returned:

1. As part of a list of devices by a DetectHardwareAsync(EConnectionTypesToDetect) or DetectHardwareAsync<T>(EConnectionTypesToDetect) call.
2. As a device by a TryDetectIPDeviceAsync(IPAddress) or TryDetectIPDeviceAsync<T>(IPAddress) call.

SimulatedPCU

Provides a simulated PCU.

NOTE

This class is intended for testing/development purposes. It operates much quicker than an actual PCU. It can only be created using a SimulatedDeviceFactory. It should only be used when returned as part of a list of devices by a DetectHardwareAsync(EConnectionTypesToDetect) or DetectHardwareAsync<T>(EConnectionTypesToDetect) call to a SimulatedHardwareDetectionService.

SimulatedPEM

Provides a simulated PEM with configurable reading values.

NOTE

This class is intended for testing/development purposes. It operates much quicker than an actual PEM. It can only be created using a SimulatedDeviceFactory. It should only be used when returned as part of a list of devices by a DetectHardwareAsync(EConnectionTypesToDetect) or DetectHardwareAsync<T>(EConnectionTypesToDetect) call to a SimulatedHardwareDetectionService. Each reading value contains a default value range but can be configured via the appropriate configuration method.

SimulatedPLM

Provides a simulated PLM with configurable reading values.

NOTE

This class is intended for testing/development purposes. It operates much quicker than an actual PLM. It can only be created using a SimulatedDeviceFactory. It should only be used when returned as part of a list of devices by a DetectHardwareAsync(EConnectionTypesToDetect) or DetectHardwareAsync<T>(EConnectionTypesToDetect) call to a SimulatedHardwareDetectionService. Each reading value contains a default value range but can be configured via the appropriate configuration method.

Interfaces

IPCU

Defines the properties of a PCU and the operations that can be performed with a connection to the device.

IPEM

Defines the properties of a PEM and operations that can be performed with a connection to the device.

IPLM

Defines the properties of a PLM and the operations that can be performed with a connection to the device.

IPolarizationController

Defines the operations that can be performed with a polarization controller.

Enums

EAutoPowerAdjustmentState

Specifies the auto power adjustment state of a PEM.

EPCUModel

Specifies the model of a PCU.

EPEMModel

Specifies the model of a PEM.

EPERMode

Specifies the PER mode of a PEM.

EPLMModel

Specifies the model of a PLM.

EPolarizationMode

Specifies the polarization mode of a PLM.

EPolarizationState

Specifies the polarization state of a polarization controller.

Enum EAutoPowerAdjustmentState

Specifies the auto power adjustment state of a PEM.

Namespace: Santec.Hardware.Devices.Polarization

Assembly: Santec.Hardware.dll

Syntax

```
public enum EAutoPowerAdjustmentState
```

Fields

NAME	DESCRIPTION
Off	Auto power adjustment off.
On	Auto power adjustment on.

Enum EPCUModel

Specifies the model of a PCU.

Namespace: Santec.Hardware.Devices.Polarization

Assembly: Santec.Hardware.dll

Syntax

```
public enum EPCUModel
```

Fields

NAME	DESCRIPTION
PCU200	The PCU-200 model.

Enum EPEMModel

Specifies the model of a PEM.

Namespace: Santec.Hardware.Devices.Polarization

Assembly: Santec.Hardware.dll

Syntax

```
public enum EPEMModel
```

Fields

NAME	DESCRIPTION
PEM400	The PEM-400 model.

Enum EPERMode

Specifies the PER mode of a PEM.

Namespace: Santec.Hardware.Devices.Polarization

Assembly: Santec.Hardware.dll

Syntax

```
public enum EPERMode
```

Fields

NAME	DESCRIPTION
Fast	Fast PER mode.
Standard	Standard PER mode

Enum EPLMModel

Specifies the model of a PLM.

Namespace: Santec.Hardware.Devices.Polarization

Assembly: Santec.Hardware.dll

Syntax

```
public enum EPLMModel
```

Fields

NAME	DESCRIPTION
PLM100	The PLM-100 model.

Enum EPolarizationMode

Specifies the polarization mode of a PLM.

Namespace: Santec.Hardware.Devices.Polarization

Assembly: Santec.Hardware.dll

Syntax

```
public enum EPolarizationMode
```

Fields

NAME	DESCRIPTION
Fast	Fast mode, using 4 states.
Standard	Standard mode, using 6 states.

Enum EPolarizationState

Specifies the polarization state of a polarization controller.

Namespace: Santec.Hardware.Devices.Polarization

Assembly: Santec.Hardware.dll

Syntax

```
public enum EPolarizationState
```

Fields

NAME	DESCRIPTION
Horizontal	Horizontal polarization state.
LeftHandCircular	Left-handed circular polarization state.
Negative45	-45 degrees polarization state.
Positive45	+45 degrees polarization state.
RightHandCircular	Right-handed circular polarization state.
Unknown	Unknown custom polarization state.
Vertical	Vertical polarization state.

Interface IPCU

Defines the properties of a PCU and the operations that can be performed with a connection to the device.

Inherited Members

ISantecDevice.GetIPAddressAsync(CancellationToken)
ISantecDevice.SetIPAddressAsync(IPAddress, CancellationToken)
ISantecDevice.EnableDHCPAsync(CancellationToken)
ISantecDevice.GetGatewayAsync(CancellationToken)
ISantecDevice.SetGatewayAsync(IPAddress, CancellationToken)
ISantecDevice.GetSubnetPrefixLengthAsync(CancellationToken)
ISantecDevice.SetSubnetPrefixLengthAsync(Int32, CancellationToken)
ISantecDevice.GetHostnameAsync(CancellationToken)
ISantecDevice.SetHostnameAsync(String, CancellationToken)
ISantecDevice.GetInteractionModeAsync(CancellationToken)
ISantecDevice.SetInteractionModeAsync(EInteractionMode, CancellationToken)
IPolarizationController.GetPolarizationStateAsync(CancellationToken)
IPolarizationController.SetPolarizationStateAsync(EPolarizationState, CancellationToken)
IPolarizationController.GetWaveplatePositionAsync(Int32, CancellationToken)
IPolarizationController.SetWaveplatePositionAsync(Int32, Double, CancellationToken)
IDeviceWithInputs.NumberOfInputs
IDeviceWithInputs.GetInputChannelAsync(CancellationToken)
IDeviceWithInputs.ChangeInputChannelAsync(Int32, CancellationToken)
ISwitchController.Switches
ISwitchController.GetSwitchChannelAsync(Int32, CancellationToken)
ISwitchController.ChangeSwitchChannelAsync(Int32, Int32, CancellationToken)
ISwitchController.ChangeMultipleSwitchesChannelsAsync(IEnumerable<DeviceSwitchAndChannelPair>, CancellationToken)
IDevice<EPCUModel>.Model
IDevice.Name
IDevice.SerialNumber
IDevice.FirmwareVersion
IDevice.Address
IDevice.IsInitialized
IDevice.InitializeAsync()
IDevice.Uninitialize()
IDevice.GetTimeout()
IDevice.SetTimeout(ETimeout)
IDevice.ResetAsync()
IDevice.RefreshAsync()
IDevice.PingAsync()
IDevice.QueryAsync(String, CancellationToken)
IDevice.WriteAsync(String, CancellationToken)
IDevice.ReadAsync(CancellationToken)
System.IDisposable.Dispose()

Namespace: Santec.Hardware.Devices.Polarization

Assembly: Santec.Hardware.dll

Syntax

```
public interface IPCU : ISantecDevice, IPolarizationController, IDeviceWithInputs, ISwitchController,
IDevice<EPCUModel>, IDevice, IDisposable
```

Interface IPEM

Defines the properties of a PEM and operations that can be performed with a connection to the device.

Inherited Members

ISantecDevice.GetIPAddressAsync(CancellationToken)
ISantecDevice.SetIPAddressAsync(IPAddress, CancellationToken)
ISantecDevice.EnableDHCPAsync(CancellationToken)
ISantecDevice.GetGatewayAsync(CancellationToken)
ISantecDevice.SetGatewayAsync(IPAddress, CancellationToken)
ISantecDevice.GetSubnetPrefixLengthAsync(CancellationToken)
ISantecDevice.SetSubnetPrefixLengthAsync(Int32, CancellationToken)
ISantecDevice.GetHostnameAsync(CancellationToken)
ISantecDevice.SetHostnameAsync(String, CancellationToken)
ISantecDevice.GetInteractionModeAsync(CancellationToken)
ISantecDevice.SetInteractionModeAsync(EInteractionMode, CancellationToken)
IInsertionLossMeter.GetILReferenceAsync(Int32, Int32, CancellationToken)
IInsertionLossMeter.SetILReferenceAsync(Int32, Int32, Double, CancellationToken)
IInsertionLossMeter.ReferenceILAsync(Int32, Int32, CancellationToken)
IInsertionLossMeter.ReferenceILAsync(Int32, Int32, Boolean, CancellationToken)
IInsertionLossMeter.ReadILAsync(Int32, Int32, CancellationToken)
ILaserSource.NumberOfOutputs
ILaserSource.SupportsTurningOffLaser
ILaserSource.GetActiveLaserAsync(CancellationToken)
ILaserSource.TurnOnLaserAsync(Int32, CancellationToken)
ILaserSource.TurnOffLaserAsync(CancellationToken)
ILaserSource.GetOutputChannelAsync(CancellationToken)
ILaserSource.ChangeOutputChannelAsync(Int32, CancellationToken)
IWavelengthDevice.Wavelengths
IDetectorDeviceWithDarking.DarkAllDetectorsAsync(CancellationToken)
IDetectorDeviceWithDarking.DarkDetectorAsync(Int32, CancellationToken)
IDetectorDeviceWithDarking.GetDetectorDarkStateAsync(Int32, CancellationToken)
IDetectorDevice.NumberOfDetectors
IDetectorDevice.Detectors
ISwitchController.Switches
ISwitchController.GetSwitchChannelAsync(Int32, CancellationToken)
ISwitchController.ChangeSwitchChannelAsync(Int32, Int32, CancellationToken)
ISwitchController.ChangeMultipleSwitchesChannelsAsync(IEnumerable<DeviceSwitchAndChannelPair>, CancellationToken)
IFiberDevice.Fiber
IDevice<EPEMModel>.Model
IDevice.Name
IDevice.SerialNumber
IDevice.FirmwareVersion
IDevice.Address
IDevice.IsInitialized
IDevice.InitializeAsync()
IDevice.Uninitialize()
IDevice.GetTimeout()
IDevice.SetTimeout(ETimeout)
IDevice.ResetAsync()
IDevice.RefreshAsync()
IDevice.PingAsync()

IDevice.QueryAsync(String, CancellationToken)
IDevice.WriteAsync(String, CancellationToken)
IDevice.ReadAsync(CancellationToken)
System.IDisposable.Dispose()

Namespace: Santec.Hardware.Devices.Polarization

Assembly: Santec.Hardware.dll

Syntax

```
public interface IPEM : ISantecDevice, IInsertionLossMeter, ILaserSource, IWavelengthDevice,
IDetectorDeviceWithDarking, IDetectorDevice, ISwitchController, IFiberDevice, IDevice<EPEMModel>, IDevice,
IDisposable
```

Methods

GetAutoPowerAdjustmentStateAsync(CancellationToken)

Asynchronously get the auto power adjustment state.

Declaration

```
Task<EAutoPowerAdjustmentState> GetAutoPowerAdjustmentStateAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<EAutoPowerAdjustmentState>	

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized PEM.
UnexpectedDeviceResponseException	Thrown when the PEM auto power adjustment state response does not correspond to an auto power adjustment state.
DeviceTimeoutException	Thrown when a communication failure causes no response to be received from the PEM.

GetPERModeAsync(CancellationToken)

Asynchronously get the PER sensitivity mode.

Declaration

```
Task<EPERMode> GetPERModeAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<EPERMode>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized PEM.
UnexpectedDeviceResponseException	Thrown when the device response does not correspond to a PER mode.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the PEM.

ReadMaximumPowerAsync(Int32, Int32, CancellationToken)

Asynchronously read the maximum power from a PER measurement for a detector at a given wavelength.

Declaration

```
Task<double> ReadMaximumPowerAsync(int detector, int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION

TYPE	NAME	DESCRIPTION
System.Int32	detector	The detector number of the detector to read the maximum power from.
System.Int32	wavelength	The wavelength to read the maximum power at.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	The task representing the asynchronous operation.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the PEM does not support the specified wavelength.
System.ArgumentException	Thrown when the specified detector is out of range for the PEM.
System.InvalidOperationException	Thrown when the method is called on an uninitialized PEM.
UnexpectedDeviceResponseException	Thrown when the device read maximum power response is not a valid number.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the PEM.

ReadMinimumPowerAsync(Int32, Int32, CancellationToken)

Asynchronously read the minimum power from a PER measurement for a detector at a given wavelength.

Declaration

```
Task<double> ReadMinimumPowerAsync(int detector, int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	The detector number of the detector to read the minimum power from.

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	The wavelength to read the minimum power at.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	The task representing the asynchronous operation.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the PEM does not support the specified wavelength.
System.ArgumentException	Thrown when the specified detector is out of range for the PEM.
System.InvalidOperationException	Thrown when the method is called on an uninitialized PEM.
UnexpectedDeviceResponseException	Thrown when the device read minimum power response is not a valid number.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the PEM.

ReadPERAsync(Int32, Int32, CancellationToken)

Asynchronously read the polarization extinction ratio from a device detector at a given wavelength.

Declaration

```
Task<PERReading> ReadPERAsync(int detector, int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	The detector number of the detector to read from.
System.Int32	wavelength	The wavelength to read the polarization extinction ratio at.

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<PERReading>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the device does not support the specified wavelength.
System.ArgumentException	Thrown when the specified detector is out of range for the device.
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device read PER response does not match the expected format.
UnexpectedDeviceResponseException	Thrown when any of the PER reading values are not valid response values.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

SetAutoPowerAdjustmentStateAsync(EAutoPowerAdjustmentState, CancellationToken)

Asynchronously change the auto power adjustment state.

Declaration

```
Task SetAutoPowerAdjustmentStateAsync(EAutoPowerAdjustmentState autoPowerAdjustmentState, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
EAutoPowerAdjustmentState	autoPowerAdjustmentState	The state to set the auto power adjustment to.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized PEM.
UnexpectedDeviceResponseException	Thrown when the PEM auto power adjustment state response does not match the expected response.
DeviceTimeoutException	Thrown when a communication failure causes no response to be received from the PEM.

SetPERModeAsync(EPERMode, CancellationToken)

Asynchronously set the PER sensitivity mode.

Declaration

```
Task SetPERModeAsync(EPERMode perMode, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
EPERMode	perMode	The PER mode to change to.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized PEM.
UnexpectedDeviceResponseException	Thrown when the device PER mode response does not match the expected response.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the PEM.

Interface IPLM

Defines the properties of a PLM and the operations that can be performed with a connection to the device.

Inherited Members

ISantecDevice.GetIPAddressAsync(CancellationToken)
ISantecDevice.SetIPAddressAsync(IPAddress, CancellationToken)
ISantecDevice.EnableDHCPAsync(CancellationToken)
ISantecDevice.GetGatewayAsync(CancellationToken)
ISantecDevice.SetGatewayAsync(IPAddress, CancellationToken)
ISantecDevice.GetSubnetPrefixLengthAsync(CancellationToken)
ISantecDevice.SetSubnetPrefixLengthAsync(Int32, CancellationToken)
ISantecDevice.GetHostnameAsync(CancellationToken)
ISantecDevice.SetHostnameAsync(String, CancellationToken)
ISantecDevice.GetInteractionModeAsync(CancellationToken)
ISantecDevice.SetInteractionModeAsync(EInteractionMode, CancellationToken)
IDiscretePowerMeter.ReadMultiplePowersAsync(IEnumerable<Int32>, Int32, CancellationToken)
IDiscretePowerMeter.ReadMultipleILsAsync(IEnumerable<Int32>, Int32, CancellationToken)
IPowerMeter.ReadPowerAsync(Int32, Int32, CancellationToken)
IInsertionLossMeter.GetILReferenceAsync(Int32, Int32, CancellationToken)
IInsertionLossMeter.SetILReferenceAsync(Int32, Int32, Double, CancellationToken)
IInsertionLossMeter.ReferenceILAsync(Int32, Int32, CancellationToken)
IInsertionLossMeter.ReferenceILAsync(Int32, Int32, Boolean, CancellationToken)
IInsertionLossMeter.ReadILAsync(Int32, Int32, CancellationToken)
ILaserSourceWithMonitor.ReadMonitorPowerAsync(Int32, CancellationToken)
ILaserSourceWithMonitor.GetFactoryPowerAsync(Int32, Int32, CancellationToken)
ILaserSource.NumberOfOutputs
ILaserSource.SupportsTurningOffLaser
ILaserSource.GetActiveLaserAsync(CancellationToken)
ILaserSource.TurnOnLaserAsync(Int32, CancellationToken)
ILaserSource.TurnOffLaserAsync(CancellationToken)
ILaserSource.GetOutputChannelAsync(CancellationToken)
ILaserSource.ChangeOutputChannelAsync(Int32, CancellationToken)
ISwitchController.Switches
ISwitchController.GetSwitchChannelAsync(Int32, CancellationToken)
ISwitchController.ChangeSwitchChannelAsync(Int32, Int32, CancellationToken)
ISwitchController.ChangeMultipleSwitchesChannelsAsync(IEnumerable<DeviceSwitchAndChannelPair>, CancellationToken)
IBackreflectionMeter.DarkBRDetectorAsync(CancellationToken)
IBackreflectionMeter.GetBRReferenceAsync(Int32, CancellationToken)
IBackreflectionMeter.SetBRReferenceAsync(Int32, Double, CancellationToken)
IBackreflectionMeter.ReferenceBRAsync(Int32, CancellationToken)
IBackreflectionMeter.ReadBRAsync(Int32, CancellationToken)
IWavelengthDevice.Wavelengths
IDetectorDeviceWithDarking.DarkAllDetectorsAsync(CancellationToken)
IDetectorDeviceWithDarking.DarkDetectorAsync(Int32, CancellationToken)
IDetectorDeviceWithDarking.GetDetectorDarkStateAsync(Int32, CancellationToken)
IDetectorDevice.NumberOfDetectors
IDetectorDevice.Detectors
IFiberDevice.Fiber
IPolarizationController.GetPolarizationStateAsync(CancellationToken)
IPolarizationController.SetPolarizationStateAsync(EPolarizationState, CancellationToken)
IPolarizationController.GetWaveplatePositionAsync(Int32, CancellationToken)

IPolarizationController.SetWaveplatePositionAsync(Int32, Double, CancellationToken)
IDevice<EPLMModel>.Model
IDevice.Name
IDevice.SerialNumber
IDevice.FirmwareVersion
IDevice.Address
IDevice.IsInitialized
IDevice.InitializeAsync()
IDevice.Uninitialize()
IDevice.GetTimeout()
IDevice.SetTimeout(ETimeout)
IDevice.ResetAsync()
IDevice.RefreshAsync()
IDevice.PingAsync()
IDevice.QueryAsync(String, CancellationToken)
IDevice.WriteAsync(String, CancellationToken)
IDevice.ReadAsync(CancellationToken)
System.IDisposable.Dispose()

Namespace: Santec.Hardware.Devices.Polarization

Assembly: Santec.Hardware.dll

Syntax

```
public interface IPLM : ISantecDevice, IDiscretePowerMeter, IPowerMeter, IInsertionLossMeter,
ILaserSourceWithMonitor, ILaserSource, ISwitchController, IBackreflectionMeter, IWavelengthDevice,
IDetectorDeviceWithDarking, IDetectorDevice, IFiberDevice, IPolarizationController, IDevice<EPLMModel>,
IDevice, IDisposable
```

Methods

GetPDLReferencesAsync(Int32, Int32, CancellationToken)

Asynchronously get the stored PDL reference values for a detector at a given wavelength.

Declaration

```
Task<List<double>> GetPDLReferencesAsync(int detector, int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	The detector to get the reference values for.
System.Int32	wavelength	The wavelength to get the reference values for.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Collections.Generic.List<System.Double>>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the PLM does not support the specified wavelength.
System.ArgumentException	Thrown when the specified detector is out of range for the PLM.
System.InvalidOperationException	Thrown when the method is called on an uninitialized PLM.
System.InvalidOperationException	Thrown when a PDL reference has not been performed for the specified detector and wavelength.
UnexpectedDeviceResponseException	Thrown when the PLM PDL references query response is not a valid list of values.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the PLM.

GetPolarizationModeAsync(CancellationToken)

Asynchronously get the polarization mode.

Declaration

```
Task<EPolarizationMode> GetPolarizationModeAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<EPolarizationMode>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized PLM.
UnexpectedDeviceResponseException	Thrown when the PLM response does not correspond to a polarization mode.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the PLM.

ReadPDLAndBRAsync(Int32, Int32, CancellationToken)

Asynchronously read the polarization dependent loss from a detector at a given wavelength and the backreflection at the wavelength.

Declaration

```
Task<PDLAndBRReading> ReadPDLAndBRAsync(int detector, int wavelength, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	The detector to read the polarization dependent loss for.
System.Int32	wavelength	The wavelength to read the polarization dependent loss and backreflection at.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION

TYPE	DESCRIPTION
System.Threading.Tasks.Task<PDLAndBRReading>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the PLM does not support the specified wavelength.
System.ArgumentException	Thrown when the specified detector is out of range for the PLM.
System.InvalidOperationException	Thrown when the method is called on an uninitialized PLM.
System.InvalidOperationException	Thrown when a PDL reference has not been performed for the specified detector and wavelength.
UnexpectedDeviceResponseException	Thrown when the PLM read PDL and BR response is not a valid set of values.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the PLM.

ReadPDLAsync(Int32, Int32, CancellationToken)

Asynchronously read the polarization dependent loss from a detector at a given wavelength.

Declaration

```
Task<PDLReading> ReadPDLAsync(int detector, int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	The detector to read the polarization dependent loss for.
System.Int32	wavelength	The wavelength to read the polarization dependent loss at.

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<PDLReading>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the PLM does not support the specified wavelength.
System.ArgumentException	Thrown when the specified detector is out of range for the PLM.
System.InvalidOperationException	Thrown when the method is called on an uninitialized PLM.
System.InvalidOperationException	Thrown when a PDL reference has not been performed for the specified detector and wavelength.
UnexpectedDeviceResponseException	Thrown when the PLM read PDL response is not a valid set of values.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the PLM.

ReferencePDLAsync(Int32, Int32, CancellationToken)

Asynchronously perform a polarization dependent loss reference for a detector at a given wavelength.

Declaration

```
Task<List<double>> ReferencePDLAsync(int detector, int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	The detector to perform the reference for.
System.Int32	wavelength	The wavelength to perform the reference at.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Collections.Generic.List<System.Double>>	The task representing the operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the PLM does not support the specified wavelength.
System.ArgumentException	Thrown when the specified detector is out of range for the PLM.
System.InvalidOperationException	Thrown when the method is called on an uninitialized PLM.
UnexpectedDeviceResponseException	Thrown when the PLM reference PDL response is not a valid list of values.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the PLM.

SetPolarizationModeAsync(EPolarizationMode, CancellationToken)

Asynchronously set the polarization mode.

Declaration

```
Task SetPolarizationModeAsync(EPolarizationMode polarizationMode, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
EPolarizationMode	polarizationMode	The polarization mode to change to.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized PLM.
UnexpectedDeviceResponseException	Thrown when the PLM polarization mode response does not match the expected response.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the PLM.

Interface IPolarizationController

Defines the operations that can be performed with a polarization controller.

Namespace: Santec.Hardware.Devices.Polarization

Assembly: Santec.Hardware.dll

Syntax

```
public interface IPolarizationController
```

Methods

GetPolarizationStateAsync(CancellationToken)

Asynchronously get the polarization state.

Declaration

```
Task<EPolarizationState> GetPolarizationStateAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<EPolarizationState>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized polarization controller.
UnexpectedDeviceResponseException	Thrown when the polarization controller response does not correspond to a polarization state.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the polarization controller.

GetWaveplatePositionAsync(Int32, CancellationToken)

Asynchronously get the position of a given waveplate.

Declaration

```
Task<double> GetWaveplatePositionAsync(int waveplateIndex, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	waveplateIndex	The waveplate to get the position of. `0` or `1` to specify which waveplate.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	The task representing the asynchronous operation.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the specified waveplate is out of range for the device.
System.InvalidOperationException	Thrown when the method is called on an uninitialized polarization controller.
UnexpectedDeviceResponseException	Thrown when the polarization controller waveplate position response is not a valid waveplate position.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the polarization controller.

SetPolarizationStateAsync(EPolarizationState, CancellationToken)

Asynchronously set the polarization mode.

Declaration

```
Task SetPolarizationStateAsync(EPolarizationState polarizationState, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
EPolarizationState	polarizationState	The polarization state to change to.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the specified polarization state is Unknown.
System.InvalidOperationException	Thrown when the method is called on an uninitialized polarization controller.
UnexpectedDeviceResponseException	Thrown when the polarization controller polarization state response does not match the expected response.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the polarization controller.

SetWaveplatePositionAsync(Int32, Double, CancellationToken)

Asynchronously set the position of a given waveplate.

Declaration

```
Task SetWaveplatePositionAsync(int waveplateIndex, double position, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	waveplateIndex	The waveplate to set the position of. `0` or `1` to specify which waveplate.

TYPE	NAME	DESCRIPTION
System.Double	position	The position to set the waveplate to. Between `0` and `360` degrees.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	The task representing the asynchronous operation.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the specified waveplate is out of range for the device.
System.ArgumentException	Thrown when the specified position is not a valid waveplate position.
System.InvalidOperationException	Thrown when the method is called on an uninitialized polarization controller.
UnexpectedDeviceResponseException	Thrown when the polarization controller waveplate position response does not match the expected response.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the polarization controller.

Class PCU

Provides a connection to a PCU along with properties of the device and operations that can be performed using it.

> **NOTE**
>
> This class cannot be used directly. It can only be used when returned:
>
> 1. As part of a list of devices by a DetectHardwareAsync(EConnectionTypesToDetect) or DetectHardwareAsync<T>(EConnectionTypesToDetect) call.
> 2. As a device by a TryDetectIPDeviceAsync(IPAddress) or TryDetectIPDeviceAsync<T>(IPAddress) call.

Inheritance

System.Object
PhysicalDevice
SantecDevice
PCU

Implements

IPCU
ISantecDevice
IPolarizationController
IDeviceWithInputs
ISwitchController
IDevice<EPCUModel>
IDevice
System.IDisposable

Inherited Members

SantecDevice.HandleIDNResponse(String[])
SantecDevice.GetIPAddressAsync(CancellationToken)
SantecDevice.SetIPAddressAsync(IPAddress, CancellationToken)
SantecDevice.EnableDHCPAsync(CancellationToken)
SantecDevice.GetGatewayAsync(CancellationToken)
SantecDevice.SetGatewayAsync(IPAddress, CancellationToken)
SantecDevice.GetSubnetPrefixLengthAsync(CancellationToken)
SantecDevice.SetSubnetPrefixLengthAsync(Int32, CancellationToken)
SantecDevice.GetHostnameAsync(CancellationToken)
SantecDevice.SetHostnameAsync(String, CancellationToken)
SantecDevice.GetInteractionModeAsync(CancellationToken)
SantecDevice.SetInteractionModeAsync(EInteractionMode, CancellationToken)
PhysicalDevice.Address
PhysicalDevice.SerialNumber
PhysicalDevice.FirmwareVersion
PhysicalDevice.IsInitialized
PhysicalDevice.IsInitializing
PhysicalDevice.SetTimeout(ETimeout)
PhysicalDevice.GetTimeout()
PhysicalDevice.Dispose()
PhysicalDevice.Dispose(Boolean)
PhysicalDevice.PingAsync()
PhysicalDevice.ResetAsync()
PhysicalDevice.QueryAsync(String, CancellationToken)

PhysicalDevice.WriteAsync(String, CancellationToken)
PhysicalDevice.ReadAsync(CancellationToken)
System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.Polarization

Assembly: Santec.Hardware.dll

Syntax

```
public class PCU : SantecDevice, IPCU, ISantecDevice, IPolarizationController, IDeviceWithInputs,
ISwitchController, IDevice<EPCUModel>, IDevice, IDisposable
```

Properties

Model

Declaration

```
public EPCUModel Model { get; }
```

Property Value

TYPE	DESCRIPTION
EPCUModel	

Name

Declaration

```
public override string Name { get; }
```

Property Value

TYPE	DESCRIPTION
System.String	

Overrides

PhysicalDevice.Name

Remarks

> **NOTE**
>
> This property will always return "PCU".

NumberOfInputs

Declaration

```
public int NumberOfInputs { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

Switches

Declaration

```
public List<DeviceSwitch> Switches { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.List<DeviceSwitch>	

Methods

ChangeInputChannelAsync(Int32, CancellationToken)

Declaration

```
public async Task ChangeInputChannelAsync(int input, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	input	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

ChangeMultipleSwitchesChannelsAsync(IEnumerable<DeviceSwitchAndChannelPair>, CancellationToken)

Declaration

```
public Task ChangeMultipleSwitchesChannelsAsync(IEnumerable<DeviceSwitchAndChannelPair>
deviceSwitchesAndChannels, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Collections.Generic.IEnumerable<DeviceSwitchAndChannelPair>	deviceSwitchesAndChannels	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

ChangeSwitchChannelAsync(Int32, Int32, CancellationToken)

Declaration

```
public Task ChangeSwitchChannelAsync(int switchIndex, int channel, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	switchIndex	
System.Int32	channel	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

GetInputChannelAsync(CancellationToken)

Declaration

```
public async Task<int> GetInputChannelAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Int32>	

GetPolarizationStateAsync(CancellationToken)

Declaration

```
public Task<EPolarizationState> GetPolarizationStateAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<EPolarizationState>	

GetSwitchChannelAsync(Int32, CancellationToken)

Declaration

```
public Task<int> GetSwitchChannelAsync(int switchIndex, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	switchIndex	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Int32>	

GetWaveplatePositionAsync(Int32, CancellationToken)

Declaration

```
public Task<double> GetWaveplatePositionAsync(int waveplateIndex, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	waveplateIndex	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

InitializeAsync()

Declaration

```
public override async Task InitializeAsync()
```

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Overrides

SantecDevice.InitializeAsync()

RefreshAsync()

Declaration

```
public override async Task RefreshAsync()
```

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Overrides

PhysicalDevice.RefreshAsync()

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized PCU.

SetPolarizationStateAsync(EPolarizationState, CancellationToken)

Declaration

```
public Task SetPolarizationStateAsync(EPolarizationState polarizationState, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
EPolarizationState	polarizationState	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetWaveplatePositionAsync(Int32, Double, CancellationToken)

Declaration

```
public Task SetWaveplatePositionAsync(int waveplateIndex, double position, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	waveplateIndex	
System.Double	position	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Uninitialize()

Declaration

```
public override void Uninitialize()
```

Overrides

PhysicalDevice.Uninitialize()

Implements

IPCU
ISantecDevice
IPolarizationController
IDeviceWithInputs
ISwitchController
IDevice<TModelType>
IDevice
System.IDisposable

Class PDLAndBRReading

Represents the results of a PDL and BR reading.

> **NOTE**
>
> This class should not be used directly. It should only be used when returned by a PDL and BR measurement.

Inheritance

System.Object
PDLReading
PDLAndBRReading

Inherited Members

PDLReading.ILAverage
PDLReading.PDL
System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.Polarization

Assembly: Santec.Hardware.dll

Syntax

```
public class PDLAndBRReading : PDLReading
```

Constructors

PDLAndBRReading(Double, Double, ReadingValue)

Initialize a new PDL and BR reading result.

Declaration

```
public PDLAndBRReading(double ilAverage, double pdl, ReadingValue br)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Double	ilAverage	The insertion loss average of the fiber.
System.Double	pdl	The polarization dependent loss of the fiber.
ReadingValue	br	The backreflection of the fiber.

Properties

BR

Get the backreflection of the fiber.

Declaration

```
public ReadingValue BR { get; }
```

Property Value

TYPE	DESCRIPTION
ReadingValue	The backreflection.

Class PDLReading

Represents the results of a PDL reading.

> **NOTE**
>
> This class should not be used directly. It should only be used when returned by a PDL measurement.

Inheritance

System.Object
PDLReading
PDLAndBRReading

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.Polarization

Assembly: Santec.Hardware.dll

Syntax

```
public class PDLReading
```

Constructors

PDLReading(Double, Double)

Initialize a new PDL reading result.

Declaration

```
public PDLReading(double ilAverage, double pdl)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Double	ilAverage	The insertion loss average of the fiber.
System.Double	pdl	The polarization dependent loss of the fiber.

Properties

ILAverage

Get the insertion loss average of the fiber.

Declaration

```
public double ILAverage { get; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The insertion loss average.

PDL

Get the polarization dependent loss of the fiber.

Declaration

```
public double PDL { get; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The polarization dependent loss.

Class PEM

Provides a connection to a PEM along with properties of the device and operations that can be performed using it.

> **NOTE**
>
> This class cannot not be used directly. It can only be used when returned:
>
> 1. As part of a list of devices by a DetectHardwareAsync(EConnectionTypesToDetect) or DetectHardwareAsync<T>(EConnectionTypesToDetect) call.
> 2. As a device by a TryDetectIPDeviceAsync(IPAddress) or TryDetectIPDeviceAsync<T>(IPAddress) call.

Inheritance

System.Object
PhysicalDevice
SantecDevice
PEM

Implements

IPEM
ISantecDevice
IInsertionLossMeter
ILaserSource
IWavelengthDevice
IDetectorDeviceWithDarking
IDetectorDevice
ISwitchController
IFiberDevice
IDevice<EPEMModel>
IDevice
System.IDisposable

Inherited Members

SantecDevice.HandleIDNResponse(String[])
SantecDevice.GetIPAddressAsync(CancellationToken)
SantecDevice.SetIPAddressAsync(IPAddress, CancellationToken)
SantecDevice.EnableDHCPAsync(CancellationToken)
SantecDevice.GetGatewayAsync(CancellationToken)
SantecDevice.SetGatewayAsync(IPAddress, CancellationToken)
SantecDevice.GetSubnetPrefixLengthAsync(CancellationToken)
SantecDevice.SetSubnetPrefixLengthAsync(Int32, CancellationToken)
SantecDevice.GetHostnameAsync(CancellationToken)
SantecDevice.SetHostnameAsync(String, CancellationToken)
SantecDevice.GetInteractionModeAsync(CancellationToken)
SantecDevice.SetInteractionModeAsync(EInteractionMode, CancellationToken)
PhysicalDevice.Address
PhysicalDevice.SerialNumber
PhysicalDevice.FirmwareVersion
PhysicalDevice.IsInitialized
PhysicalDevice.IsInitializing
PhysicalDevice.SetTimeout(ETimeout)
PhysicalDevice.GetTimeout()
PhysicalDevice.Dispose()

PhysicalDevice.Dispose(Boolean)
PhysicalDevice.PingAsync()
PhysicalDevice.ResetAsync()
PhysicalDevice.QueryAsync(String, CancellationToken)
PhysicalDevice.WriteAsync(String, CancellationToken)
PhysicalDevice.ReadAsync(CancellationToken)
System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.Polarization

Assembly: Santec.Hardware.dll

Syntax

```
public class PEM : SantecDevice, IPEM, ISantecDevice, IInsertionLossMeter, ILaserSource, IWavelengthDevice,
IDetectorDeviceWithDarking, IDetectorDevice, ISwitchController, IFiberDevice, IDevice<EPEMModel>, IDevice,
IDisposable
```

Properties

Detectors

Declaration

```
public IReadOnlyList<IModule> Detectors { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<IModule>	

Fiber

Declaration

```
public FiberInformation Fiber { get; }
```

Property Value

TYPE	DESCRIPTION
FiberInformation	

Model

Declaration

```
public EPEMModel Model { get; }
```

Property Value

TYPE	DESCRIPTION
EPEMModel	

Name

Declaration

```
public override string Name { get; }
```

Property Value

TYPE	DESCRIPTION
System.String	

Overrides

PhysicalDevice.Name

Remarks

> **NOTE**
>
> This property will always return "PEM".

NumberOfDetectors

Declaration

```
public int NumberOfDetectors { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

NumberOfOutputs

Declaration

```
public int NumberOfOutputs { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

SupportsTurningOffLaser

Declaration

```
public bool SupportsTurningOffLaser { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	

Remarks

> **NOTE**
>
> This property will always return `false`.

Switches

Declaration

```
public List<DeviceSwitch> Switches { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.List<DeviceSwitch>	

Wavelengths

Declaration

```
public List<int> Wavelengths { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.List<System.Int32>	

Methods

ChangeMultipleSwitchesChannelsAsync(IEnumerable<DeviceSwitchAndChannelPair>, CancellationToken)

Declaration

```
public Task ChangeMultipleSwitchesChannelsAsync(IEnumerable<DeviceSwitchAndChannelPair>
deviceSwitchesAndChannels, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Collections.Generic.IEnumerable<DeviceSwitchAndChannelPair>	deviceSwitchesAndChannels	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

ChangeOutputChannelAsync(Int32, CancellationToken)

Declaration

```
public Task ChangeOutputChannelAsync(int output, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	output	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

ChangeSwitchChannelAsync(Int32, Int32, CancellationToken)

Declaration

```
public Task ChangeSwitchChannelAsync(int switchIndex, int channel, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	switchIndex	
System.Int32	channel	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

DarkAllDetectorsAsync(CancellationToken)

Declaration

```
public Task DarkAllDetectorsAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

DarkDetectorAsync(Int32, CancellationToken)

Declaration

```
public Task DarkDetectorAsync(int detector, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

GetActiveLaserAsync(CancellationToken)

Declaration

```
public Task<int> GetActiveLaserAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Int32>	

GetAutoPowerAdjustmentStateAsync(CancellationToken)

Declaration

```
public async Task<EAutoPowerAdjustmentState> GetAutoPowerAdjustmentStateAsync(CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<EAutoPowerAdjustmentState>	

GetDetectorDarkStateAsync(Int32, CancellationToken)

Declaration

```
public Task<EDarkState> GetDetectorDarkStateAsync(int detector, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<EDarkState>	

GetILReferenceAsync(Int32, Int32, CancellationToken)

Declaration

```
public Task<double> GetILReferenceAsync(int detector, int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

GetOutputChannelAsync(CancellationToken)

Declaration

```
public Task<int> GetOutputChannelAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Int32>	

GetPERModeAsync(CancellationToken)

Declaration

```
public async Task<EPERMode> GetPERModeAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<EPERMode>	

GetSwitchChannelAsync(Int32, CancellationToken)

Declaration

```
public Task<int> GetSwitchChannelAsync(int switchIndex, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	switchIndex	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Int32>	

InitializeAsync()

Declaration

```
public override async Task InitializeAsync()
```

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Overrides

SantecDevice.InitializeAsync()

Exceptions

TYPE	CONDITION
UnexpectedDeviceResponseException	Thrown when the PEM wavelengths response contains one or more wavelengths that are not valid values.
UnexpectedDeviceResponseException	Thrown when the PEM detectors response is not a valid number.
UnexpectedDeviceResponseException	Thrown when the PEM switches response contains one or more switch channel counts that are not valid values.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the PEM.

ReadILAsync(Int32, Int32, CancellationToken)

Declaration

```
public Task<double> ReadILAsync(int detector, int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

ReadMaximumPowerAsync(Int32, Int32, CancellationToken)

Declaration

```
public async Task<double> ReadMaximumPowerAsync(int detector, int wavelength, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

ReadMinimumPowerAsync(Int32, Int32, CancellationToken)

Declaration

```
public async Task<double> ReadMinimumPowerAsync(int detector, int wavelength, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

ReadPERAsync(Int32, Int32, CancellationToken)

Declaration

```
public async Task<PERReading> ReadPERAsync(int detector, int wavelength, CancellationToken cancellationToken
= default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<PERReading>	

ReferenceILAsync(Int32, Int32, Boolean, CancellationToken)

Declaration

```
public Task<double> ReferenceILAsync(int detector, int wavelength, bool applyReferenceToAllDetectors,
CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Int32	wavelength	
System.Boolean	applyReferenceToAllDetectors	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

ReferenceILAsync(Int32, Int32, CancellationToken)

Declaration

```
public Task<double> ReferenceILAsync(int detector, int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

RefreshAsync()

Declaration

```
public override async Task RefreshAsync()
```

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Overrides

PhysicalDevice.RefreshAsync()

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized PEM.
UnexpectedDeviceResponseException	Thrown when the PEM wavelengths response contains one or more wavelengths that are not valid values.
UnexpectedDeviceResponseException	Thrown when the PEM detectors response is not a valid number.
UnexpectedDeviceResponseException	Thrown when the PEM switches response contains one or more switch channel counts that are not valid values.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the PEM.

SetAutoPowerAdjustmentStateAsync(EAutoPowerAdjustmentState, CancellationToken)

Declaration

```
public async Task SetAutoPowerAdjustmentStateAsync(EAutoPowerAdjustmentState autoPowerAdjustmentState,
CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
EAutoPowerAdjustmentState	autoPowerAdjustmentState	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetILReferenceAsync(Int32, Int32, Double, CancellationToken)

Declaration

```
public Task SetILReferenceAsync(int detector, int wavelength, double reference, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Int32	wavelength	
System.Double	reference	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetPERModeAsync(EPERMode, CancellationToken)

Declaration

```
public async Task SetPERModeAsync(EPERMode perMode, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
EPERMode	perMode	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

TurnOffLaserAsync(CancellationToken)

Declaration

```
public Task TurnOffLaserAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

TurnOnLaserAsync(Int32, CancellationToken)

Declaration

```
public Task TurnOnLaserAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Uninitialize()

Declaration

```
public override void Uninitialize()
```

Overrides

PhysicalDevice.Uninitialize()

Implements

IPEM
ISantecDevice
IInsertionLossMeter
ILaserSource
IWavelengthDevice
IDetectorDeviceWithDarking
IDetectorDevice
ISwitchController
IFiberDevice
IDevice<TModelType>
IDevice
System.IDisposable

Class PERReading

Represents the results of a PER reading.

> **NOTE**
>
> This class should not be used directly. It should only be used when returned by a PER measurement.

Inheritance

System.Object
PERReading

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.Polarization

Assembly: Santec.Hardware.dll

Syntax

```
public class PERReading
```

Constructors

PERReading(Double, Double, Double, Double)

Initialize a new PER reading result.

Declaration

```
public PERReading(double il, double per, double angle, double maximumPower)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Double	il	The insertion loss of the fiber.
System.Double	per	The polarization extinction ratio of the fiber.
System.Double	angle	The angle of the polarization with respect to the vertical axis.
System.Double	maximumPower	The maximum power of the PER measurement.

Properties

Angle

Get the angle of the polarization with repect to the vertical axis.

Declaration

```
public double Angle { get; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The polarization angle.

IL

Get the insertion loss average of the fiber.

Declaration

```
public double IL { get; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The insertion loss average.

MaximumPower

Get the maximum power of the PER measurement.

Declaration

```
public double MaximumPower { get; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The maximum power.

PER

Get the polarization extinction ratio of the fiber.

Declaration

```
public double PER { get; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The polarization extinction ratio.

Class PLM

Provides a connection to a PLM along with properties of the device and operations that can be performed using it.

> **NOTE**
>
> This class cannot not be used directly. It can only be used when returned:
>
> 1. As part of a list of devices by a DetectHardwareAsync(EConnectionTypesToDetect) or DetectHardwareAsync<T>(EConnectionTypesToDetect) call.
> 2. As a device by a TryDetectIPDeviceAsync(IPAddress) or TryDetectIPDeviceAsync<T>(IPAddress) call.

Inheritance

System.Object
PhysicalDevice
SantecDevice
PLM

Implements

IPLM
ISantecDevice
IDiscretePowerMeter
IPowerMeter
IInsertionLossMeter
ILaserSourceWithMonitor
ILaserSource
ISwitchController
IBackreflectionMeter
IWavelengthDevice
IDetectorDeviceWithDarking
IDetectorDevice
IFiberDevice
IPolarizationController
IDevice<EPLMModel>
IDevice
System.IDisposable

Inherited Members

SantecDevice.HandleIDNResponse(String[])
SantecDevice.GetIPAddressAsync(CancellationToken)
SantecDevice.SetIPAddressAsync(IPAddress, CancellationToken)
SantecDevice.EnableDHCPAsync(CancellationToken)
SantecDevice.GetGatewayAsync(CancellationToken)
SantecDevice.SetGatewayAsync(IPAddress, CancellationToken)
SantecDevice.GetSubnetPrefixLengthAsync(CancellationToken)
SantecDevice.SetSubnetPrefixLengthAsync(Int32, CancellationToken)
SantecDevice.GetHostnameAsync(CancellationToken)
SantecDevice.SetHostnameAsync(String, CancellationToken)
SantecDevice.GetInteractionModeAsync(CancellationToken)
SantecDevice.SetInteractionModeAsync(EInteractionMode, CancellationToken)
PhysicalDevice.Address
PhysicalDevice.SerialNumber
PhysicalDevice.FirmwareVersion

PhysicalDevice.IsInitialized
PhysicalDevice.IsInitializing
PhysicalDevice.SetTimeout(ETimeout)
PhysicalDevice.GetTimeout()
PhysicalDevice.Dispose()
PhysicalDevice.Dispose(Boolean)
PhysicalDevice.PingAsync()
PhysicalDevice.ResetAsync()
PhysicalDevice.QueryAsync(String, CancellationToken)
PhysicalDevice.WriteAsync(String, CancellationToken)
PhysicalDevice.ReadAsync(CancellationToken)
System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.Polarization

Assembly: Santec.Hardware.dll

Syntax

```
public class PLM : SantecDevice, IPLM, ISantecDevice, IDiscretePowerMeter, IPowerMeter, IInsertionLossMeter,
ILaserSourceWithMonitor, ILaserSource, ISwitchController, IBackreflectionMeter, IWavelengthDevice,
IDetectorDeviceWithDarking, IDetectorDevice, IFiberDevice, IPolarizationController, IDevice<EPLMModel>,
IDevice, IDisposable
```

Properties

Detectors

Declaration

```
public IReadOnlyList<DeviceDetector> Detectors { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<DeviceDetector>	

Fiber

Declaration

```
public FiberInformation Fiber { get; }
```

Property Value

TYPE	DESCRIPTION
FiberInformation	

Model

Declaration

```
public EPLMModel Model { get; }
```

Property Value

TYPE	DESCRIPTION
EPLMModel	

Name

Declaration

```
public override string Name { get; }
```

Property Value

TYPE	DESCRIPTION
System.String	

Overrides

PhysicalDevice.Name

Remarks

> **NOTE**
>
> This property will always return "PLM".

NumberOfDetectors

Declaration

```
public int NumberOfDetectors { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

NumberOfOutputs

Declaration

```
public int NumberOfOutputs { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

SupportsTurningOffLaser

Declaration

```
public bool SupportsTurningOffLaser { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	

Remarks

> **NOTE**
>
> This property will always return `true`.

Switches

Declaration

```
public List<DeviceSwitch> Switches { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.List<DeviceSwitch>	

Wavelengths

Declaration

```
public List<int> Wavelengths { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.List<System.Int32>	

Methods

ChangeMultipleSwitchesChannelsAsync(IEnumerable<DeviceSwitchAndChannelPair>, CancellationToken)

Declaration

```
public Task ChangeMultipleSwitchesChannelsAsync(IEnumerable<DeviceSwitchAndChannelPair>
deviceSwitchesAndChannels, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Collections.Generic.IEnumerable<DeviceSwitchAndChannelPair>	deviceSwitchesAndChannels	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

ChangeOutputChannelAsync(Int32, CancellationToken)

Declaration

```
public Task ChangeOutputChannelAsync(int output, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	output	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

ChangeSwitchChannelAsync(Int32, Int32, CancellationToken)

Declaration

```
public Task ChangeSwitchChannelAsync(int switchIndex, int channel, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	switchIndex	
System.Int32	channel	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

DarkAllDetectorsAsync(CancellationToken)

Declaration

```
public Task DarkAllDetectorsAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

DarkBRDetectorAsync(CancellationToken)

Declaration

```
public Task DarkBRDetectorAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

DarkDetectorAsync(Int32, CancellationToken)

Declaration

```
public Task DarkDetectorAsync(int detector, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

GetActiveLaserAsync(CancellationToken)

Declaration

```
public Task<int> GetActiveLaserAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Int32>	

GetBRReferenceAsync(Int32, CancellationToken)

Declaration

```
public Task<double> GetBRReferenceAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

GetDetectorDarkStateAsync(Int32, CancellationToken)

Declaration

```
public Task<EDarkState> GetDetectorDarkStateAsync(int detector, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<EDarkState>	

GetFactoryPowerAsync(Int32, Int32, CancellationToken)

Declaration

```
public Task<FactoryPowerReading> GetFactoryPowerAsync(int output, int wavelength, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	output	
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<FactoryPowerReading>	

GetILReferenceAsync(Int32, Int32, CancellationToken)

Declaration

```
public Task<double> GetILReferenceAsync(int detector, int wavelength, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

GetOutputChannelAsync(CancellationToken)

Declaration

```
public Task<int> GetOutputChannelAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Int32>	

GetPDLReferencesAsync(Int32, Int32, CancellationToken)

Declaration

```
public async Task<List<double>> GetPDLReferencesAsync(int detector, int wavelength, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Collections.Generic.List<System.Double>>	

GetPolarizationModeAsync(CancellationToken)

Declaration

```
public async Task<EPolarizationMode> GetPolarizationModeAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<EPolarizationMode>	

GetPolarizationStateAsync(CancellationToken)

Declaration

```
public Task<EPolarizationState> GetPolarizationStateAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<EPolarizationState>	

GetSwitchChannelAsync(Int32, CancellationToken)

Declaration

```
public Task<int> GetSwitchChannelAsync(int switchIndex, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	switchIndex	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Int32>	

GetWaveplatePositionAsync(Int32, CancellationToken)

Declaration

```
public Task<double> GetWaveplatePositionAsync(int waveplateIndex, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	waveplateIndex	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

InitializeAsync()

Declaration

```
public override async Task InitializeAsync()
```

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Overrides

SantecDevice.InitializeAsync()

Exceptions

TYPE	CONDITION
UnexpectedDeviceResponseException	Thrown when the PLM wavelengths response contains one or more wavelengths that are not valid values.
UnexpectedDeviceResponseException	Thrown when the PLM detectors response is not a valid number.
UnexpectedDeviceResponseException	Thrown when the PLM switches response contains one or more switch channel counts that are not valid values.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the PLM.

ReadBRAsync(Int32, CancellationToken)

Declaration

```
public Task<ReadingValue> ReadBRAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<ReadingValue>	

ReadILAsync(Int32, Int32, CancellationToken)

Declaration

```
public Task<double> ReadILAsync(int detector, int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Int32	wavelength	

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

ReadMonitorPowerAsync(Int32, CancellationToken)

Declaration

```
public Task<double> ReadMonitorPowerAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

ReadMultipleILsAsync(IEnumerable<Int32>, Int32, CancellationToken)

Declaration

```
public Task<List<double>> ReadMultipleILsAsync(IEnumerable<int> detectors, int wavelength, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Collections.Generic.IEnumerable<System.Int32>	detectors	
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Collections.Generic.List<System.Double>>	

ReadMultiplePowersAsync(IEnumerable<Int32>, Int32, CancellationToken)

Declaration

```
public Task<List<double>> ReadMultiplePowersAsync(IEnumerable<int> detectors, int wavelength,
CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Collections.Generic.IEnumerable<System.Int32>	detectors	
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Collections.Generic.List<System.Double>>	

ReadPDLAndBRAsync(Int32, Int32, CancellationToken)

Declaration

```
public async Task<PDLAndBRReading> ReadPDLAndBRAsync(int detector, int wavelength, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<PDLAndBRReading>	

ReadPDLAsync(Int32, Int32, CancellationToken)

Declaration

```
public async Task<PDLReading> ReadPDLAsync(int detector, int wavelength, CancellationToken cancellationToken
= default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Int32	wavelength	

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<PDLReading>	

ReadPowerAsync(Int32, Int32, CancellationToken)

Declaration

```
public Task<double> ReadPowerAsync(int detector, int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

ReferenceBRAsync(Int32, CancellationToken)

Declaration

```
public Task<double> ReferenceBRAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

ReferenceILAsync(Int32, Int32, Boolean, CancellationToken)

Declaration

```
public Task<double> ReferenceILAsync(int detector, int wavelength, bool applyReferenceToAllDetectors,
CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Int32	wavelength	
System.Boolean	applyReferenceToAllDetectors	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

ReferenceILAsync(Int32, Int32, CancellationToken)

Declaration

```
public Task<double> ReferenceILAsync(int detector, int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

ReferencePDLAsync(Int32, Int32, CancellationToken)

Declaration

```
public async Task<List<double>> ReferencePDLAsync(int detector, int wavelength, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Collections.Generic.List<System.Double>>	

RefreshAsync()

Declaration

```
public override async Task RefreshAsync()
```

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Overrides

PhysicalDevice.RefreshAsync()

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized PLM.
UnexpectedDeviceResponseException	Thrown when the PLM wavelengths response contains one or more wavelengths that are not valid values.
UnexpectedDeviceResponseException	Thrown when the PLM detectors response is not a valid number.
UnexpectedDeviceResponseException	Thrown when the PLM switches response contains one or more switch channel counts that are not valid values.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the PLM.

SetBRReferenceAsync(Int32, Double, CancellationToken)

Declaration

```
public Task SetBRReferenceAsync(int wavelength, double reference, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	
System.Double	reference	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetILReferenceAsync(Int32, Int32, Double, CancellationToken)

Declaration

```
public Task SetILReferenceAsync(int detector, int wavelength, double reference, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Int32	wavelength	
System.Double	reference	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetPolarizationModeAsync(EPolarizationMode, CancellationToken)

Declaration

```
public async Task SetPolarizationModeAsync(EPolarizationMode polarizationMode, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
EPolarizationMode	polarizationMode	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetPolarizationStateAsync(EPolarizationState, CancellationToken)

Declaration

```
public Task SetPolarizationStateAsync(EPolarizationState polarizationState, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
EPolarizationState	polarizationState	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetWaveplatePositionAsync(Int32, Double, CancellationToken)

Declaration

```
public Task SetWaveplatePositionAsync(int waveplateIndex, double position, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	waveplateIndex	
System.Double	position	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

TurnOffLaserAsync(CancellationToken)

Declaration

```
public Task TurnOffLaserAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

TurnOnLaserAsync(Int32, CancellationToken)

Declaration

```
public Task TurnOnLaserAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Uninitialize()

Declaration

```
public override void Uninitialize()
```

Overrides

PhysicalDevice.Uninitialize()

Explicit Interface Implementations

IDetectorDevice.Detectors

Declaration

```
IReadOnlyList<IModule> IDetectorDevice.Detectors { get; }
```

Returns

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<IModule>	

Implements

IPLM

ISantecDevice

IDiscretePowerMeter
IPowerMeter
IInsertionLossMeter
ILaserSourceWithMonitor
ILaserSource
ISwitchController
IBackreflectionMeter
IWavelengthDevice
IDetectorDeviceWithDarking
IDetectorDevice
IFiberDevice
IPolarizationController
IDevice<TModelType>
IDevice
System.IDisposable

Class SimulatedPCU

Provides a simulated PCU.

> **NOTE**
>
> This class is intended for testing/development purposes. It operates much quicker than an actual PCU. It can only be created using a SimulatedDeviceFactory. It should only be used when returned as part of a list of devices by a DetectHardwareAsync(EConnectionTypesToDetect) or DetectHardwareAsync<T>(EConnectionTypesToDetect) call to a SimulatedHardwareDetectionService.

Inheritance

System.Object
SimulatedSantecDevice
SimulatedPCU

Implements

ISimulatedDevice
IPCU
ISantecDevice
IPolarizationController
IDeviceWithInputs
ISwitchController
IDevice<EPCUModel>
IDevice
System.IDisposable

Inherited Members

SimulatedSantecDevice.creatingDevice
SimulatedSantecDevice.Dispose()
SimulatedSantecDevice.SerialNumber
SimulatedSantecDevice.FirmwareVersion
SimulatedSantecDevice.Address
SimulatedSantecDevice.IsInitialized
SimulatedSantecDevice.QueryResponses
SimulatedSantecDevice.ConfigureQueryResponses(IEnumerable<QueryResponsePair>)
SimulatedSantecDevice.GetTimeout()
SimulatedSantecDevice.SetTimeout(ETimeout)
SimulatedSantecDevice.GetInteractionModeAsync(CancellationToken)
SimulatedSantecDevice.GetIPAddressAsync(CancellationToken)
SimulatedSantecDevice.SetIPAddressAsync(IPAddress, CancellationToken)
SimulatedSantecDevice.EnableDHCPAsync(CancellationToken)
SimulatedSantecDevice.GetGatewayAsync(CancellationToken)
SimulatedSantecDevice.SetGatewayAsync(IPAddress, CancellationToken)
SimulatedSantecDevice.GetSubnetPrefixLengthAsync(CancellationToken)
SimulatedSantecDevice.SetSubnetPrefixLengthAsync(Int32, CancellationToken)
SimulatedSantecDevice.GetHostnameAsync(CancellationToken)
SimulatedSantecDevice.SetHostnameAsync(String, CancellationToken)
SimulatedSantecDevice.SetInteractionModeAsync(EInteractionMode, CancellationToken)
SimulatedSantecDevice.InitializeAsync()
SimulatedSantecDevice.PingAsync()
SimulatedSantecDevice.Uninitialize()
SimulatedSantecDevice.ResetAsync()
SimulatedSantecDevice.RefreshAsync()

SimulatedSantecDevice.QueryAsync(String, CancellationToken)
SimulatedSantecDevice.WriteAsync(String, CancellationToken)
SimulatedSantecDevice.ReadAsync(CancellationToken)
System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.Polarization

Assembly: Santec.Hardware.dll

Syntax

```
public class SimulatedPCU : SimulatedSantecDevice, ISimulatedDevice, IPCU, ISantecDevice,
IPolarizationController, IDeviceWithInputs, ISwitchController, IDevice<EPCUModel>, IDevice, IDisposable
```

Properties

Model

Declaration

```
public EPCUModel Model { get; }
```

Property Value

TYPE	DESCRIPTION
EPCUModel	

Name

Declaration

```
public override string Name { get; }
```

Property Value

TYPE	DESCRIPTION
System.String	

Overrides

SimulatedSantecDevice.Name

NumberOfInputs

Declaration

```
public int NumberOfInputs { get; }
```

Property Value

TYPE	DESCRIPTION

TYPE	DESCRIPTION
System.Int32	

Switches

Declaration

```
public List<DeviceSwitch> Switches { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.List<DeviceSwitch>	

Methods

ChangeInputChannelAsync(Int32, CancellationToken)

Declaration

```
public async Task ChangeInputChannelAsync(int input, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	input	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

ChangeMultipleSwitchesChannelsAsync(IEnumerable<DeviceSwitchAndChannelPair>, CancellationToken)

Declaration

```
public async Task ChangeMultipleSwitchesChannelsAsync(IEnumerable<DeviceSwitchAndChannelPair>
deviceSwitchesAndChannels, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Collections.Generic.IEnumerable<DeviceSwitchAndChannelPair>	deviceSwitchesAndChannels	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

ChangeSwitchChannelAsync(Int32, Int32, CancellationToken)

Declaration

```
public async Task ChangeSwitchChannelAsync(int switchIndex, int channel, CancellationToken cancellationToken
= default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	switchIndex	
System.Int32	channel	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

GetInputChannelAsync(CancellationToken)

Declaration

```
public async Task<int> GetInputChannelAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Int32>	

GetPolarizationStateAsync(CancellationToken)

Declaration

```
public async Task<EPolarizationState> GetPolarizationStateAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<EPolarizationState>	

GetSwitchChannelAsync(Int32, CancellationToken)

Declaration

```
public async Task<int> GetSwitchChannelAsync(int switchIndex, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	switchIndex	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Int32>	

GetWaveplatePositionAsync(Int32, CancellationToken)

Declaration

```
public async Task<double> GetWaveplatePositionAsync(int waveplateIndex, CancellationToken cancellationToken
= default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	waveplateIndex	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

SetPolarizationStateAsync(EPolarizationState, CancellationToken)

Declaration

```
public async Task SetPolarizationStateAsync(EPolarizationState polarizationState, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
EPolarizationState	polarizationState	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetWaveplatePositionAsync(Int32, Double, CancellationToken)

Declaration

```
public async Task SetWaveplatePositionAsync(int waveplateIndex, double position, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	waveplateIndex	
System.Double	position	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Implements

ISimulatedDevice
IPCU
ISantecDevice
IPolarizationController
IDeviceWithInputs
ISwitchController
IDevice<TModelType>
IDevice
System.IDisposable

Class SimulatedPEM

Provides a simulated PEM with configurable reading values.

> **NOTE**
>
> This class is intended for testing/development purposes. It operates much quicker than an actual PEM. It can only be created using a SimulatedDeviceFactory. It should only be used when returned as part of a list of devices by a DetectHardwareAsync(EConnectionTypesToDetect) or DetectHardwareAsync<T>(EConnectionTypesToDetect) call to a SimulatedHardwareDetectionService. Each reading value contains a default value range but can be configured via the appropriate configuration method.

Inheritance

System.Object
SimulatedSantecDevice
SimulatedPEM

Implements

ISimulatedDevice
IPEM
ISantecDevice
ILaserSource
IWavelengthDevice
IDetectorDeviceWithDarking
ISwitchController
IFiberDevice
IDevice<EPEMModel>
IInsertionLossMeter
IDetectorDevice
IDevice
System.IDisposable

Inherited Members

SimulatedSantecDevice.creatingDevice
SimulatedSantecDevice.Dispose()
SimulatedSantecDevice.SerialNumber
SimulatedSantecDevice.FirmwareVersion
SimulatedSantecDevice.Address
SimulatedSantecDevice.IsInitialized
SimulatedSantecDevice.QueryResponses
SimulatedSantecDevice.ConfigureQueryResponses(IEnumerable<QueryResponsePair>)
SimulatedSantecDevice.GetTimeout()
SimulatedSantecDevice.SetTimeout(ETimeout)
SimulatedSantecDevice.GetInteractionModeAsync(CancellationToken)
SimulatedSantecDevice.GetIPAddressAsync(CancellationToken)
SimulatedSantecDevice.SetIPAddressAsync(IPAddress, CancellationToken)
SimulatedSantecDevice.EnableDHCPAsync(CancellationToken)
SimulatedSantecDevice.GetGatewayAsync(CancellationToken)
SimulatedSantecDevice.SetGatewayAsync(IPAddress, CancellationToken)
SimulatedSantecDevice.GetSubnetPrefixLengthAsync(CancellationToken)
SimulatedSantecDevice.SetSubnetPrefixLengthAsync(Int32, CancellationToken)
SimulatedSantecDevice.GetHostnameAsync(CancellationToken)
SimulatedSantecDevice.SetHostnameAsync(String, CancellationToken)
SimulatedSantecDevice.SetInteractionModeAsync(EInteractionMode, CancellationToken)

SimulatedSantecDevice.InitializeAsync()
SimulatedSantecDevice.PingAsync()
SimulatedSantecDevice.Uninitialize()
SimulatedSantecDevice.ResetAsync()
SimulatedSantecDevice.RefreshAsync()
SimulatedSantecDevice.QueryAsync(String, CancellationToken)
SimulatedSantecDevice.WriteAsync(String, CancellationToken)
SimulatedSantecDevice.ReadAsync(CancellationToken)
System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.Polarization

Assembly: Santec.Hardware.dll

Syntax

```
public class SimulatedPEM : SimulatedSantecDevice, ISimulatedDevice, IPEM, ISantecDevice, ILaserSource,
IWavelengthDevice, IDetectorDeviceWithDarking, ISwitchController, IFiberDevice, IDevice<EPEMModel>,
IInsertionLossMeter, IDetectorDevice, IDevice, IDisposable
```

Properties

AngleMax

Get the maximum Angle value returned by a PER reading.

Declaration

```
public double AngleMax { get; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The maximum Angle reading value. The default value is `90`.

Remarks

Use ConfigureAngle(Double, Double) to set this value.

> **NOTE**
>
> This value must be between -90 and 90.

AngleMin

Get the minimum Angle value returned by a PER reading.

Declaration

```
public double AngleMin { get; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The minimum Angle reading value. The default value is `0`.

Remarks

Use ConfigureAngle(Double, Double) to set this value.

> **NOTE**
>
> This value must be between -90 and 90.

Detectors

Declaration

```
public IReadOnlyList<DeviceDetector> Detectors { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<DeviceDetector>	

Fiber

Declaration

```
public FiberInformation Fiber { get; }
```

Property Value

TYPE	DESCRIPTION
FiberInformation	

ILMax

Declaration

```
public double ILMax { get; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The maximum IL reading value. The default value is `0.15`.

ILMin

Declaration

```
public double ILMin { get; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The minimum IL reading value. The default value is `0`.

ILReferenceMax

Declaration

```
public double ILReferenceMax { get; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The maximum IL reference value. The default value is `5`.

ILReferenceMin

Declaration

```
public double ILReferenceMin { get; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The minimum IL reference value. The default value is `2`.

MaximumPowerMax

Get the maximum Maximum Power value returned by a PER reading or maximum power reading.

Declaration

```
public double MaximumPowerMax { get; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The maximum Maximum Power reading value. The default value is `-3`.

Remarks

Use ConfigureMaximumPower(Double, Double) to set this value.

MaximumPowerMin

Get the minimum Maximum Power value returned by a PER reading or maximum power reading.

Declaration

```
public double MaximumPowerMin { get; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The minimum Maximum Power reading value. The default value is `-3.15`.

Remarks

Use ConfigureMaximumPower(Double, Double) to set this value.

Model

Declaration

```
public EPEMModel Model { get; }
```

Property Value

TYPE	DESCRIPTION
EPEMModel	

Name

Declaration

```
public override string Name { get; }
```

Property Value

TYPE	DESCRIPTION
System.String	

Overrides

SimulatedSantecDevice.Name

NumberOfDetectors

Declaration

```
public int NumberOfDetectors { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

NumberOfOutputs

Declaration

```
public int NumberOfOutputs { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

PERMax

Get the maximum PER value returned by a PER reading.

Declaration

```
public double PERMax { get; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The maximum PER reading value. The default value is `30`.

Remarks

Use ConfigurePER(Double, Double) to set this value.

PERMin

Get the minimum PER value returned by a PER reading.

Declaration

```
public double PERMin { get; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The minimum PER reading value. The default value is `25`.

Remarks

Use ConfigurePER(Double, Double) to set this value.

PowerMax

Declaration

```
public double PowerMax { get; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The maximum power reading value. The default value is `-3`.

PowerMin

Declaration

```
public double PowerMin { get; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The minimum power reading value. The default value is `-3.15`.

SupportsTurningOffLaser

Declaration

```
public bool SupportsTurningOffLaser { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	

Remarks

> **NOTE**
>
> This property will always return `false`.

Switches

Declaration

```
public List<DeviceSwitch> Switches { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.List<DeviceSwitch>	

Wavelengths

Declaration

```
public List<int> Wavelengths { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.List<System.Int32>	

Methods

ChangeMultipleSwitchesChannelsAsync(IEnumerable<DeviceSwitchAndChannelPair>, CancellationToken)

Declaration

```
public async Task ChangeMultipleSwitchesChannelsAsync(IEnumerable<DeviceSwitchAndChannelPair>
deviceSwitchesAndChannels, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Collections.Generic.IEnumerable<DeviceSwitchAndChannelPair>	deviceSwitchesAndChannels	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

ChangeOutputChannelAsync(Int32, CancellationToken)

Declaration

```
public async Task ChangeOutputChannelAsync(int output, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	output	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

ChangeSwitchChannelAsync(Int32, Int32, CancellationToken)

Declaration

```
public async Task ChangeSwitchChannelAsync(int switchIndex, int channel, CancellationToken cancellationToken
= default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	switchIndex	
System.Int32	channel	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

ConfigureAngle(Double, Double)

Configure the value range for the Angle returned by ReadPERAsync(Int32, Int32, CancellationToken).

Declaration

```
public void ConfigureAngle(double angleMin, double angleMax)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Double	angleMin	The minimum Angle value.
System.Double	angleMax	The maximum Angle value.

Remarks

> **NOTE**
>
> The angles must be between -90 and 90.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the minimum Angle is greater than the maximum Angle.
System.ArgumentException	Thrown when either the minimum Angle or the maximum Angle is out of the angle range of -90 to 90.

ConfigureIL(Double, Double)

Declaration

```
public void ConfigureIL(double ilMin, double ilMax)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Double	ilMin	
System.Double	ilMax	

ConfigureILReference(Double, Double)

Declaration

```
public void ConfigureILReference(double ilReferenceMin, double ilReferenceMax)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Double	ilReferenceMin	
System.Double	ilReferenceMax	

ConfigureMaximumPower(Double, Double)

Configure the value range for the maximum power returned by ReadPERAsync(Int32, Int32, CancellationToken) and ReadMaximumPowerAsync(Int32, Int32, CancellationToken).

Declaration

```
public void ConfigureMaximumPower(double maximumPowerMin, double maximumPowerMax)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Double	maximumPowerMin	The minimum Maximum Power value.
System.Double	maximumPowerMax	The maximum Maximum Power value.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the minimum PER is greater than the maximum PER.

ConfigurePER(Double, Double)

Configure the value range for the PER returned by ReadPERAsync(Int32, Int32, CancellationToken).

Declaration

```
public void ConfigurePER(double perMin, double perMax)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Double	perMin	The minimum PER value.
System.Double	perMax	The maximum PER value.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the minimum PER is greater than the maximum PER.

ConfigurePower(Double, Double)

Declaration

```
public void ConfigurePower(double powerMin, double powerMax)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Double	powerMin	
System.Double	powerMax	

DarkAllDetectorsAsync(CancellationToken)

Declaration

```
public async Task DarkAllDetectorsAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

DarkDetectorAsync(Int32, CancellationToken)

Declaration

```
public Task DarkDetectorAsync(int detector, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

GetActiveLaserAsync(CancellationToken)

Declaration

```
public async Task<int> GetActiveLaserAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Int32>	

GetAutoPowerAdjustmentStateAsync(CancellationToken)

Declaration

```
public async Task<EAutoPowerAdjustmentState> GetAutoPowerAdjustmentStateAsync(CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<EAutoPowerAdjustmentState>	

GetDetectorDarkStateAsync(Int32, CancellationToken)

Declaration

```
public Task<EDarkState> GetDetectorDarkStateAsync(int detector, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<EDarkState>	

GetILReferenceAsync(Int32, Int32, CancellationToken)

Declaration

```
public async Task<double> GetILReferenceAsync(int detector, int wavelength, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

GetOutputChannelAsync(CancellationToken)

Declaration

```
public async Task<int> GetOutputChannelAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Int32>	

GetPERModeAsync(CancellationToken)

Declaration

```
public async Task<EPERMode> GetPERModeAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<EPERMode>	

GetSwitchChannelAsync(Int32, CancellationToken)

Declaration

```
public async Task<int> GetSwitchChannelAsync(int switchIndex, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	switchIndex	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Int32>	

ReadILAsync(Int32, Int32, CancellationToken)

Declaration

```
public Task<double> ReadILAsync(int detector, int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

ReadMaximumPowerAsync(Int32, Int32, CancellationToken)

Declaration

```
public async Task<double> ReadMaximumPowerAsync(int detector, int wavelength, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Int32	wavelength	

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

ReadMinimumPowerAsync(Int32, Int32, CancellationToken)

Declaration

```
public async Task<double> ReadMinimumPowerAsync(int detector, int wavelength, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

ReadPERAsync(Int32, Int32, CancellationToken)

Declaration

```
public async Task<PERReading> ReadPERAsync(int detector, int wavelength, CancellationToken cancellationToken
= default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<PERReading>	

ReferenceILAsync(Int32, Int32, Boolean, CancellationToken)

Declaration

```
public async Task<double> ReferenceILAsync(int detector, int wavelength, bool applyReferenceToAllDetectors,
CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Int32	wavelength	
System.Boolean	applyReferenceToAllDetectors	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

ReferenceILAsync(Int32, Int32, CancellationToken)

Declaration

```
public async Task<double> ReferenceILAsync(int detector, int wavelength, CancellationToken cancellationToken
= default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

SetAutoPowerAdjustmentStateAsync(EAutoPowerAdjustmentState, CancellationToken)

Declaration

```
public async Task SetAutoPowerAdjustmentStateAsync(EAutoPowerAdjustmentState autoPowerAdjustmentState,
CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
EAutoPowerAdjustmentState	autoPowerAdjustmentState	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetILReferenceAsync(Int32, Int32, Double, CancellationToken)

Declaration

```
public Task SetILReferenceAsync(int detector, int wavelength, double reference, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Int32	wavelength	
System.Double	reference	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetPERModeAsync(EPERMode, CancellationToken)

Declaration

```
public async Task SetPERModeAsync(EPERMode perMode, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
EPERMode	perMode	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

TurnOffLaserAsync(CancellationToken)

Declaration

```
public async Task TurnOffLaserAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

TurnOnLaserAsync(Int32, CancellationToken)

Declaration

```
public async Task TurnOnLaserAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Explicit Interface Implementations

IDetectorDevice.Detectors

Declaration

```
IReadOnlyList<IModule> IDetectorDevice.Detectors { get; }
```

Returns

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<IModule>	

Implements

ISimulatedDevice
IPEM
ISantecDevice
ILaserSource
IWavelengthDevice
IDetectorDeviceWithDarking
ISwitchController
IFiberDevice
IDevice<TModelType>
IInsertionLossMeter
IDetectorDevice
IDevice
System.IDisposable

Class SimulatedPLM

Provides a simulated PLM with configurable reading values.

> **NOTE**
>
> This class is intended for testing/development purposes. It operates much quicker than an actual PLM. It can only be created using a SimulatedDeviceFactory. It should only be used when returned as part of a list of devices by a DetectHardwareAsync(EConnectionTypesToDetect) or DetectHardwareAsync<T>(EConnectionTypesToDetect) call to a SimulatedHardwareDetectionService. Each reading value contains a default value range but can be configured via the appropriate configuration method.

Inheritance

System.Object
SimulatedSantecDevice
SimulatedPLM

Implements

ISimulatedDevice
IPLM
ISantecDevice
IDiscretePowerMeter
ISwitchController
IFiberDevice
IPolarizationController
IDevice<EPLMModel>
IPowerMeter
IInsertionLossMeter
IBackreflectionMeter
IDetectorDeviceWithDarking
IDetectorDevice
ILaserSourceWithMonitor
ILaserSource
IWavelengthDevice
IDevice
System.IDisposable

Inherited Members

SimulatedSantecDevice.creatingDevice
SimulatedSantecDevice.Dispose()
SimulatedSantecDevice.SerialNumber
SimulatedSantecDevice.FirmwareVersion
SimulatedSantecDevice.Address
SimulatedSantecDevice.IsInitialized
SimulatedSantecDevice.QueryResponses
SimulatedSantecDevice.ConfigureQueryResponses(IEnumerable<QueryResponsePair>)
SimulatedSantecDevice.GetTimeout()
SimulatedSantecDevice.SetTimeout(ETimeout)
SimulatedSantecDevice.GetInteractionModeAsync(CancellationToken)
SimulatedSantecDevice.GetIPAddressAsync(CancellationToken)
SimulatedSantecDevice.SetIPAddressAsync(IPAddress, CancellationToken)
SimulatedSantecDevice.EnableDHCPAsync(CancellationToken)
SimulatedSantecDevice.GetGatewayAsync(CancellationToken)
SimulatedSantecDevice.SetGatewayAsync(IPAddress, CancellationToken)

SimulatedSantecDevice.GetSubnetPrefixLengthAsync(CancellationToken)
SimulatedSantecDevice.SetSubnetPrefixLengthAsync(Int32, CancellationToken)
SimulatedSantecDevice.GetHostnameAsync(CancellationToken)
SimulatedSantecDevice.SetHostnameAsync(String, CancellationToken)
SimulatedSantecDevice.SetInteractionModeAsync(EInteractionMode, CancellationToken)
SimulatedSantecDevice.InitializeAsync()
SimulatedSantecDevice.PingAsync()
SimulatedSantecDevice.Uninitialize()
SimulatedSantecDevice.ResetAsync()
SimulatedSantecDevice.RefreshAsync()
SimulatedSantecDevice.QueryAsync(String, CancellationToken)
SimulatedSantecDevice.WriteAsync(String, CancellationToken)
SimulatedSantecDevice.ReadAsync(CancellationToken)
System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.Polarization

Assembly: Santec.Hardware.dll

Syntax

```
public class SimulatedPLM : SimulatedSantecDevice, ISimulatedDevice, IPLM, ISantecDevice,
IDiscretePowerMeter, ISwitchController, IFiberDevice, IPolarizationController, IDevice<EPLMModel>,
IPowerMeter, IInsertionLossMeter, IBackreflectionMeter, IDetectorDeviceWithDarking, IDetectorDevice,
ILaserSourceWithMonitor, ILaserSource, IWavelengthDevice, IDevice, IDisposable
```

Properties

BRMax

Declaration

```
public double BRMax { get; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The maximum BR reading value. The default value is `-65`.

BRMin

Declaration

```
public double BRMin { get; }
```

Property Value

TYPE	DESCRIPTION

TYPE	DESCRIPTION
System.Double	The minimum BR reading value. The default value is `-75`.

BRReferenceMax

Declaration

```
public double BRReferenceMax { get; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The maximum BR reference value. The default value is `-35`.

BRReferenceMin

Declaration

```
public double BRReferenceMin { get; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The minimum BR reference value. The default value is `-36`.

Detectors

Declaration

```
public IReadOnlyList<DeviceDetector> Detectors { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<DeviceDetector>	

Fiber

Declaration

```
public FiberInformation Fiber { get; }
```

Property Value

TYPE	DESCRIPTION
FiberInformation	

ILMax

Declaration

```
public double ILMax { get; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The maximum IL reading value. The default value is `0.15`.

ILMin

Declaration

```
public double ILMin { get; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The minimum IL reading value. The default value is `0`.

ILReferenceMax

Declaration

```
public double ILReferenceMax { get; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The maximum IL reference value. The default value is `5`.

ILReferenceMin

Declaration

```
public double ILReferenceMin { get; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The minimum IL reference value. The default value is `2`.

Model

Declaration

```
public EPLMModel Model { get; }
```

Property Value

TYPE	DESCRIPTION
EPLMModel	

MonitorPowerMax

Declaration

```
public double MonitorPowerMax { get; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The maximum monitor power value. The default value is `-30`.

MonitorPowerMin

Declaration

```
public double MonitorPowerMin { get; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The minimum monitor power value. The default value is `-30`.

Name

Declaration

```
public override string Name { get; }
```

Property Value

TYPE	DESCRIPTION
System.String	

Overrides

SimulatedSantecDevice.Name

NumberOfDetectors

Declaration

```
public int NumberOfDetectors { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

NumberOfOutputs

Declaration

```
public int NumberOfOutputs { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

PDLMax

Get the maximum PDL value returned by a PDL reading.

Declaration

```
public double PDLMax { get; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The maximum PDL reading value. The default value is `0.15`.

Remarks

Use ConfigurePDL(Double, Double) to set this value.

PDLMin

Get the minimum PDL value returned by a PDL reading.

Declaration

```
public double PDLMin { get; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The minimum PDL reading value. The default value is `0`.

Remarks

Use ConfigurePDL(Double, Double) to set this value.

PowerMax

Declaration

```
public double PowerMax { get; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The maximum power reading value. The default value is `-3`.

PowerMin

Declaration

```
public double PowerMin { get; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The minimum power reading value. The default value is `-3.15`.

SupportsTurningOffLaser

Declaration

```
public bool SupportsTurningOffLaser { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	

Remarks

> **NOTE**
>
> This property will always return `true`.

Switches

Declaration

```
public List<DeviceSwitch> Switches { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.List<DeviceSwitch>	

Wavelengths

Declaration

```
public List<int> Wavelengths { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.List<System.Int32>	

Methods

ChangeMultipleSwitchesChannelsAsync(IEnumerable<DeviceSwitchAndChannelPair>, CancellationToken)

Declaration

```
public async Task ChangeMultipleSwitchesChannelsAsync(IEnumerable<DeviceSwitchAndChannelPair>
deviceSwitchesAndChannels, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Collections.Generic.IEnumerable<DeviceSwitchAndChannelPair>	deviceSwitchesAndChannels	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

ChangeOutputChannelAsync(Int32, CancellationToken)

Declaration

```
public async Task ChangeOutputChannelAsync(int output, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	output	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

ChangeSwitchChannelAsync(Int32, Int32, CancellationToken)

Declaration

```
public async Task ChangeSwitchChannelAsync(int switchIndex, int channel, CancellationToken cancellationToken
= default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	switchIndex	
System.Int32	channel	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

ConfigureBR(Double, Double)

Declaration

```
public void ConfigureBR(double brMin, double brMax)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Double	brMin	
System.Double	brMax	

ConfigureBRReference(Double, Double)

Declaration

```
public void ConfigureBRReference(double brReferenceMin, double brReferenceMax)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Double	brReferenceMin	
System.Double	brReferenceMax	

ConfigureIL(Double, Double)

Declaration

```
public void ConfigureIL(double ilMin, double ilMax)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Double	ilMin	
System.Double	ilMax	

ConfigureILReference(Double, Double)

Declaration

```
public void ConfigureILReference(double ilReferenceMin, double ilReferenceMax)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Double	ilReferenceMin	
System.Double	ilReferenceMax	

ConfigureMonitorPower(Double, Double)

Declaration

```
public void ConfigureMonitorPower(double monitorPowerMin, double monitorPowerMax)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Double	monitorPowerMin	
System.Double	monitorPowerMax	

ConfigurePDL(Double, Double)

Configure the value range for the PDL returned by ReadPDLAsync(Int32, Int32, CancellationToken) or ReadPDLAndBRAsync(Int32, Int32, CancellationToken).

Declaration

```
public void ConfigurePDL(double pdlMin, double pdlMax)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Double	pdlMin	The minimum PDL value.
System.Double	pdlMax	The maximum PDL value.

ConfigurePower(Double, Double)

Declaration

```
public void ConfigurePower(double powerMin, double powerMax)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Double	powerMin	
System.Double	powerMax	

DarkAllDetectorsAsync(CancellationToken)

Declaration

```
public async Task DarkAllDetectorsAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

DarkBRDetectorAsync(CancellationToken)

Declaration

```
public Task DarkBRDetectorAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

DarkDetectorAsync(Int32, CancellationToken)

Declaration

```
public Task DarkDetectorAsync(int detector, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

GetActiveLaserAsync(CancellationToken)

Declaration

```
public async Task<int> GetActiveLaserAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Int32>	

GetBRReferenceAsync(Int32, CancellationToken)

Declaration

```
public async Task<double> GetBRReferenceAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

GetDetectorDarkStateAsync(Int32, CancellationToken)

Declaration

```
public Task<EDarkState> GetDetectorDarkStateAsync(int detector, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<EDarkState>	

GetFactoryPowerAsync(Int32, Int32, CancellationToken)

Declaration

```
public async Task<FactoryPowerReading> GetFactoryPowerAsync(int output, int wavelength, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	output	
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<FactoryPowerReading>	

GetILReferenceAsync(Int32, Int32, CancellationToken)

Declaration

```
public async Task<double> GetILReferenceAsync(int detector, int wavelength, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

GetOutputChannelAsync(CancellationToken)

Declaration

```
public async Task<int> GetOutputChannelAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Int32>	

GetPDLReferencesAsync(Int32, Int32, CancellationToken)

Declaration

```
public async Task<List<double>> GetPDLReferencesAsync(int detector, int wavelength, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Collections.Generic.List<System.Double>>	

GetPolarizationModeAsync(CancellationToken)

Declaration

```
public async Task<EPolarizationMode> GetPolarizationModeAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<EPolarizationMode>	

GetPolarizationStateAsync(CancellationToken)

Declaration

```
public async Task<EPolarizationState> GetPolarizationStateAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<EPolarizationState>	

GetSwitchChannelAsync(Int32, CancellationToken)

Declaration

```
public async Task<int> GetSwitchChannelAsync(int switchIndex, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	switchIndex	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Int32>	

GetWaveplatePositionAsync(Int32, CancellationToken)

Declaration

```
public async Task<double> GetWaveplatePositionAsync(int waveplateIndex, CancellationToken cancellationToken
= default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	waveplateIndex	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

ReadBRAsync(Int32, CancellationToken)

Declaration

```
public async Task<ReadingValue> ReadBRAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<ReadingValue>	

ReadILAsync(Int32, Int32, CancellationToken)

Declaration

```
public async Task<double> ReadILAsync(int detector, int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

ReadMonitorPowerAsync(Int32, CancellationToken)

Declaration

```
public async Task<double> ReadMonitorPowerAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

ReadMultipleILsAsync(IEnumerable<Int32>, Int32, CancellationToken)

Declaration

```
public async Task<List<double>> ReadMultipleILsAsync(IEnumerable<int> detectors, int wavelength,
CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Collections.Generic.IEnumerable<System.Int32>	detectors	
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Collections.Generic.List<System.Double>>	

ReadMultiplePowersAsync(IEnumerable<Int32>, Int32, CancellationToken)

Declaration

```
public async Task<List<double>> ReadMultiplePowersAsync(IEnumerable<int> detectors, int wavelength,
CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Collections.Generic.IEnumerable<System.Int32>	detectors	
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Collections.Generic.List<System.Double>>	

ReadPDLAndBRAsync(Int32, Int32, CancellationToken)

Declaration

```
public async Task<PDLAndBRReading> ReadPDLAndBRAsync(int detector, int wavelength, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<PDLAndBRReading>	

ReadPDLAsync(Int32, Int32, CancellationToken)

Declaration

```
public async Task<PDLReading> ReadPDLAsync(int detector, int wavelength, CancellationToken cancellationToken
= default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<PDLReading>	

ReadPowerAsync(Int32, Int32, CancellationToken)

Declaration

```
public async Task<double> ReadPowerAsync(int detector, int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

ReferenceBRAsync(Int32, CancellationToken)

Declaration

```
public async Task<double> ReferenceBRAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

ReferenceILAsync(Int32, Int32, Boolean, CancellationToken)

Declaration

```
public async Task<double> ReferenceILAsync(int detector, int wavelength, bool applyReferenceToAllDetectors,
CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Int32	wavelength	
System.Boolean	applyReferenceToAllDetectors	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

ReferenceILAsync(Int32, Int32, CancellationToken)

Declaration

```
public async Task<double> ReferenceILAsync(int detector, int wavelength, CancellationToken cancellationToken
= default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

ReferencePDLAsync(Int32, Int32, CancellationToken)

Declaration

```
public async Task<List<double>> ReferencePDLAsync(int detector, int wavelength, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Collections.Generic.List<System.Double>>	

SetBRReferenceAsync(Int32, Double, CancellationToken)

Declaration

```
public async Task SetBRReferenceAsync(int wavelength, double reference, CancellationToken cancellationToken
= default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	
System.Double	reference	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetILReferenceAsync(Int32, Int32, Double, CancellationToken)

Declaration

```
public async Task SetILReferenceAsync(int detector, int wavelength, double reference, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Int32	wavelength	
System.Double	reference	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetPolarizationModeAsync(EPolarizationMode, CancellationToken)

Declaration

```
public async Task SetPolarizationModeAsync(EPolarizationMode polarizationMode, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
EPolarizationMode	polarizationMode	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetPolarizationStateAsync(EPolarizationState, CancellationToken)

Declaration

```
public async Task SetPolarizationStateAsync(EPolarizationState polarizationState, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
EPolarizationState	polarizationState	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetWaveplatePositionAsync(Int32, Double, CancellationToken)

Declaration

```
public async Task SetWaveplatePositionAsync(int waveplateIndex, double position, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	waveplateIndex	
System.Double	position	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

TurnOffLaserAsync(CancellationToken)

Declaration

```
public async Task TurnOffLaserAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

TurnOnLaserAsync(Int32, CancellationToken)

Declaration

```
public async Task TurnOnLaserAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Explicit Interface Implementations

IDetectorDevice.Detectors

Declaration

```
IReadOnlyList<IModule> IDetectorDevice.Detectors { get; }
```

Returns

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<IModule>	

Implements

ISimulatedDevice
IPLM
ISantecDevice
IDiscretePowerMeter
ISwitchController
IFiberDevice
IPolarizationController
IDevice<TModelType>
IPowerMeter
IInsertionLossMeter

IBackreflectionMeter
IDetectorDeviceWithDarking
IDetectorDevice
ILaserSourceWithMonitor
ILaserSource
IWavelengthDevice
IDevice
System.IDisposable

Namespace Santec.Hardware.Devices.Power

Classes

GainRange

Represents a gain range of a power meter/power meter module.

OPM

Provides a connection to an OPM along with properties of the device and operations that can be performed using it.

> **NOTE**
>
> This class cannot not be used directly. It can only be used when returned:
>
> 1. As part of a list of devices by a DetectHardwareAsync(EConnectionTypesToDetect) or DetectHardwareAsync<T>(EConnectionTypesToDetect) call.
> 2. As a device by a TryDetectIPDeviceAsync(IPAddress) or TryDetectIPDeviceAsync<T>(IPAddress) call.

Interfaces

IContinuousPowerMeter

Defines the properties of a power meter with a continuous wavelength range and the operations that can be performed with a connection to the device.

IDiscretePowerMeter

Defines the properties of a power meter with discrete wavelengths and the operations that can be performed with a connection to the device.

IOPM

Defines the properties of an OPM and operations that can be performed with a connection to the device.

IPowerMeter

Defines the properties of a power meter and the operations that can be performed with a connection to the device.

IPowerMeterWithReferencePersistence

Defines the properties of a Power Meter that supports persisting a reference.

Enums

EAutoGainRangeModeState

Specifies the auto gain range mode state of a power meter module.

ELoggingStatus

Specifies the logging status of a device.

EOPMModel

Specifies the model of an OPM.

EReadingMode

Specifies the reading mode of a power meter module.

EResolution

Specifies the reading resolution of a power meter module.

ETriggerBehaviourMode

Specifies how a device responds to a trigger.

Enum EAutoGainRangeModeState

Specifies the auto gain range mode state of a power meter module.

Namespace: Santec.Hardware.Devices.Power

Assembly: Santec.Hardware.dll

Syntax

```
public enum EAutoGainRangeModeState
```

Fields

NAME	DESCRIPTION
Off	Auto gain range mode off.
On	Auto gain range mode on.

Enum ELoggingStatus

Specifies the logging status of a device.

Namespace: Santec.Hardware.Devices.Power

Assembly: Santec.Hardware.dll

Syntax

```
public enum ELoggingStatus
```

Fields

NAME	DESCRIPTION
Complete	Logging complete.
InProgess	Logging in progress.
WaitingForTrigger	Waiting for trigger.

Enum EOPMModel

Specifies the model of an OPM.

Namespace: Santec.Hardware.Devices.Power

Assembly: Santec.Hardware.dll

Syntax

```
public enum EOPMModel
```

Fields

NAME	DESCRIPTION
OPM200	The OPM-200 model.

Enum EReadingMode

Specifies the reading mode of a power meter module.

Namespace: Santec.Hardware.Devices.Power

Assembly: Santec.Hardware.dll

Syntax

```
public enum EReadingMode
```

Fields

NAME	DESCRIPTION
IL	IL readings.
Power	Power readings.
PowerInWatts	Power readings in watts.

Enum EResolution

Specifies the reading resolution of a power meter module.

Namespace: Santec.Hardware.Devices.Power

Assembly: Santec.Hardware.dll

Syntax

```
public enum EResolution
```

Fields

NAME	DESCRIPTION
OneDigit	One digit resolution.
ThreeDigits	Three digit resolution.
TwoDigits	Two digit resolution.

Enum ETriggerBehaviourMode

Specifies how a device responds to a trigger.

Namespace: Santec.Hardware.Devices.Power

Assembly: Santec.Hardware.dll

Syntax

```
public enum ETriggerBehaviourMode
```

Fields

NAME	DESCRIPTION
FreeRun	Measurements are taken continuously after the first trigger.
Ignore	Triggers are ignored.
SingleMeasurement	A single measurement is performed for each trigger.

Class GainRange

Represents a gain range of a power meter/power meter module.

Inheritance

System.Object
GainRange

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.Power

Assembly: Santec.Hardware.dll

Syntax

```
public class GainRange
```

Properties

LowerLimit

Get the lower limit of the gain range.

Declaration

```
public int LowerLimit { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	The lower limit of the gain range.

UpperLimit

Get the upper limit of the gain range.

Declaration

```
public int UpperLimit { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	The upper limit of the gain range.

Interface IContinuousPowerMeter

Defines the properties of a power meter with a continuous wavelength range and the operations that can be performed with a connection to the device.

Inherited Members

IPowerMeter.ReadPowerAsync(Int32, Int32, CancellationToken)
IInsertionLossMeter.GetILReferenceAsync(Int32, Int32, CancellationToken)
IInsertionLossMeter.SetILReferenceAsync(Int32, Int32, Double, CancellationToken)
IInsertionLossMeter.ReferenceILAsync(Int32, Int32, CancellationToken)
IInsertionLossMeter.ReferenceILAsync(Int32, Int32, Boolean, CancellationToken)
IInsertionLossMeter.ReadILAsync(Int32, Int32, CancellationToken)
IDetectorDevice.NumberOfDetectors
IDetectorDevice.Detectors
IDevice.Name
IDevice.SerialNumber
IDevice.FirmwareVersion
IDevice.Address
IDevice.IsInitialized
IDevice.InitializeAsync()
IDevice.Uninitialize()
IDevice.GetTimeout()
IDevice.SetTimeout(ETimeout)
IDevice.ResetAsync()
IDevice.RefreshAsync()
IDevice.PingAsync()
IDevice.QueryAsync(String, CancellationToken)
IDevice.WriteAsync(String, CancellationToken)
IDevice.ReadAsync(CancellationToken)
System.IDisposable.Dispose()

Namespace: Santec.Hardware.Devices.Power

Assembly: Santec.Hardware.dll

Syntax

```
public interface IContinuousPowerMeter : IPowerMeter, IInsertionLossMeter, IDetectorDevice, IDevice,
IDisposable
```

Interface IDiscretePowerMeter

Defines the properties of a power meter with discrete wavelengths and the operations that can be performed with a connection to the device.

Inherited Members

IPowerMeter.ReadPowerAsync(Int32, Int32, CancellationToken)
IInsertionLossMeter.GetILReferenceAsync(Int32, Int32, CancellationToken)
IInsertionLossMeter.SetILReferenceAsync(Int32, Int32, Double, CancellationToken)
IInsertionLossMeter.ReferenceILAsync(Int32, Int32, CancellationToken)
IInsertionLossMeter.ReferenceILAsync(Int32, Int32, Boolean, CancellationToken)
IInsertionLossMeter.ReadILAsync(Int32, Int32, CancellationToken)
IDetectorDevice.NumberOfDetectors
IDetectorDevice.Detectors
IWavelengthDevice.Wavelengths
IDevice.Name
IDevice.SerialNumber
IDevice.FirmwareVersion
IDevice.Address
IDevice.IsInitialized
IDevice.InitializeAsync()
IDevice.Uninitialize()
IDevice.GetTimeout()
IDevice.SetTimeout(ETimeout)
IDevice.ResetAsync()
IDevice.RefreshAsync()
IDevice.PingAsync()
IDevice.QueryAsync(String, CancellationToken)
IDevice.WriteAsync(String, CancellationToken)
IDevice.ReadAsync(CancellationToken)
System.IDisposable.Dispose()

Namespace: Santec.Hardware.Devices.Power

Assembly: Santec.Hardware.dll

Syntax

```
public interface IDiscretePowerMeter : IPowerMeter, IInsertionLossMeter, IDetectorDevice, IWavelengthDevice,
IDevice, IDisposable
```

Methods

ReadMultipleILsAsync(IEnumerable<Int32>, Int32, CancellationToken)

Asynchronously read the insertion loss from multiple detectors at a given wavelength.

Declaration

```
Task<List<double>> ReadMultipleILsAsync(IEnumerable<int> detectors, int wavelength, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION

TYPE	NAME	DESCRIPTION
System.Collections.Generic.IEnumerable<System.Int32>	detectors	The detectors to read the insertion loss from.
System.Int32	wavelength	The wavelength to read the insertion loss at.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Collections.Generic.List<System.Double>>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the device does not support the specified wavelength.
System.ArgumentException	Thrown when any of the specified detectors are out of range for the device.
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device read multiple ILs query response does not match the expected format.
UnexpectedDeviceResponseException	Thrown when any of the detector power responses are not valid numbers.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

ReadMultiplePowersAsync(IEnumerable<Int32>, Int32, CancellationToken)

Asynchronously read the power from multiple detectors at a given wavelength.

Declaration

```
Task<List<double>> ReadMultiplePowersAsync(IEnumerable<int> detectors, int wavelength, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Collections.Generic.IEnumerable<System.Int32>	detectors	The detectors to read from.
System.Int32	wavelength	The wavelength to read the power at.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Collections.Generic.List<System.Double>>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the device does not support the specified wavelength.
System.ArgumentException	Thrown when any of the specified detectors are out of range for the device.
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device read multiple powers query response does not match the expected format.
UnexpectedDeviceResponseException	Thrown when any of the detector power responses are not valid numbers.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

Interface IOPM

Defines the properties of an OPM and operations that can be performed with a connection to the device.

Inherited Members

ISantecDevice.GetIPAddressAsync(CancellationToken)
ISantecDevice.SetIPAddressAsync(IPAddress, CancellationToken)
ISantecDevice.EnableDHCPAsync(CancellationToken)
ISantecDevice.GetGatewayAsync(CancellationToken)
ISantecDevice.SetGatewayAsync(IPAddress, CancellationToken)
ISantecDevice.GetSubnetPrefixLengthAsync(CancellationToken)
ISantecDevice.SetSubnetPrefixLengthAsync(Int32, CancellationToken)
ISantecDevice.GetHostnameAsync(CancellationToken)
ISantecDevice.SetHostnameAsync(String, CancellationToken)
ISantecDevice.GetInteractionModeAsync(CancellationToken)
ISantecDevice.SetInteractionModeAsync(EInteractionMode, CancellationToken)
IDetectorDeviceWithDarking.DarkAllDetectorsAsync(CancellationToken)
IDetectorDeviceWithDarking.DarkDetectorAsync(Int32, CancellationToken)
IDetectorDeviceWithDarking.GetDetectorDarkStateAsync(Int32, CancellationToken)
IModuleController.NumberOfModules
IModuleController.Modules
IPowerMeter.ReadPowerAsync(Int32, Int32, CancellationToken)
IInsertionLossMeter.GetILReferenceAsync(Int32, Int32, CancellationToken)
IInsertionLossMeter.SetILReferenceAsync(Int32, Int32, Double, CancellationToken)
IInsertionLossMeter.ReferenceILAsync(Int32, Int32, CancellationToken)
IInsertionLossMeter.ReferenceILAsync(Int32, Int32, Boolean, CancellationToken)
IInsertionLossMeter.ReadILAsync(Int32, Int32, CancellationToken)
IDetectorDevice.NumberOfDetectors
IDevice<EOPMModel>.Model
IDevice.Name
IDevice.SerialNumber
IDevice.FirmwareVersion
IDevice.Address
IDevice.IsInitialized
IDevice.InitializeAsync()
IDevice.Uninitialize()
IDevice.GetTimeout()
IDevice.SetTimeout(ETimeout)
IDevice.ResetAsync()
IDevice.RefreshAsync()
IDevice.PingAsync()
IDevice.QueryAsync(String, CancellationToken)
IDevice.WriteAsync(String, CancellationToken)
IDevice.ReadAsync(CancellationToken)
System.IDisposable.Dispose()

Namespace: Santec.Hardware.Devices.Power

Assembly: Santec.Hardware.dll

Syntax

```
public interface IOPM : ISantecDevice, IDetectorDeviceWithDarking, IModuleController, IContinuousPowerMeter,
IPowerMeter, IInsertionLossMeter, IDetectorDevice, IDevice<EOPMModel>, IDevice, IDisposable
```

Properties

Detectors

Get the detectors of the OPM.

> **NOTE**
>
> The OPM must be initialized before this property contains the proper values.

Declaration

```
IReadOnlyList<IOPMModule> Detectors { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<IOPMModule>	A list of the detectors the OPM contains.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the property is accessed before the OPM has been initialized.

Methods

GetDetectorAveragingTimeAsync(Int32, CancellationToken)

Asynchronously get the averaging time, in milliseconds, for a device detector.

Declaration

```
Task<double> GetDetectorAveragingTimeAsync(int detector, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	The detector number of the detector to get the averaging time for.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the specified detector is out of range for the device.
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device averaging time response is not a valid averaging time.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

GetDetectorGainRangeAsync(Int32, CancellationToken)

Asynchronously get the gain range for a device detector.

Declaration

```
Task<GainRange> GetDetectorGainRangeAsync(int detector, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	The detector number of the detector to get the gain range for.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<GainRange>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the specified detector is out of range for the device.
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device gain range response does not match the expected response.
UnexpectedDeviceResponseException	Thrown when the device gain range response does not correspond to a detector gain range.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

GetDetectorReadingModeAsync(Int32, CancellationToken)

Asynchronously get the reading mode for a device detector.

Declaration

```
Task<EReadingMode> GetDetectorReadingModeAsync(int detector, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	The detector number of the detector to get the reading mode for.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<EReadingMode>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the specified detector is out of range for the device.
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device reading mode response does not correspond to a reading mode.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

GetDetectorResolutionAsync(Int32, CancellationToken)

Asynchronously get the reading resolution for a device detector.

Declaration

```
Task<EResolution> GetDetectorResolutionAsync(int detector, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	The detector number of the detector to get the resolution for.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<EResolution>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the specified detector is out of range for the device.

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device response does not correspond to a resolution.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

GetDetectorWavelengthAsync(Int32, CancellationToken)

Asynchronously get the wavelength for a device detector.

Declaration

```
Task<int> GetDetectorWavelengthAsync(int detector, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	The detector number of the detector to get the wavelength for.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Int32>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the specified detector is out of range for the device.
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.

TYPE	CONDITION
UnexpectedDeviceResponseException	Thrown when the device wavelength response is not a valid wavelength.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

LogDetectorAsync(Int32, Int32, GainRange, Double, ETriggerEdge, ETriggerBehaviourMode, CancellationToken)

Asynchronously perform a logging operation for a device detector.

Declaration

```
Task<IReadOnlyList<double>> LogDetectorAsync(int detector, int numberOfPoints, GainRange gainRange, double
averagingTime, ETriggerEdge triggerEdge, ETriggerBehaviourMode triggerBehaviourMode, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	The detector number of the detector to perform the logging operation for.
System.Int32	numberOfPoints	The number of points for the logging operation.
GainRange	gainRange	The gain range to use for the logging operation.
System.Double	averagingTime	The averaging time, in milliseconds, for the logging operation.
ETriggerEdge	triggerEdge	The trigger edge setting for the logging operation.
ETriggerBehaviourMode	triggerBehaviourMode	The trigger behaviour mode for the logging operation.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Collections.Generic.IReadOnlyList<System.Double>>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the specified detector is out of range for the device.
System.ArgumentException	Thrown when the OPM detector does not support the specified gain range.
System.ArgumentException	Thrown when the specified number of logging points is out of range for the detector.
System.ArgumentException	Thrown when the specified averaging time is out of range for the detector.
System.ArgumentException	Thrown when the specified trigger behaviour mode is ignore triggers.
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when a logging response does not match the expected response.
UnexpectedDeviceResponseException	Thrown when a logging response is not a valid number.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

LogDetectorAsync(Int32, Int32, Int32, GainRange, Double, ETriggerEdge, ETriggerBehaviourMode, CancellationToken)

Asynchronously perform a logging operation for a device detector.

Declaration

```
Task<IReadOnlyList<double>> LogDetectorAsync(int detector, int numberOfPoints, int wavelength, GainRange gainRange, double averagingTime, ETriggerEdge triggerEdge, ETriggerBehaviourMode triggerBehaviourMode, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	The detector number of the detector to perform the logging operation for.

TYPE	NAME	DESCRIPTION
System.Int32	numberOfPoints	The number of points for the logging operation.
System.Int32	wavelength	The wavelength to perform the logging operation at.
GainRange	gainRange	The gain range to use for the logging operation.
System.Double	averagingTime	The averaging time, in milliseconds, for the logging operation.
ETriggerEdge	triggerEdge	The trigger edge setting for the logging operation.
ETriggerBehaviourMode	triggerBehaviourMode	The trigger behaviour mode for the logging operation.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Collections.Generic.IReadOnlyList<System.Double>>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the specified detector is out of range for the device.
System.ArgumentException	Thrown when the specified wavelength is out of range for the OPM detector.
System.ArgumentException	Thrown when the OPM detector does not support the specified gain range.

TYPE	CONDITION
System.ArgumentException	Thrown when the specified number of logging points is out of range for the detector.
System.ArgumentException	Thrown when the specified averaging time is out of range for the detector.
System.ArgumentException	Thrown when the specified trigger behaviour mode is ignore triggers.
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when a logging response does not match the expected response.
UnexpectedDeviceResponseException	Thrown when a logging response is not a valid number.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

ReadAllDetectorsAsync(Int32, CancellationToken)

Asynchronously read the power/power in watts/IL from all detectors at a given wavelength.

Declaration

```
Task<List<double>> ReadAllDetectorsAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	The wavelength to read the powers/powers in watts/ILs at.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Collections.Generic.List<System.Double>>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the specified wavelength is out of range for the device.
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device read all query response does not match the expected format.
UnexpectedDeviceResponseException	Thrown when any of the detector responses are not valid numbers.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

ReadAllDetectorsAsync(CancellationToken)

Asynchronously read the power/power in watts/IL from all detectors at the currently selected wavelength.

Declaration

```
Task<List<double>> ReadAllDetectorsAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Collections.Generic.List<System.Double>>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device read all query response does not match the expected format.
UnexpectedDeviceResponseException	Thrown when any of the detector responses are not valid numbers.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

ReadAllILsAsync(Int32, CancellationToken)

Asynchronously read the insertion loss from all detectors at a given wavelength.

Declaration

```
Task<List<double>> ReadAllILsAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	The wavelength to read the ILs at.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Collections.Generic.List<System.Double>>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the specified wavelength is out of range for the device.

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device read all powers query response does not match the expected format.
UnexpectedDeviceResponseException	Thrown when any of the detector IL responses are not valid numbers.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

ReadAllILsAsync(CancellationToken)

Asynchronously read the insertion loss from all detectors at the currently selected wavelength.

Declaration

```
Task<List<double>> ReadAllILsAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Collections.Generic.List<System.Double>>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device read all powers query response does not match the expected format.

TYPE	CONDITION
UnexpectedDeviceResponseException	Thrown when any of the detector IL responses are not valid numbers.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

ReadAllPowersAsync(Int32, CancellationToken)

Asynchronously read the power from all detectors at a given wavelength.

Declaration

```
Task<List<double>> ReadAllPowersAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	The wavelength to read the powers at.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Collections.Generic.List<System.Double>>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the specified wavelength is out of range for the device.
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device read all powers query response does not match the expected format.

TYPE	CONDITION
UnexpectedDeviceResponseException	Thrown when any of the detector power responses are not valid numbers.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

ReadAllPowersAsync(CancellationToken)

Asynchronously read the power from all detectors at the currently selected wavelength.

Declaration

```
Task<List<double>> ReadAllPowersAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Collections.Generic.List<System.Double>>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device read all powers query response does not match the expected format.
UnexpectedDeviceResponseException	Thrown when any of the detector power responses are not valid numbers.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

ReadAllPowersInWattsAsync(Int32, CancellationToken)

Asynchronously read the power in watts from all detectors at a given wavelength.

Declaration

```
Task<List<double>> ReadAllPowersInWattsAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	The wavelength to read the powers at.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Collections.Generic.List<System.Double>>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the specified wavelength is out of range for the device.
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device read all powers in watts query response does not match the expected format.
UnexpectedDeviceResponseException	Thrown when any of the detector power responses are not valid numbers.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

ReadAllPowersInWattsAsync(CancellationToken)

Asynchronously read the power in watts from all detectors at the currently selected wavelength.

Declaration

```
Task<List<double>> ReadAllPowersInWattsAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Collections.Generic.List<System.Double>>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device read all powers in watts query response does not match the expected format.
UnexpectedDeviceResponseException	Thrown when any of the detector power responses are not valid numbers.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

ReadDetectorAsync(Int32, Int32, CancellationToken)

Asynchronously read the power/power in watts/IL from a device detector at a given wavelength.

Declaration

```
Task<double> ReadDetectorAsync(int detector, int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION

TYPE	NAME	DESCRIPTION
System.Int32	detector	The detector number of the detector to read from.
System.Int32	wavelength	The wavelength to read the power/power in watts/IL at.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the specified wavelength is out of range for the device.
System.ArgumentException	Thrown when the specified detector is out of range for the device.
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device read response is not a valid number.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

ReadDetectorAsync(Int32, CancellationToken)

Asynchronously read the power/power in watts/IL from a device detector at the currently selected wavelength.

Declaration

```
Task<double> ReadDetectorAsync(int detector, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	The detector number of the detector to read from.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the specified detector is out of range for the device.
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device read response is not a valid number.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

ReadILAsync(Int32, CancellationToken)

Asynchronously read the IL from a device detector at the currently selected wavelength.

Declaration

```
Task<double> ReadILAsync(int detector, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	The detector number of the detector to read from.

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the specified detector is out of range for the device.
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device read IL response is not a valid number.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

ReadPowerAsync(Int32, CancellationToken)

Asynchronously read the power from a device detector at the currently selected wavelength.

Declaration

```
Task<double> ReadPowerAsync(int detector, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	The detector number of the detector to read from.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the specified detector is out of range for the device.
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device read power response is not a valid number.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

ReadPowerInWattsAsync(Int32, Int32, CancellationToken)

Asynchronously read the power in watts from a device detector at a given wavelength.

Declaration

```
Task<double> ReadPowerInWattsAsync(int detector, int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	The detector number of the detector to read from.
System.Int32	wavelength	The wavelength to read the power in watts at.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the specified wavelength is out of range for the device.
System.ArgumentException	Thrown when the specified detector is out of range for the device.
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device read response is not a valid number.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

ReadPowerInWattsAsync(Int32, CancellationToken)

Asynchronously read the power in watts from a device detector at the currently selected wavelength.

Declaration

```
Task<double> ReadPowerInWattsAsync(int detector, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	The detector number of the detector to read from.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the specified detector is out of range for the device.
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device read power in watts response is not a valid number.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

ReferenceILAsync(Int32, Boolean, CancellationToken)

Asynchronously perform an IL reference for a device detector at the currently selected wavelength.

Declaration

```
Task<double> ReferenceILAsync(int detector, bool applyReferenceToAllDetectors, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	The detector number of the detector to perform the reference for.
System.Boolean	applyReferenceToAllDetectors	Indicate if the reference read should be applied to all detectors.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the specified detector is out of range for the device.
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device read IL response is not a valid number.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

ReferenceILAsync(Int32, CancellationToken)

Asynchronously perform an IL reference for a device detector at the currently selected wavelength.

Declaration

```
Task<double> ReferenceILAsync(int detector, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	The detector number of the detector to perform the reference for.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the specified detector is out of range for the device.
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device read IL response is not a valid number.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

SendDetectorSoftwareTriggerAsync(Int32, CancellationToken)

Asynchronously send a software trigger to a device detector.

Declaration

```
Task SendDetectorSoftwareTriggerAsync(int detector, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	The detector number of the detector to send the software trigger to.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the specified detector is out of range for the device.
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when a software trigger response does not match the expected response.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

SetAllDetectorAveragingTimesAsync(Double, CancellationToken)

Asynchronously set the averaging time, in milliseconds, for all device detectors.

Declaration

```
Task SetAllDetectorAveragingTimesAsync(double averagingTime, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Double	averagingTime	The averaging time, in milliseconds, to set all detectors to.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the specified averaging time is out of range for the device.

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device set all detector resolutions response does not match the expected response.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

SetAllDetectorReadingModesAsync(EReadingMode, CancellationToken)

Asynchronously set the reading mode for all device detectors.

Declaration

```
Task SetAllDetectorReadingModesAsync(EReadingMode readingMode, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
EReadingMode	readingMode	The reading mode to set the detectors to.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device set all detector reading modes response does not match the expected response.

TYPE	CONDITION
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

SetAllDetectorResolutionsAsync(EResolution, CancellationToken)

Asynchronously set the reading resolution for all device detectors.

Declaration

```
Task SetAllDetectorResolutionsAsync(EResolution resolution, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
EResolution	resolution	The resolution to set all the detectors to.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device set all detector resolutions response does not match the expected response.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

SetAllDetectorWavelengthsAsync(Int32, CancellationToken)

Asynchronously set the wavelength for all device detectors.

Declaration

```
Task SetAllDetectorWavelengthsAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	The wavelength to set all the detectors to.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the specified wavelength is out of range for the device.
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device set all detector wavelengths response does not match the expected response.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

SetDetectorAveragingTimeAsync(Int32, Double, CancellationToken)

Asynchronously set the averaging time, in milliseconds, for a device detector.

Declaration

```
Task SetDetectorAveragingTimeAsync(int detector, double averagingTime, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	The detector number of the detector to set the resolution for.
System.Double	averagingTime	The averaging time, in milliseconds, to set the detector to.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the specified averaging time is out of range for the device.
System.ArgumentException	Thrown when the specified detector is out of range for the device.
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device set device set detector resolution response does not match the expected response.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

SetDetectorGainRangeAsync(Int32, GainRange, CancellationToken)

Asynchronously set the gain range for a device detector.

Declaration

```
Task SetDetectorGainRangeAsync(int detector, GainRange gainRange, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	The detector number of the detector to set the reading mode for.
GainRange	gainRange	The gain range to set the detector to.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the specified detector is out of range for the device.
System.ArgumentException	Thrown when the specified detector does not support the specified gain range.
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device set gain range response does not match the expected response.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

SetDetectorReadingModeAsync(Int32, EReadingMode, CancellationToken)

Asynchronously set the reading mode for a device detector.

Declaration

```
Task SetDetectorReadingModeAsync(int detector, EReadingMode readingMode, CancellationToken cancellationToken
= default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	The detector number of the detector to set the reading mode for.
EReadingMode	readingMode	The reading mode to set the detector to.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the specified detector is out of range for the device.
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device set reading mode response does not match the expected response.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

SetDetectorResolutionAsync(Int32, EResolution, CancellationToken)

Asynchronously set the reading resolution for a device detector.

Declaration

```
Task SetDetectorResolutionAsync(int detector, EResolution resolution, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	The detector number of the detector to set the resolution for.
EResolution	resolution	The resolution to set the detector to.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the specified detector is out of range for the device.
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device set device set detector resolution response does not match the expected response.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

SetDetectorWavelengthAsync(Int32, Int32, CancellationToken)

Asynchronously set the wavelength for a device detector.

Declaration

```
Task SetDetectorWavelengthAsync(int detector, int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	The detector number of the detector to set the wavelength for.
System.Int32	wavelength	The wavelength to set the detector to.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the specified wavelength is out of range for the device.
System.ArgumentException	Thrown when the specified detector is out of range for the device.
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device set device wavelength response does not match the expected response.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

SetILReferenceAsync(Int32, Double, CancellationToken)

Asynchronously set the IL reference for a device detector at the currently selected wavelength.

Declaration

```
Task SetILReferenceAsync(int detector, double reference, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	The detector number of the detector to set the IL reference for.
System.Double	reference	The IL reference value.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	The task representing the asynchronous operation.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the specified detector is out of range for the device.
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device IL reference query response does not match the expected response.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

Interface IPowerMeter

Defines the properties of a power meter and the operations that can be performed with a connection to the device.

Inherited Members

IInsertionLossMeter.GetILReferenceAsync(Int32, Int32, CancellationToken)
IInsertionLossMeter.SetILReferenceAsync(Int32, Int32, Double, CancellationToken)
IInsertionLossMeter.ReferenceILAsync(Int32, Int32, CancellationToken)
IInsertionLossMeter.ReferenceILAsync(Int32, Int32, Boolean, CancellationToken)
IInsertionLossMeter.ReadILAsync(Int32, Int32, CancellationToken)
IDetectorDevice.NumberOfDetectors
IDetectorDevice.Detectors
IDevice.Name
IDevice.SerialNumber
IDevice.FirmwareVersion
IDevice.Address
IDevice.IsInitialized
IDevice.InitializeAsync()
IDevice.Uninitialize()
IDevice.GetTimeout()
IDevice.SetTimeout(ETimeout)
IDevice.ResetAsync()
IDevice.RefreshAsync()
IDevice.PingAsync()
IDevice.QueryAsync(String, CancellationToken)
IDevice.WriteAsync(String, CancellationToken)
IDevice.ReadAsync(CancellationToken)
System.IDisposable.Dispose()

Namespace: Santec.Hardware.Devices.Power

Assembly: Santec.Hardware.dll

Syntax

```
public interface IPowerMeter : IInsertionLossMeter, IDetectorDevice, IDevice, IDisposable
```

Methods

ReadPowerAsync(Int32, Int32, CancellationToken)

Asynchronously read the power from an device detector at a given wavelength.

Declaration

```
Task<double> ReadPowerAsync(int detector, int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	The detector number of the detector to read from.
System.Int32	wavelength	The wavelength to read the power at.

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the device does not support the specified wavelength.
System.ArgumentException	Thrown when the specified detector is out of range for the device.
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device read power response is not a valid number.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

Interface IPowerMeterWithReferencePersistence

Defines the properties of a Power Meter that supports persisting a reference.

Inherited Members

IPowerMeter.ReadPowerAsync(Int32, Int32, CancellationToken)
IInsertionLossMeter.GetILReferenceAsync(Int32, Int32, CancellationToken)
IInsertionLossMeter.SetILReferenceAsync(Int32, Int32, Double, CancellationToken)
IInsertionLossMeter.ReferenceILAsync(Int32, Int32, CancellationToken)
IInsertionLossMeter.ReferenceILAsync(Int32, Int32, Boolean, CancellationToken)
IInsertionLossMeter.ReadILAsync(Int32, Int32, CancellationToken)
IDetectorDevice.NumberOfDetectors
IDetectorDevice.Detectors
IDevice.Name
IDevice.SerialNumber
IDevice.FirmwareVersion
IDevice.Address
IDevice.IsInitialized
IDevice.InitializeAsync()
IDevice.Uninitialize()
IDevice.GetTimeout()
IDevice.SetTimeout(ETimeout)
IDevice.ResetAsync()
IDevice.RefreshAsync()
IDevice.PingAsync()
IDevice.QueryAsync(String, CancellationToken)
IDevice.WriteAsync(String, CancellationToken)
IDevice.ReadAsync(CancellationToken)
System.IDisposable.Dispose()

Namespace: Santec.Hardware.Devices.Power

Assembly: Santec.Hardware.dll

Syntax

```
public interface IPowerMeterWithReferencePersistence : IPowerMeter, IInsertionLossMeter, IDetectorDevice,
IDevice, IDisposable
```

Properties

RequiresILReferenceSave

A flag indicating whether the device requires saving the reference.

Declaration

```
bool RequiresILReferenceSave { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	

Methods

SaveILReferencesAsync(CancellationToken)

Asynchronously set a persistent reference. This method sets the current reference to persist even after the device is rebooted.

Declaration

```
Task SaveILReferencesAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

IMPORTANT

A reference should be set before calling this method.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized RLM.
System.NotSupportedException	Thrown when the device does not support this operation.
UnexpectedDeviceResponseException	Thrown when the device returns an unexpected response.

Class OPM

Provides a connection to an OPM along with properties of the device and operations that can be performed using it.

> **NOTE**
>
> This class cannot not be used directly. It can only be used when returned:
>
> 1. As part of a list of devices by a DetectHardwareAsync(EConnectionTypesToDetect) or DetectHardwareAsync<T>(EConnectionTypesToDetect) call.
> 2. As a device by a TryDetectIPDeviceAsync(IPAddress) or TryDetectIPDeviceAsync<T>(IPAddress) call.

Inheritance

System.Object
PhysicalDevice
SantecDevice
OPM

Implements

IOPM
ISantecDevice
IDetectorDeviceWithDarking
IModuleController
IContinuousPowerMeter
IPowerMeter
IInsertionLossMeter
IDetectorDevice
IDevice<EOPMModel>
IDevice
System.IDisposable

Inherited Members

SantecDevice.HandleIDNResponse(String[])
SantecDevice.GetIPAddressAsync(CancellationToken)
SantecDevice.SetIPAddressAsync(IPAddress, CancellationToken)
SantecDevice.EnableDHCPAsync(CancellationToken)
SantecDevice.GetGatewayAsync(CancellationToken)
SantecDevice.SetGatewayAsync(IPAddress, CancellationToken)
SantecDevice.GetSubnetPrefixLengthAsync(CancellationToken)
SantecDevice.SetSubnetPrefixLengthAsync(Int32, CancellationToken)
SantecDevice.GetHostnameAsync(CancellationToken)
SantecDevice.SetHostnameAsync(String, CancellationToken)
SantecDevice.GetInteractionModeAsync(CancellationToken)
SantecDevice.SetInteractionModeAsync(EInteractionMode, CancellationToken)
PhysicalDevice.Address
PhysicalDevice.SerialNumber
PhysicalDevice.FirmwareVersion
PhysicalDevice.IsInitialized
PhysicalDevice.IsInitializing
PhysicalDevice.SetTimeout(ETimeout)
PhysicalDevice.GetTimeout()
PhysicalDevice.Dispose()
PhysicalDevice.Dispose(Boolean)

PhysicalDevice.PingAsync()
PhysicalDevice.ResetAsync()
PhysicalDevice.QueryAsync(String, CancellationToken)
PhysicalDevice.WriteAsync(String, CancellationToken)
PhysicalDevice.ReadAsync(CancellationToken)
System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.Power

Assembly: Santec.Hardware.dll

Syntax

```
public class OPM : SantecDevice, IOPM, ISantecDevice, IDetectorDeviceWithDarking, IModuleController,
IContinuousPowerMeter, IPowerMeter, IInsertionLossMeter, IDetectorDevice, IDevice<EOPMModel>, IDevice,
IDisposable
```

Properties

Detectors

Declaration

```
public IReadOnlyList<IOPMModule> Detectors { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<IOPMModule>	

Model

Declaration

```
public EOPMModel Model { get; }
```

Property Value

TYPE	DESCRIPTION
EOPMModel	

Modules

Declaration

```
public IReadOnlyList<IModule> Modules { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<IModule>	

Name

Declaration

```
public override string Name { get; }
```

Property Value

TYPE	DESCRIPTION
System.String	

Overrides

PhysicalDevice.Name

Remarks

> **NOTE**
>
> This property will always return "OPM".

NumberOfDetectors

Declaration

```
public int NumberOfDetectors { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

Remarks

> **NOTE**
>
> This property is an alias for NumberOfModules.

NumberOfModules

Declaration

```
public int NumberOfModules { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

Methods

DarkAllDetectorsAsync(CancellationToken)

Declaration

```
public Task DarkAllDetectorsAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

DarkDetectorAsync(Int32, CancellationToken)

Declaration

```
public Task DarkDetectorAsync(int detector, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

GetDetectorAveragingTimeAsync(Int32, CancellationToken)

Declaration

```
public Task<double> GetDetectorAveragingTimeAsync(int detector, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

GetDetectorDarkStateAsync(Int32, CancellationToken)

Declaration

```
public Task<EDarkState> GetDetectorDarkStateAsync(int detector, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<EDarkState>	

GetDetectorGainRangeAsync(Int32, CancellationToken)

Declaration

```
public Task<GainRange> GetDetectorGainRangeAsync(int detector, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<GainRange>	

GetDetectorReadingModeAsync(Int32, CancellationToken)

Declaration

```
public Task<EReadingMode> GetDetectorReadingModeAsync(int detector, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<EReadingMode>	

GetDetectorResolutionAsync(Int32, CancellationToken)

Declaration

```
public Task<EResolution> GetDetectorResolutionAsync(int detector, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<EResolution>	

GetDetectorWavelengthAsync(Int32, CancellationToken)

Declaration

```
public Task<int> GetDetectorWavelengthAsync(int detector, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Int32>	

GetILReferenceAsync(Int32, Int32, CancellationToken)

Declaration

```
public Task<double> GetILReferenceAsync(int detector, int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

InitializeAsync()

Declaration

```
public override async Task InitializeAsync()
```

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Overrides

SantecDevice.InitializeAsync()

LogDetectorAsync(Int32, Int32, GainRange, Double, ETriggerEdge, ETriggerBehaviourMode, CancellationToken)

Declaration

```
public Task<IReadOnlyList<double>> LogDetectorAsync(int detector, int numberOfPoints, GainRange gainRange,
double averagingTime, ETriggerEdge triggerEdge, ETriggerBehaviourMode triggerBehaviourMode,
CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Int32	numberOfPoints	
GainRange	gainRange	
System.Double	averagingTime	
ETriggerEdge	triggerEdge	
ETriggerBehaviourMode	triggerBehaviourMode	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Collections.Generic.IReadOnlyList<System.Double>>	

LogDetectorAsync(Int32, Int32, Int32, GainRange, Double, ETriggerEdge, ETriggerBehaviourMode, CancellationToken)

Declaration

```
public Task<IReadOnlyList<double>> LogDetectorAsync(int detector, int numberOfPoints, int wavelength,
GainRange gainRange, double averagingTime, ETriggerEdge triggerEdge, ETriggerBehaviourMode
triggerBehaviourMode, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Int32	numberOfPoints	
System.Int32	wavelength	
GainRange	gainRange	
System.Double	averagingTime	
ETriggerEdge	triggerEdge	
ETriggerBehaviourMode	triggerBehaviourMode	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Collections.Generic.IReadOnlyList<System.Double>>	

ReadAllDetectorsAsync(Int32, CancellationToken)

Declaration

```
public Task<List<double>> ReadAllDetectorsAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Collections.Generic.List<System.Double>>	

ReadAllDetectorsAsync(CancellationToken)

Declaration

```
public Task<List<double>> ReadAllDetectorsAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Collections.Generic.List<System.Double>>	

ReadAllILsAsync(Int32, CancellationToken)

Declaration

```
public Task<List<double>> ReadAllILsAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Collections.Generic.List<System.Double>>	

ReadAllILsAsync(CancellationToken)

Declaration

```
public Task<List<double>> ReadAllILsAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Collections.Generic.List<System.Double>>	

ReadAllPowersAsync(Int32, CancellationToken)

Declaration

```
public Task<List<double>> ReadAllPowersAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Collections.Generic.List<System.Double>>	

ReadAllPowersAsync(CancellationToken)

Declaration

```
public Task<List<double>> ReadAllPowersAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Collections.Generic.List<System.Double>>	

ReadAllPowersInWattsAsync(Int32, CancellationToken)

Declaration

```
public Task<List<double>> ReadAllPowersInWattsAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Collections.Generic.List<System.Double>>	

ReadAllPowersInWattsAsync(CancellationToken)

Declaration

```
public Task<List<double>> ReadAllPowersInWattsAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Collections.Generic.List<System.Double>>	

ReadDetectorAsync(Int32, Int32, CancellationToken)

Declaration

```
public Task<double> ReadDetectorAsync(int detector, int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

ReadDetectorAsync(Int32, CancellationToken)

Declaration

```
public Task<double> ReadDetectorAsync(int detector, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

ReadILAsync(Int32, Int32, CancellationToken)

Declaration

```
public Task<double> ReadILAsync(int detector, int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

ReadILAsync(Int32, CancellationToken)

Declaration

```
public Task<double> ReadILAsync(int detector, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

ReadPowerAsync(Int32, Int32, CancellationToken)

Declaration

```
public Task<double> ReadPowerAsync(int detector, int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

ReadPowerAsync(Int32, CancellationToken)

Declaration

```
public Task<double> ReadPowerAsync(int detector, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

ReadPowerInWattsAsync(Int32, Int32, CancellationToken)

Declaration

```
public Task<double> ReadPowerInWattsAsync(int detector, int wavelength, CancellationToken cancellationToken
= default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Int32	wavelength	

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

ReadPowerInWattsAsync(Int32, CancellationToken)

Declaration

```
public Task<double> ReadPowerInWattsAsync(int detector, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

ReferenceILAsync(Int32, Boolean, CancellationToken)

Declaration

```
public async Task<double> ReferenceILAsync(int detector, bool applyReferenceToAllDetectors,
CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Boolean	applyReferenceToAllDetectors	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

ReferenceILAsync(Int32, Int32, Boolean, CancellationToken)

Declaration

```
public async Task<double> ReferenceILAsync(int detector, int wavelength, bool applyReferenceToAllDetectors,
CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Int32	wavelength	
System.Boolean	applyReferenceToAllDetectors	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

ReferenceILAsync(Int32, Int32, CancellationToken)

Declaration

```
public Task<double> ReferenceILAsync(int detector, int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

ReferenceILAsync(Int32, CancellationToken)

Declaration

```
public Task<double> ReferenceILAsync(int detector, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

RefreshAsync()

Declaration

```
public override async Task RefreshAsync()
```

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Overrides

PhysicalDevice.RefreshAsync()

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized RLM.
UnexpectedDeviceResponseException	Thrown when the OPM modules response is not a valid module count.

SendDetectorSoftwareTriggerAsync(Int32, CancellationToken)

Declaration

```
public Task SendDetectorSoftwareTriggerAsync(int detector, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetAllDetectorAveragingTimesAsync(Double, CancellationToken)

Declaration

```
public async Task SetAllDetectorAveragingTimesAsync(double averagingTime, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Double	averagingTime	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetAllDetectorReadingModesAsync(EReadingMode, CancellationToken)

Declaration

```
public async Task SetAllDetectorReadingModesAsync(EReadingMode readingMode, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
EReadingMode	readingMode	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetAllDetectorResolutionsAsync(EResolution, CancellationToken)

Declaration

```
public async Task SetAllDetectorResolutionsAsync(EResolution resolution, CancellationToken cancellationToken
= default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
EResolution	resolution	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetAllDetectorWavelengthsAsync(Int32, CancellationToken)

Declaration

```
public async Task SetAllDetectorWavelengthsAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetDetectorAveragingTimeAsync(Int32, Double, CancellationToken)

Declaration

```
public Task SetDetectorAveragingTimeAsync(int detector, double averagingTime, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Double	averagingTime	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetDetectorGainRangeAsync(Int32, GainRange, CancellationToken)

Declaration

```
public Task SetDetectorGainRangeAsync(int detector, GainRange gainRange, CancellationToken cancellationToken
= default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
GainRange	gainRange	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetDetectorReadingModeAsync(Int32, EReadingMode, CancellationToken)

Declaration

```
public Task SetDetectorReadingModeAsync(int detector, EReadingMode readingMode, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
EReadingMode	readingMode	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetDetectorResolutionAsync(Int32, EResolution, CancellationToken)

Declaration

```
public Task SetDetectorResolutionAsync(int detector, EResolution resolution, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
EResolution	resolution	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetDetectorWavelengthAsync(Int32, Int32, CancellationToken)

Declaration

```
public Task SetDetectorWavelengthAsync(int detector, int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetILReferenceAsync(Int32, Double, CancellationToken)

Declaration

```
public Task SetILReferenceAsync(int detector, double reference, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Double	reference	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetILReferenceAsync(Int32, Int32, Double, CancellationToken)

Declaration

```
public Task SetILReferenceAsync(int detector, int wavelength, double reference, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Int32	wavelength	
System.Double	reference	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Uninitialize()

Declaration

```
public override void Uninitialize()
```

Overrides

PhysicalDevice.Uninitialize()

Explicit Interface Implementations

IDetectorDevice.Detectors

Declaration

```
IReadOnlyList<IModule> IDetectorDevice.Detectors { get; }
```

Returns

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<IModule>	

Implements

IOPM
ISantecDevice
IDetectorDeviceWithDarking
IModuleController
IContinuousPowerMeter
IPowerMeter
IInsertionLossMeter
IDetectorDevice
IDevice<TModelType>
IDevice
System.IDisposable

Namespace Santec.Hardware.Devices.Requirements

Classes

HardwareRequirementFailure<T>

Represents a failure to meet a hardware requirement.

HardwareRequirementsEvaluationResult<T>

Represents the result of evaluating hardware requirements.

SystemHardwareRequirementsEvaluationResult<T>

Represents the result of evaluating system-level hardware requirements across multiple devices.

Class HardwareRequirementFailure<T>

Represents a failure to meet a hardware requirement.

Inheritance

System.Object
HardwareRequirementFailure<T>

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.Requirements

Assembly: Santec.Hardware.dll

Syntax

```
public class HardwareRequirementFailure<T>

    where T : Enum
```

Type Parameters

NAME	DESCRIPTION
T	The type of the hardware requirement failure reason.

Constructors

HardwareRequirementFailure(T, String)

Initializes a new instance of the HardwareRequirementFailure<T> class.

Declaration

```
public HardwareRequirementFailure(T reason, string message)
```

Parameters

TYPE	NAME	DESCRIPTION
T	reason	The reason for the hardware requirement failure.
System.String	message	The failure message.

Properties

Message

Gets the hardware requirement failure message.

Declaration

```
public string Message { get; }
```

Property Value

TYPE	DESCRIPTION
System.String	The hardware requirement failure message.

Reason

Gets the reason for the hardware requirement failure.

Declaration

```
public T Reason { get; }
```

Property Value

TYPE	DESCRIPTION
T	The reason for the hardware requirement failure.

Class HardwareRequirementsEvaluationResult<T>

Represents the result of evaluating hardware requirements.

Inheritance

System.Object
HardwareRequirementsEvaluationResult<T>

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.Requirements

Assembly: Santec.Hardware.dll

Syntax

```
public class HardwareRequirementsEvaluationResult<T>

    where T : Enum
```

Type Parameters

NAME	DESCRIPTION
T	The type of the hardware requirement failure reason.

Constructors

HardwareRequirementsEvaluationResult(IDevice, Boolean, IReadOnlyList<HardwareRequirementFailure<T>>)

Initializes a new instance of the HardwareRequirementsEvaluationResult<T> class.

Declaration

```
public HardwareRequirementsEvaluationResult(IDevice device, bool deviceMeetsRequirements,
IReadOnlyList<HardwareRequirementFailure<T>> requirementFailures)
```

Parameters

TYPE	NAME	DESCRIPTION
IDevice	device	The device evaluated.
System.Boolean	deviceMeetsRequirements	Whether the device meets the hardware requirements.

TYPE	NAME	DESCRIPTION
System.Collections.Generic.IReadOnlyList<HardwareRequirementFailure<T>>	requirementFailures	Details of why the device failed to meet the hardware requirements.

Properties

Device

Gets the device evaluated.

Declaration

```
public IDevice Device { get; }
```

Property Value

TYPE	DESCRIPTION
IDevice	The evaluated device.

DeviceMeetsRequirements

Gets whether the device meets the hardware requirements.

Declaration

```
public bool DeviceMeetsRequirements { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	`true` if the device meets the hardware requirements; otherwise, `false`.

RequirementFailures

Gets a list of the details of why the device failed to meet the hardware requirements.

> **NOTE**
>
> This list will be empty if the device meets the hardware requirements.

Declaration

```
public IReadOnlyList<HardwareRequirementFailure<T>> RequirementFailures { get; }
```

Property Value

TYPE	DESCRIPTION

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<HardwareRequirementFailure<T>>	A read-only list of the details of why the device failed to meet the hardware requirements.

Class SystemHardwareRequirementsEvaluationResult<T>

Represents the result of evaluating system-level hardware requirements across multiple devices.

Inheritance

System.Object
SystemHardwareRequirementsEvaluationResult<T>

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.Requirements

Assembly: Santec.Hardware.dll

Syntax

```
public class SystemHardwareRequirementsEvaluationResult<T>

    where T : Enum
```

Type Parameters

NAME	DESCRIPTION
T	The type of the hardware requirement failure reason.

Constructors

SystemHardwareRequirementsEvaluationResult(IReadOnlyList<IDevice>, Boolean, IReadOnlyList<HardwareRequirementFailure<T>>)

Initializes a new instance of the SystemHardwareRequirementsEvaluationResult<T> class.

Declaration

```
public SystemHardwareRequirementsEvaluationResult(IReadOnlyList<IDevice> devices, bool
systemMeetsRequirements, IReadOnlyList<HardwareRequirementFailure<T>> requirementFailures)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Collections.Generic.IReadOnlyList<IDevice>	devices	The devices evaluated.
System.Boolean	systemMeetsRequirements	Whether the system meets the hardware requirements.

TYPE	NAME	DESCRIPTION
System.Collections.Generic.IReadOnlyList<HardwareRequirementFailure<T>>	requirementFailures	Details of why the system failed to meet the hardware requirements.

Properties

Devices

Gets the devices evaluated.

Declaration

```
public IReadOnlyList<IDevice> Devices { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<IDevice>	A read-only list of the evaluated devices.

RequirementFailures

Gets a list of the details of why the system failed to meet the hardware requirements.

> **NOTE**
>
> This list will be empty if the system meets the hardware requirements.

Declaration

```
public IReadOnlyList<HardwareRequirementFailure<T>> RequirementFailures { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<HardwareRequirementFailure<T>>	A read-only list of the details of why the system failed to meet the hardware requirements.

SystemMeetsRequirements

Gets whether the system meets the hardware requirements.

Declaration

```
public bool SystemMeetsRequirements { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	`true` if the system meets the hardware requirements; otherwise, `false`.

Namespace Santec.Hardware.Devices.ReturnLoss

Classes

RLM

Provides a connection to an RLM along with properties of the device and operations that can be performed using it.

> **NOTE**
>
> This class cannot not be used directly. It can only be used when returned:
>
> 1. As part of a list of devices by a DetectHardwareAsync(EConnectionTypesToDetect) or DetectHardwareAsync<T>(EConnectionTypesToDetect) call.
> 2. As a device by a TryDetectIPDeviceAsync(IPAddress) or TryDetectIPDeviceAsync<T>(IPAddress) call.

RLReading

Represents the results from an RLM return loss reading.

> **NOTE**
>
> This class should not be used directly. It should only be used when returned by an RLM return loss measurement.

SimulatedRLM

Provides a simulated RLM with configurable reading values.

> **NOTE**
>
> This class is intended for testing/development purposes. It operates much quicker than an actual RLM. It can only be created using a SimulatedDeviceFactory. It should only be used when returned as part of a list of devices by a DetectHardwareAsync(EConnectionTypesToDetect) or DetectHardwareAsync<T>(EConnectionTypesToDetect) call to a SimulatedHardwareDetectionService. Each reading value contains a default value range but can be configured via the appropriate configuration method.

Interfaces

IRLM

Defines the properties of an RLM and operations that can be performed with a connection to the device.

Enums

EGainMode

Specifies the gain mode of an RLM.

ELengthMode

Specifies the length mode of an RLM.

EPositionBMode

Specifies the RL position B mode of an RLM.

ERLMModel

Specifies the model of an RLM.

ERLMode

Specifies the RL mode of an RLM.

Enum EGainMode

Specifies the gain mode of an RLM.

Namespace: Santec.Hardware.Devices.ReturnLoss

Assembly: Santec.Hardware.dll

Syntax

```
public enum EGainMode
```

Fields

NAME	DESCRIPTION
Low	Low gain mode.
Normal	Normal gain mode.

Enum ELengthMode

Specifies the length mode of an RLM.

Namespace: Santec.Hardware.Devices.ReturnLoss

Assembly: Santec.Hardware.dll

Syntax

```
public enum ELengthMode
```

Fields

NAME	DESCRIPTION
LessThan100m	Lengths less than 100m.
LessThan1500m	Lengths less than 1500m.
LessThan4km	Lengths less than 4km.
LessThan500m	Lengths less than 500m.

Enum EPositionBMode

Specifies the RL position B mode of an RLM.

Namespace: Santec.Hardware.Devices.ReturnLoss

Assembly: Santec.Hardware.dll

Syntax

```
public enum EPositionBMode
```

Fields

NAME	DESCRIPTION
EOF	RL position B specified from end of fiber.
Zero	RL position B specified from the RLM output.

Enum ERLMModel

Specifies the model of an RLM.

Namespace: Santec.Hardware.Devices.ReturnLoss

Assembly: Santec.Hardware.dll

Syntax

```
public enum ERLMModel
```

Fields

NAME	DESCRIPTION
RLM100	The RLM-100 model.
RLM110	The RLM-110 model.

Enum ERLMode

Specifies the RL mode of an RLM.

Namespace: Santec.Hardware.Devices.ReturnLoss

Assembly: Santec.Hardware.dll

Syntax

```
public enum ERLMode
```

Fields

NAME	DESCRIPTION
Fast	Fast RL mode.
MaximumResolution	Maximum resolution RL mode.
Standard	Standard RL mode.
Unknown	Unknown custom RL mode.

Interface IRLM

Defines the properties of an RLM and operations that can be performed with a connection to the device.

Inherited Members

ISantecDevice.GetIPAddressAsync(CancellationToken)
ISantecDevice.SetIPAddressAsync(IPAddress, CancellationToken)
ISantecDevice.EnableDHCPAsync(CancellationToken)
ISantecDevice.GetGatewayAsync(CancellationToken)
ISantecDevice.SetGatewayAsync(IPAddress, CancellationToken)
ISantecDevice.GetSubnetPrefixLengthAsync(CancellationToken)
ISantecDevice.SetSubnetPrefixLengthAsync(Int32, CancellationToken)
ISantecDevice.GetHostnameAsync(CancellationToken)
ISantecDevice.SetHostnameAsync(String, CancellationToken)
ISantecDevice.GetInteractionModeAsync(CancellationToken)
ISantecDevice.SetInteractionModeAsync(EInteractionMode, CancellationToken)
IDiscretePowerMeter.ReadMultiplePowersAsync(IEnumerable<Int32>, Int32, CancellationToken)
IDiscretePowerMeter.ReadMultipleILsAsync(IEnumerable<Int32>, Int32, CancellationToken)
ILaserSourceWithMonitor.ReadMonitorPowerAsync(Int32, CancellationToken)
ILaserSourceWithMonitor.GetFactoryPowerAsync(Int32, Int32, CancellationToken)
ILaserSource.NumberOfOutputs
ILaserSource.SupportsTurningOffLaser
ILaserSource.GetActiveLaserAsync(CancellationToken)
ILaserSource.TurnOnLaserAsync(Int32, CancellationToken)
ILaserSource.TurnOffLaserAsync(CancellationToken)
ILaserSource.GetOutputChannelAsync(CancellationToken)
ILaserSource.ChangeOutputChannelAsync(Int32, CancellationToken)
IWavelengthDevice.Wavelengths
ISwitchController.Switches
ISwitchController.GetSwitchChannelAsync(Int32, CancellationToken)
ISwitchController.ChangeSwitchChannelAsync(Int32, Int32, CancellationToken)
ISwitchController.ChangeMultipleSwitchesChannelsAsync(IEnumerable<DeviceSwitchAndChannelPair>, CancellationToken)
IFiberDevice.Fiber
IPowerMeterWithReferencePersistence.RequiresILReferenceSave
IPowerMeterWithReferencePersistence.SaveILReferencesAsync(CancellationToken)
IPowerMeter.ReadPowerAsync(Int32, Int32, CancellationToken)
IInsertionLossMeter.GetILReferenceAsync(Int32, Int32, CancellationToken)
IInsertionLossMeter.SetILReferenceAsync(Int32, Int32, Double, CancellationToken)
IInsertionLossMeter.ReferenceILAsync(Int32, Int32, CancellationToken)
IInsertionLossMeter.ReferenceILAsync(Int32, Int32, Boolean, CancellationToken)
IInsertionLossMeter.ReadILAsync(Int32, Int32, CancellationToken)
IDetectorDevice.NumberOfDetectors
IDevice<ERLMModel>.Model
IDevice.Name
IDevice.SerialNumber
IDevice.FirmwareVersion
IDevice.Address
IDevice.IsInitialized
IDevice.InitializeAsync()
IDevice.Uninitialize()
IDevice.GetTimeout()
IDevice.SetTimeout(ETimeout)

IDevice.ResetAsync()
IDevice.RefreshAsync()
IDevice.PingAsync()
IDevice.QueryAsync(String, CancellationToken)
IDevice.WriteAsync(String, CancellationToken)
IDevice.ReadAsync(CancellationToken)
System.IDisposable.Dispose()

Namespace: Santec.Hardware.Devices.ReturnLoss

Assembly: Santec.Hardware.dll

Syntax

```
public interface IRLM : ISantecDevice, IDiscretePowerMeter, ILaserSourceWithMonitor, ILaserSource,
IWavelengthDevice, ISwitchController, IFiberDevice, IPowerMeterWithReferencePersistence, IPowerMeter,
IInsertionLossMeter, IDetectorDevice, IDevice<ERLMModel>, IDevice, IDisposable
```

Properties

Detectors

Get the detectors of the RLM.

> **NOTE**
>
> The RLM must be initialized before this property contains the proper values.

Declaration

```
IReadOnlyList<DeviceDetector> Detectors { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<DeviceDetector>	A list of the modules the attenuator contains.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the property is accessed before the RLM has been initialized.

SupportedLengthModes

Get the length modes supported by the RLM.

> **NOTE**
>
> The device must be initialized before this property contains the proper value.

Declaration

```
IReadOnlyList<ELengthMode> SupportedLengthModes { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<ELengthMode>	A read-only list of the supported length modes.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the property is accessed before the RLM has been initialized.

SupportedRLModes

Get the RL modes supported by the RLM.

> **NOTE**
>
> The device must be initialized before this property contains the proper value.

Declaration

```
IReadOnlyList<ERLMode> SupportedRLModes { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<ERLMode>	A read-only list of the supported RL modes.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the property is accessed before the RLM has been initialized.

SupportsILResolution

Get whether the RLM supports changing the IL resolution.

> **NOTE**
>
> The RLM firmware must be 02.03.00 or later to support changing the Il resolution.

> **NOTE**
>
> The default resolution is 2 digits. If the RLM supports changing the IL resolution, it can be changed to 3 digits.

> **NOTE**
>
> The device must be initialized before this property contains the proper value.

Declaration

```
bool SupportsILResolution { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	`true` if the RLM supports changing the IL resolution; otherwise, `false`.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the property is accessed before the RLM has been initialized.

SupportsMaximumResolutionRL

Get whether the RLM supports maximum resolution return loss.

> **NOTE**
>
> The RLM firmware must be 02.08.05 or later to support maximum resolution return loss.

> **NOTE**
>
> The device must be initialized before this property contains the proper value.

Declaration

```
bool SupportsMaximumResolutionRL { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	`true` if the RLM supports maximum resolution return loss; otherwise `false`.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the property is accessed before the RLM has been initialized.

Methods

GetDUTILAsync(Int32, CancellationToken)

Asynchronously get the DUT IL.

Declaration

```
Task<double> GetDUTILAsync(int wavelength, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	The wavelength to get the DUT IL for.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	The task representing the operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the RLM does not support the specified wavelength.
System.InvalidOperationException	Thrown when the method is called on an uninitialized RLM.
UnexpectedDeviceResponseException	Thrown when the device DUT IL response is not a valid number.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the RLM.

GetGainModeAsync(CancellationToken)

Asynchronously get the gain mode.

Declaration

```
Task<EGainMode> GetGainModeAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<EGainMode>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized RLM.
UnexpectedDeviceResponseException	Thrown when the device response does not correspond to a gain mode.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the RLM.

GetILResolutionAsync(CancellationToken)

Asynchronously get the IL reading resolution.

Declaration

```
Task<EResolution> GetILResolutionAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<EResolution>	The task representing the asynchronous operation.

Remarks

NOTE

The RLM must have firmware 02.03.00 or later to support this operation.

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
System.InvalidOperationException	Thrown when the RLM does not support IL resolution operations.
UnexpectedDeviceResponseException	Thrown when the device response does not correspond to a resolution.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

GetLengthModeAsync(CancellationToken)

Asynchronously get the length mode.

Declaration

```
Task<ELengthMode> GetLengthModeAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<ELengthMode>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized RLM.
UnexpectedDeviceResponseException	Thrown when the device response does not correspond to a length mode.
UnexpectedDeviceResponseException	Thrown when the device response does not correspond to a supported length mode.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the RLM.

GetPositionBModeAsync(CancellationToken)

Asynchronously get the RL position B mode.

Declaration

```
Task<EPositionBMode> GetPositionBModeAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<EPositionBMode>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized RLM.
UnexpectedDeviceResponseException	Thrown when the device response does not correspond to a position B mode.

TYPE	CONDITION
System.TimeoutException	Thrown when a communication failure causes no response to be received from the RLM.

GetRLModeAsync(CancellationToken)

Asynchronously get the RL sensitivity mode.

Declaration

```
Task<ERLMode> GetRLModeAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<ERLMode>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized RLM.
UnexpectedDeviceResponseException	Thrown when the device response does not correspond to an RL mode.
UnexpectedDeviceResponseException	Thrown when the device response does not correspond to a supported RL mode.
UnexpectedDeviceResponseException	Thrown when the device RL mode response does not have the correct format.
UnexpectedDeviceResponseException	Thrown when a device RL mode response does not match the expected response.

TYPE	CONDITION
UnexpectedDeviceResponseException	Thrown when a device RL mode response is not a valid number.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the RLM.

GetRLReferenceAsync(CancellationToken)

Asynchronously get the stored RL reference.

Declaration

```
Task<double> GetRLReferenceAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized RLM.
UnexpectedDeviceResponseException	Thrown when the device RL reference query response is not a valid RL reference value.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the RLM.

GetTraceDiagnosticsAsync(CancellationToken)

Asynchronously get the trace diagnostics for the last RL measurement.

Declaration

```
Task<TraceDiagnostics> GetTraceDiagnosticsAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<TraceDiagnostics>	The task representing the operation.

Remarks

NOTE

This operation will not block.

IMPORTANT

An RL measurement must be performed before the trace diagnostics can be retrieved. Otherwise, an exception will be thrown.

CAUTION

Do not use this method! This operation is currently not supported for USB connections. For Ethernet connections, use the TraceData API call instead of this method.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized RLM.
System.InvalidOperationException	Thrown when the method is called on an RLM that has no trace from an RL measurement.
UnexpectedDeviceResponseException	Thrown when the device trace diagnostics response does not match the expected format.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the RLM.
System.NotSupportedException	Thrown when the method is called on an RLM through a USB connection.

ReadRLAsync(Int32, Double, Double, CancellationToken)

Asynchronously read the return loss at a given wavelength using the specified reference lengths.

Declaration

```
Task<RLReading> ReadRLAsync(int wavelength, double lengthReferenceA, double lengthReferenceB,
CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	The wavelength to read the return loss at.
System.Double	lengthReferenceA	The reference length to use for position A.
System.Double	lengthReferenceB	The reference length to use for position B.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<RLReading>	The task representing the operation.

Remarks

The RL reading does not always return a value for RLa, RLb, RLtotal, or length. Depending on the setup, not all of these values can be read at the same time. This method will return `double.NaN` for the any value that cannot be read. Reference length B is specified from either the output panel of the RLM or from the measured end of fiber depending on the position B mode of the RLM. This setting can be retrieved using the GetPositionBModeAsync(CancellationToken) method and set using the SetPositionBModeAsync(EPositionBMode, CancellationToken) method.

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the RLM does not support the specified wavelength.
System.ArgumentException	Thrown when either RL reference length is less than `0`.

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized RLM.
UnexpectedDeviceResponseException	Thrown when the device read RL response does not match the expected format.
UnexpectedDeviceResponseException	Thrown when the RL reading length value is not a valid value.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the RLM.

ReadRLAsync(Int32, Double, CancellationToken)

Asynchronously read the return loss at a given wavelength using the specified reference length.

Declaration

```
Task<RLReading> ReadRLAsync(int wavelength, double lengthReference, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	The wavelength to read the return loss at.
System.Double	lengthReference	The reference length to use for the reading.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<RLReading>	The task representing the operation.

Remarks

The RL reading does not always return a value for RLa, RLb, RLtotal, or length. Depending on the setup, not all of these values can be read at the same time. This method will return `double.NaN` for the any value that cannot be read.

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the RLM does not support the specified wavelength.
System.ArgumentException	Thrown when the RLM reference length is less than `0`.
System.InvalidOperationException	Thrown when the method is called on an uninitialized RLM.
UnexpectedDeviceResponseException	Thrown when the device read RL response does not match the expected format.
UnexpectedDeviceResponseException	Thrown when the RL reading length value is not a valid value.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the RLM.

ReadRLAsync(Int32, CancellationToken)

Asynchronously read the return loss at a given wavelength.

Declaration

```
Task<RLReading> ReadRLAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	The wavelength to read the return loss at.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<RLReading>	The task representing the operation.

Remarks

The RL reading does not always return a value for RLa, RLb, RLtotal, or length. Depending on the setup, not all of these values can be read at the same time. This method will return `double.NaN` for the any value that cannot be read.

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the RLM does not support the specified wavelength.
System.InvalidOperationException	Thrown when the method is called on an uninitialized RLM.
UnexpectedDeviceResponseException	Thrown when the device read RL response does not match the expected format.
UnexpectedDeviceResponseException	Thrown when any of the RL reading values are not valid response values.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the RLM.

ReferenceRLAsync(CancellationToken)

Asynchronously perform a return loss length reference.

Declaration

```
Task<double> ReferenceRLAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	The task representing the operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized RLM.
UnexpectedDeviceResponseException	Thrown when the device reference RL response is not a valid number.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the RLM.

SetDUTILAsync(Int32, Double, CancellationToken)

Asynchronously set the DUT IL.

Declaration

```
Task SetDUTILAsync(int wavelength, double il, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	The wavelength to set the DUT ILfor.
System.Double	il	The DUT IL value.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	The task representing the asynchronous operation.

Remarks

The specified DUT IL must be greater than `0`.

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION

TYPE	CONDITION
System.ArgumentException	Thrown when the RLM does not support the specified wavelength.
System.ArgumentException	Thrown when the specified DUT IL is less than `0`.
System.InvalidOperationException	Thrown when the method is called on an uninitialized RLM.
UnexpectedDeviceResponseException	Thrown when the device DUT IL response does not match the expected response.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the RLM.

SetGainModeAsync(EGainMode, CancellationToken)

Asynchronously set the gain mode.

Declaration

```
Task SetGainModeAsync(EGainMode gainMode, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
EGainMode	gainMode	The gain mode to change to.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized RLM.
UnexpectedDeviceResponseException	Thrown when the device gain mode response does not match the expected response.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the RLM.

SetILResolutionAsync(EResolution, CancellationToken)

Asynchronously set the IL reading resolution.

Declaration

```
Task SetILResolutionAsync(EResolution resolution, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
EResolution	resolution	The resolution to set the detector to.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	The task representing the asynchronous operation.

Remarks

NOTE

The RLM must have firmware 02.03.00 or later to support this operation.

NOTE

The RLM only supports 2 digit and 3 digit IL resolution.

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the IL resolution is not supported by the RLM.
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
System.InvalidOperationException	Thrown when the RLM does not support IL resolution operations.
UnexpectedDeviceResponseException	Thrown when the device set IL resolution response does not match the expected response.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

SetLengthModeAsync(ELengthMode, CancellationToken)

Asynchronously set the length mode.

Declaration

```
Task SetLengthModeAsync(ELengthMode lengthMode, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
ELengthMode	lengthMode	The length mode to change to.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the length mode is not supported by the RLM.
System.InvalidOperationException	Thrown when the method is called on an uninitialized RLM.
UnexpectedDeviceResponseException	Thrown when the device length mode response does not match the expected response.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the RLM.

SetPositionBModeAsync(EPositionBMode, CancellationToken)

Asynchronously set the RL position B mode.

Declaration

```
Task SetPositionBModeAsync(EPositionBMode positionBMode, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
EPositionBMode	positionBMode	The position B mode to change to.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized RLM.

TYPE	CONDITION
UnexpectedDeviceResponseException	Thrown when the device positionB mode response does not match the expected response.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the RLM.

SetRLModeAsync(ERLMode, CancellationToken)

Asynchronously set the RL sensitivity mode.

Declaration

```
Task SetRLModeAsync(ERLMode rlMode, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
ERLMode	rlMode	The RL mode to change to.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the specified RL mode is `ERLMode.Unknown`.
System.ArgumentException	Thrown when the RL mode is not supported by the RLM.
System.InvalidOperationException	Thrown when the method is called on an uninitialized RLM.

TYPE	CONDITION
UnexpectedDeviceResponseException	Thrown when the device RL mode response does not have the correct format.
UnexpectedDeviceResponseException	Thrown when a device RL mode response does not match the expected response.
UnexpectedDeviceResponseException	Thrown when a device RL mode response is not a valid number.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the RLM.

SetRLReferenceAsync(Double, CancellationToken)

Asynchronously set the RL reference length.

Declaration

```
Task SetRLReferenceAsync(double reference, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Double	reference	The RL length reference value.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	The task representing the asynchronous operation.

Remarks

The reference value must be greater than or equal to `0`.

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the RL length reference value is less than `0`.

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized RLM.
UnexpectedDeviceResponseException	Thrown when the device RL reference query response does not match the expected response.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the RLM.

Extension Methods

RLMExtensions.IsMultimodeRLM100(IRLM)

Class RLM

Provides a connection to an RLM along with properties of the device and operations that can be performed using it.

> **NOTE**
>
> This class cannot not be used directly. It can only be used when returned:
>
> 1. As part of a list of devices by a DetectHardwareAsync(EConnectionTypesToDetect) or DetectHardwareAsync<T>(EConnectionTypesToDetect) call.
> 2. As a device by a TryDetectIPDeviceAsync(IPAddress) or TryDetectIPDeviceAsync<T>(IPAddress) call.

Inheritance

System.Object
PhysicalDevice
SantecDevice
RLM

Implements

IRLM
ISantecDevice
IDiscretePowerMeter
ILaserSourceWithMonitor
ILaserSource
IWavelengthDevice
ISwitchController
IFiberDevice
IPowerMeterWithReferencePersistence
IPowerMeter
IInsertionLossMeter
IDetectorDevice
IDevice<ERLMModel>
IDevice
System.IDisposable

Inherited Members

SantecDevice.GetIPAddressAsync(CancellationToken)
SantecDevice.SetIPAddressAsync(IPAddress, CancellationToken)
SantecDevice.EnableDHCPAsync(CancellationToken)
SantecDevice.GetGatewayAsync(CancellationToken)
SantecDevice.SetGatewayAsync(IPAddress, CancellationToken)
SantecDevice.GetSubnetPrefixLengthAsync(CancellationToken)
SantecDevice.SetSubnetPrefixLengthAsync(Int32, CancellationToken)
SantecDevice.GetHostnameAsync(CancellationToken)
SantecDevice.SetHostnameAsync(String, CancellationToken)
SantecDevice.GetInteractionModeAsync(CancellationToken)
SantecDevice.SetInteractionModeAsync(EInteractionMode, CancellationToken)
PhysicalDevice.Address
PhysicalDevice.SerialNumber
PhysicalDevice.FirmwareVersion
PhysicalDevice.IsInitialized
PhysicalDevice.IsInitializing
PhysicalDevice.SetTimeout(ETimeout)

PhysicalDevice.GetTimeout()
PhysicalDevice.Dispose()
PhysicalDevice.Dispose(Boolean)
PhysicalDevice.PingAsync()
PhysicalDevice.ResetAsync()
PhysicalDevice.QueryAsync(String, CancellationToken)
PhysicalDevice.WriteAsync(String, CancellationToken)
PhysicalDevice.ReadAsync(CancellationToken)
System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.ReturnLoss

Assembly: Santec.Hardware.dll

Syntax

```
public class RLM : SantecDevice, IRLM, ISantecDevice, IDiscretePowerMeter, ILaserSourceWithMonitor,
ILaserSource, IWavelengthDevice, ISwitchController, IFiberDevice, IPowerMeterWithReferencePersistence,
IPowerMeter, IInsertionLossMeter, IDetectorDevice, IDevice<ERLMModel>, IDevice, IDisposable
```

Properties

Detectors

Declaration

```
public IReadOnlyList<DeviceDetector> Detectors { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<DeviceDetector>	

Fiber

Declaration

```
public FiberInformation Fiber { get; }
```

Property Value

TYPE	DESCRIPTION
FiberInformation	

Model

Declaration

```
public ERLMModel Model { get; }
```

Property Value

TYPE	DESCRIPTION
ERLMModel	

Name

Declaration

```
public override string Name { get; }
```

Property Value

TYPE	DESCRIPTION
System.String	

Overrides

PhysicalDevice.Name

Remarks

> **NOTE**
>
> This property will always return "RLM".

NumberOfDetectors

Declaration

```
public int NumberOfDetectors { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

NumberOfOutputs

Declaration

```
public int NumberOfOutputs { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

RequiresILReferenceSave

Declaration

```
public bool RequiresILReferenceSave { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	

SupportedLengthModes

Declaration

```
public IReadOnlyList<ELengthMode> SupportedLengthModes { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<ELengthMode>	

SupportedRLModes

Declaration

```
public IReadOnlyList<ERLMode> SupportedRLModes { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<ERLMode>	

SupportsILResolution

Declaration

```
public bool SupportsILResolution { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	

SupportsMaximumResolutionRL

Declaration

```
public bool SupportsMaximumResolutionRL { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	

SupportsTurningOffLaser

Declaration

```
public bool SupportsTurningOffLaser { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	

Remarks

> **NOTE**
>
> This property will always return `true`.

Switches

Declaration

```
public List<DeviceSwitch> Switches { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.List<DeviceSwitch>	

Wavelengths

Declaration

```
public List<int> Wavelengths { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.List<System.Int32>	

Methods

ChangeMultipleSwitchesChannelsAsync(IEnumerable<DeviceSwitchAndChannelPair>, CancellationToken)

Declaration

```
public Task ChangeMultipleSwitchesChannelsAsync(IEnumerable<DeviceSwitchAndChannelPair>
deviceSwitchesAndChannels, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Collections.Generic.IEnumerable<DeviceSwitchAndChannelPair>	deviceSwitchesAndChannels	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

ChangeOutputChannelAsync(Int32, CancellationToken)

Declaration

```
public Task ChangeOutputChannelAsync(int output, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	output	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

ChangeSwitchChannelAsync(Int32, Int32, CancellationToken)

Declaration

```
public Task ChangeSwitchChannelAsync(int switchIndex, int channel, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	switchIndex	
System.Int32	channel	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

GetActiveLaserAsync(CancellationToken)

Declaration

```
public Task<int> GetActiveLaserAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Int32>	

GetDUTILAsync(Int32, CancellationToken)

Declaration

```
public async Task<double> GetDUTILAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

GetFactoryPowerAsync(Int32, Int32, CancellationToken)

Declaration

```
public Task<FactoryPowerReading> GetFactoryPowerAsync(int output, int wavelength, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	output	
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<FactoryPowerReading>	

GetGainModeAsync(CancellationToken)

Declaration

```
public async Task<EGainMode> GetGainModeAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<EGainMode>	

GetILReferenceAsync(Int32, Int32, CancellationToken)

Declaration

```
public Task<double> GetILReferenceAsync(int detector, int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

GetILResolutionAsync(CancellationToken)

Declaration

```
public async Task<EResolution> GetILResolutionAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<EResolution>	

GetLengthModeAsync(CancellationToken)

Declaration

```
public async Task<ELengthMode> GetLengthModeAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<ELengthMode>	

GetOutputChannelAsync(CancellationToken)

Declaration

```
public Task<int> GetOutputChannelAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Int32>	

GetPositionBModeAsync(CancellationToken)

Declaration

```
public async Task<EPositionBMode> GetPositionBModeAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<EPositionBMode>	

GetRLModeAsync(CancellationToken)

Declaration

```
public async Task<ERLMode> GetRLModeAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<ERLMode>	

GetRLReferenceAsync(CancellationToken)

Declaration

```
public async Task<double> GetRLReferenceAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

GetSwitchChannelAsync(Int32, CancellationToken)

Declaration

```
public Task<int> GetSwitchChannelAsync(int switchIndex, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	switchIndex	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Int32>	

GetTraceDiagnosticsAsync(CancellationToken)

Declaration

```
public async Task<TraceDiagnostics> GetTraceDiagnosticsAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<TraceDiagnostics>	

HandleIDNResponse(String[])

Declaration

```
protected override void HandleIDNResponse(string[] idnResponseParts)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String[]	idnResponseParts	

Overrides

SantecDevice.HandleIDNResponse(String[])

InitializeAsync()

Declaration

```
public override async Task InitializeAsync()
```

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Overrides

SantecDevice.InitializeAsync()

Exceptions

TYPE	CONDITION
UnexpectedDeviceResponseException	Thrown when the RLM model response does not correspond to an RLM model.
UnexpectedDeviceResponseException	Thrown when the RLM wavelengths response contains one or more wavelengths that are not valid values.
UnexpectedDeviceResponseException	Thrown when the RLM detectors response is not a valid number.

TYPE	CONDITION
UnexpectedDeviceResponseException	Thrown when the RLM switches response contains one or more switch channel counts that are not valid values.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the RLM.

ReadILAsync(Int32, Int32, CancellationToken)

Declaration

```
public Task<double> ReadILAsync(int detector, int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

ReadMonitorPowerAsync(Int32, CancellationToken)

Declaration

```
public Task<double> ReadMonitorPowerAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

ReadMultipleILsAsync(IEnumerable<Int32>, Int32, CancellationToken)

Declaration

```
public Task<List<double>> ReadMultipleILsAsync(IEnumerable<int> detectors, int wavelength, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Collections.Generic.IEnumerable<System.Int32>	detectors	
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Collections.Generic.List<System.Double>>	

ReadMultiplePowersAsync(IEnumerable<Int32>, Int32, CancellationToken)

Declaration

```
public Task<List<double>> ReadMultiplePowersAsync(IEnumerable<int> detectors, int wavelength,
CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Collections.Generic.IEnumerable<System.Int32>	detectors	
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Collections.Generic.List<System.Double>>	

ReadPowerAsync(Int32, Int32, CancellationToken)

Declaration

```
public Task<double> ReadPowerAsync(int detector, int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Int32	wavelength	

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

ReadRLAsync(Int32, Double, Double, CancellationToken)

Declaration

```
public async Task<RLReading> ReadRLAsync(int wavelength, double lengthReferenceA, double lengthReferenceB,
CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	
System.Double	lengthReferenceA	
System.Double	lengthReferenceB	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<RLReading>	

ReadRLAsync(Int32, Double, CancellationToken)

Declaration

```
public async Task<RLReading> ReadRLAsync(int wavelength, double lengthReference, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	
System.Double	lengthReference	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION

TYPE	DESCRIPTION
System.Threading.Tasks.Task<RLReading>	

ReadRLAsync(Int32, CancellationToken)

Declaration

```
public async Task<RLReading> ReadRLAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<RLReading>	

ReferenceILAsync(Int32, Int32, Boolean, CancellationToken)

Declaration

```
public Task<double> ReferenceILAsync(int detector, int wavelength, bool applyReferenceToAllDetectors,
CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Int32	wavelength	
System.Boolean	applyReferenceToAllDetectors	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

ReferenceILAsync(Int32, Int32, CancellationToken)

Declaration

```
public Task<double> ReferenceILAsync(int detector, int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

ReferenceRLAsync(CancellationToken)

Declaration

```
public async Task<double> ReferenceRLAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

RefreshAsync()

Declaration

```
public override async Task RefreshAsync()
```

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Overrides

PhysicalDevice.RefreshAsync()

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized RLM.

TYPE	CONDITION
UnexpectedDeviceResponseException	Thrown when the RLM wavelengths response contains one or more wavelengths that are not valid values.
UnexpectedDeviceResponseException	Thrown when the RLM detectors response is not a valid number.
UnexpectedDeviceResponseException	Thrown when the RLM switches response contains one or more switch channel counts that are not valid values.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the RLM.

SaveILReferencesAsync(CancellationToken)

Declaration

```
public async Task SaveILReferencesAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetDUTILAsync(Int32, Double, CancellationToken)

Declaration

```
public async Task SetDUTILAsync(int wavelength, double il, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	
System.Double	il	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetGainModeAsync(EGainMode, CancellationToken)

Declaration

```
public async Task SetGainModeAsync(EGainMode gainMode, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
EGainMode	gainMode	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetILReferenceAsync(Int32, Int32, Double, CancellationToken)

Declaration

```
public Task SetILReferenceAsync(int detector, int wavelength, double reference, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Int32	wavelength	
System.Double	reference	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetILResolutionAsync(EResolution, CancellationToken)

Declaration

```
public async Task SetILResolutionAsync(EResolution resolution, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
EResolution	resolution	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetLengthModeAsync(ELengthMode, CancellationToken)

Declaration

```
public async Task SetLengthModeAsync(ELengthMode lengthMode, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
ELengthMode	lengthMode	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetPositionBModeAsync(EPositionBMode, CancellationToken)

Declaration

```
public async Task SetPositionBModeAsync(EPositionBMode positionBMode, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
EPositionBMode	positionBMode	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetRLModeAsync(ERLMode, CancellationToken)

Declaration

```
public async Task SetRLModeAsync(ERLMode rlMode, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
ERLMode	rlMode	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetRLReferenceAsync(Double, CancellationToken)

Declaration

```
public async Task SetRLReferenceAsync(double reference, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Double	reference	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

TurnOffLaserAsync(CancellationToken)

Declaration

```
public Task TurnOffLaserAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

TurnOnLaserAsync(Int32, CancellationToken)

Declaration

```
public Task TurnOnLaserAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Uninitialize()

Declaration

```
public override void Uninitialize()
```

Overrides

PhysicalDevice.Uninitialize()

Explicit Interface Implementations

IDetectorDevice.Detectors

Declaration

```
IReadOnlyList<IModule> IDetectorDevice.Detectors { get; }
```

Returns

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<IModule>	

Implements

IRLM
ISantecDevice
IDiscretePowerMeter
ILaserSourceWithMonitor
ILaserSource
IWavelengthDevice
ISwitchController
IFiberDevice
IPowerMeterWithReferencePersistence
IPowerMeter
IInsertionLossMeter
IDetectorDevice
IDevice<TModelType>

IDevice

System.IDisposable

Extension Methods

RLMExtensions.IsMultimodeRLM100(IRLM)

Class RLReading

Represents the results from an RLM return loss reading.

> **NOTE**
>
> This class should not be used directly. It should only be used when returned by an RLM return loss measurement.

Inheritance

System.Object
RLReading

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.ReturnLoss

Assembly: Santec.Hardware.dll

Syntax

```
public class RLReading
```

Constructors

RLReading(ReadingValue, ReadingValue, ReadingValue, Double)

Initialize a new return loss result.

Declaration

```
public RLReading(ReadingValue rlA, ReadingValue rlB, ReadingValue rlTotal, double length)
```

Parameters

TYPE	NAME	DESCRIPTION
ReadingValue	rlA	The return loss reading of connector A.
ReadingValue	rlB	The return loss reading of connector B.
ReadingValue	rlTotal	The total return loss reading of the fiber.
System.Double	length	The measured length of the fiber.

Properties

Length

The length of the fiber.

Declaration

```
public double Length { get; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The length of the fiber.

Remarks

> **NOTE**
>
> This property should be set to `double.NaN` if the fiber length could not properly be measured by the RLM.

RLa

The return loss reading of connector A.

Declaration

```
public ReadingValue RLa { get; }
```

Property Value

TYPE	DESCRIPTION
ReadingValue	The return loss reading of connector A.

Remarks

> **NOTE**
>
> The Value of this property should be set to `double.NaN` if the RLa could not properly be measured by the RLM.

RLb

The return loss reading of connector B.

Declaration

```
public ReadingValue RLb { get; }
```

Property Value

TYPE	DESCRIPTION
ReadingValue	The return loss reading of connector B.

Remarks

NOTE

The Value of this property should be set to `double.NaN` if the RLb could not properly be measured by the RLM.

RLtotal

The total return loss reading of the fiber.

Declaration

```
public ReadingValue RLtotal { get; }
```

Property Value

TYPE	DESCRIPTION
ReadingValue	The total return loss reading of the fiber.

Remarks

NOTE

The Value of this property should be set to `double.NaN` if the RLtotal could not properly be measured by the RLM.

Class SimulatedRLM

Provides a simulated RLM with configurable reading values.

> **NOTE**
>
> This class is intended for testing/development purposes. It operates much quicker than an actual RLM. It can only be created using a SimulatedDeviceFactory. It should only be used when returned as part of a list of devices by a DetectHardwareAsync(EConnectionTypesToDetect) or DetectHardwareAsync<T>(EConnectionTypesToDetect) call to a SimulatedHardwareDetectionService. Each reading value contains a default value range but can be configured via the appropriate configuration method.

Inheritance

System.Object
SimulatedSantecDevice
SimulatedRLM

Implements

ISimulatedDevice
IRLM
ISantecDevice
IDiscretePowerMeter
ISwitchController
IFiberDevice
IPowerMeterWithReferencePersistence
IDevice<ERLMModel>
IPowerMeter
IInsertionLossMeter
IDetectorDevice
ILaserSourceWithMonitor
ILaserSource
IWavelengthDevice
IDevice
System.IDisposable

Inherited Members

SimulatedSantecDevice.creatingDevice
SimulatedSantecDevice.Dispose()
SimulatedSantecDevice.SerialNumber
SimulatedSantecDevice.FirmwareVersion
SimulatedSantecDevice.Address
SimulatedSantecDevice.IsInitialized
SimulatedSantecDevice.QueryResponses
SimulatedSantecDevice.ConfigureQueryResponses(IEnumerable<QueryResponsePair>)
SimulatedSantecDevice.GetTimeout()
SimulatedSantecDevice.SetTimeout(ETimeout)
SimulatedSantecDevice.GetInteractionModeAsync(CancellationToken)
SimulatedSantecDevice.GetIPAddressAsync(CancellationToken)
SimulatedSantecDevice.SetIPAddressAsync(IPAddress, CancellationToken)
SimulatedSantecDevice.EnableDHCPAsync(CancellationToken)
SimulatedSantecDevice.GetGatewayAsync(CancellationToken)
SimulatedSantecDevice.SetGatewayAsync(IPAddress, CancellationToken)
SimulatedSantecDevice.GetSubnetPrefixLengthAsync(CancellationToken)
SimulatedSantecDevice.SetSubnetPrefixLengthAsync(Int32, CancellationToken)

SimulatedSantecDevice.GetHostnameAsync(CancellationToken)
SimulatedSantecDevice.SetHostnameAsync(String, CancellationToken)
SimulatedSantecDevice.SetInteractionModeAsync(EInteractionMode, CancellationToken)
SimulatedSantecDevice.InitializeAsync()
SimulatedSantecDevice.PingAsync()
SimulatedSantecDevice.Uninitialize()
SimulatedSantecDevice.ResetAsync()
SimulatedSantecDevice.RefreshAsync()
SimulatedSantecDevice.QueryAsync(String, CancellationToken)
SimulatedSantecDevice.WriteAsync(String, CancellationToken)
SimulatedSantecDevice.ReadAsync(CancellationToken)
System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.ReturnLoss

Assembly: Santec.Hardware.dll

Syntax

```
public class SimulatedRLM : SimulatedSantecDevice, ISimulatedDevice, IRLM, ISantecDevice,
IDiscretePowerMeter, ISwitchController, IFiberDevice, IPowerMeterWithReferencePersistence,
IDevice<ERLMModel>, IPowerMeter, IInsertionLossMeter, IDetectorDevice, ILaserSourceWithMonitor,
ILaserSource, IWavelengthDevice, IDevice, IDisposable
```

Properties

Detectors

Declaration

```
public IReadOnlyList<DeviceDetector> Detectors { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<DeviceDetector>	

Fiber

Declaration

```
public FiberInformation Fiber { get; }
```

Property Value

TYPE	DESCRIPTION
FiberInformation	

ILMax

Declaration

```
public double ILMax { get; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The maximum IL reading value. The default value is `0.15`.

ILMin

Declaration

```
public double ILMin { get; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The minimum IL reading value. The default value is `0`.

ILReferenceMax

Declaration

```
public double ILReferenceMax { get; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The maximum IL reference value. The default value is `5`.

ILReferenceMin

Declaration

```
public double ILReferenceMin { get; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The minimum IL reference value. The default value is `2`.

LengthMax

Get the maximum length value returned by an RL reading.

Declaration

```
public double LengthMax { get; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The maximum length reading value. The default value is `3.05`.

Remarks

Use ConfigureLength(Double, Double) to set this value.

LengthMin

Get the minimum length value returned by an RL reading.

Declaration

```
public double LengthMin { get; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The minimum length reading value. The default value is `2.95`.

Remarks

Use ConfigureLength(Double, Double) to set this value.

Model

Declaration

```
public ERLMModel Model { get; }
```

Property Value

TYPE	DESCRIPTION
ERLMModel	

MonitorPowerMax

Declaration

```
public double MonitorPowerMax { get; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The maximum monitor power value. The default value is `-30`.

MonitorPowerMin

Declaration

```
public double MonitorPowerMin { get; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The minimum monitor power value. The default value is `-30`.

Name

Declaration

```
public override string Name { get; }
```

Property Value

TYPE	DESCRIPTION
System.String	

Overrides

SimulatedSantecDevice.Name

NumberOfDetectors

Declaration

```
public int NumberOfDetectors { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

NumberOfOutputs

Declaration

```
public int NumberOfOutputs { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

PowerMax

Declaration

```
public double PowerMax { get; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The maximum power reading value. The default value is `-3`.

PowerMin

Declaration

```
public double PowerMin { get; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The minimum power reading value. The default value is `-3.15`.

RLaMax

Get the maximum RLa value returned by an RL reading.

Declaration

```
public double RLaMax { get; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The maximum RLa reading value. The default value is `75`.

Remarks

Use ConfigureRLa(Double, Double) to set this value.

RLaMin

Get the minimum RLa value returned by an RL reading.

Declaration

```
public double RLaMin { get; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The minimum RLa reading value. The default value is `55`.

Remarks

Use ConfigureRLa(Double, Double) to set this value.

RLbMax

Get the maximum RLb value returned by an RL reading.

Declaration

```
public double RLbMax { get; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The maximum RLb reading value. The default value is `75`.

Remarks

Use ConfigureRLb(Double, Double) to set this value.

RLbMin

Get the minimum RLb value returned by an RL reading.

Declaration

```
public double RLbMin { get; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The minimum RLb reading value. The default value is `55`.

Remarks

Use ConfigureRLb(Double, Double) to set this value.

RLReferenceMax

Get the maximum value returned by an RL length reference.

Declaration

```
public double RLReferenceMax { get; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The maximum RL length reference value. The default value is `3.05`.

Remarks

Use ConfigureRLReference(Double, Double) to set this value.

RLReferenceMin

Get the minimum value returned by an RL length reference.

Declaration

```
public double RLReferenceMin { get; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The minimum RL length reference value. The default value is `2.95`.

Remarks

Use ConfigureRLReference(Double, Double) to set this value.

RLtotalMax

Get the maximum RLtotal value returned by an RL reading.

Declaration

```
public double RLtotalMax { get; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The maximum RLa reading value. The default value is `75`.

Remarks

Use ConfigureRLtotal(Double, Double) to set this value.

RLtotalMin

Get the minimum RLtotal value returned by an RL reading.

Declaration

```
public double RLtotalMin { get; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The minimum RLtotal reading value. The default value is `55`.

Remarks

Use ConfigureRLtotal(Double, Double) to set this value.

SupportedLengthModes

Declaration

```
public IReadOnlyList<ELengthMode> SupportedLengthModes { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<ELengthMode>	

SupportedRLModes

Declaration

```
public IReadOnlyList<ERLMode> SupportedRLModes { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<ERLMode>	

SupportsILResolution

Declaration

```
public bool SupportsILResolution { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	

SupportsMaximumResolutionRL

Declaration

```
public bool SupportsMaximumResolutionRL { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	

SupportsTurningOffLaser

Declaration

```
public bool SupportsTurningOffLaser { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	

Remarks

NOTE

This property will always return `true`.

Switches

Declaration

```
public List<DeviceSwitch> Switches { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.List<DeviceSwitch>	

Wavelengths

Declaration

```
public List<int> Wavelengths { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.List<System.Int32>	

Methods

ChangeMultipleSwitchesChannelsAsync(IEnumerable<DeviceSwitchAndChannelPair>, CancellationToken)

Declaration

```
public async Task ChangeMultipleSwitchesChannelsAsync(IEnumerable<DeviceSwitchAndChannelPair>
deviceSwitchesAndChannels, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Collections.Generic.IEnumerable<DeviceSwitchAndChannelPair>	deviceSwitchesAndChannels	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

ChangeOutputChannelAsync(Int32, CancellationToken)

Declaration

```
public async Task ChangeOutputChannelAsync(int output, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	output	

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

ChangeSwitchChannelAsync(Int32, Int32, CancellationToken)

Declaration

```
public async Task ChangeSwitchChannelAsync(int switchIndex, int channel, CancellationToken cancellationToken
= default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	switchIndex	
System.Int32	channel	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

ConfigureIL(Double, Double)

Declaration

```
public void ConfigureIL(double ilMin, double ilMax)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Double	ilMin	
System.Double	ilMax	

ConfigureILReference(Double, Double)

Declaration

```
public void ConfigureILReference(double ilReferenceMin, double ilReferenceMax)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Double	ilReferenceMin	
System.Double	ilReferenceMax	

ConfigureLength(Double, Double)

Configure the value range for the length returned by ReadRLAsync(Int32, CancellationToken).

Declaration

```
public void ConfigureLength(double lengthMin, double lengthMax)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Double	lengthMin	The minimum length value.
System.Double	lengthMax	The maximum length value.

Remarks

The value limits can be `double.NaN`. If either limit is `double.NaN`, ReadRLAsync(Int32, CancellationToken) will return `double.NaN` for the length reading.

ConfigureMonitorPower(Double, Double)

Declaration

```
public void ConfigureMonitorPower(double monitorPowerMin, double monitorPowerMax)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Double	monitorPowerMin	
System.Double	monitorPowerMax	

ConfigurePower(Double, Double)

Declaration

```
public void ConfigurePower(double powerMin, double powerMax)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Double	powerMin	
System.Double	powerMax	

ConfigureRLa(Double, Double)

Configure the value range for the RLa returned by ReadRLAsync(Int32, CancellationToken).

Declaration

```
public void ConfigureRLa(double rlAMin, double rlAMax)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Double	rlAMin	The minimum RLa value.
System.Double	rlAMax	The maximum RLa value.

Remarks

The value limits can be `double.NaN`. If either limit is `double.NaN`, ReadRLAsync(Int32, CancellationToken) will return `double.NaN` for the RLa reading.

ConfigureRLb(Double, Double)

Configure the value range for the RLb returned by ReadRLAsync(Int32, CancellationToken).

Declaration

```
public void ConfigureRLb(double rlBMin, double rlBMax)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Double	rlBMin	The minimum RLb value.
System.Double	rlBMax	The maximum RLb value.

Remarks

The value limits can be `double.NaN`. If either limit is `double.NaN`, ReadRLAsync(Int32, CancellationToken) will return `double.NaN` for the RLb reading.

ConfigureRLReference(Double, Double)

Configure the value range for the RL length reference returned by ReferenceRLAsync(CancellationToken).

Declaration

```
public void ConfigureRLReference(double rlReferenceMin, double rlReferenceMax)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Double	rlReferenceMin	The minimum length value.
System.Double	rlReferenceMax	The maximum length value.

Remarks

The value limits can be `double.NaN`. If either limit is `double.NaN`, ReferenceRLAsync(CancellationToken) will return `double.NaN`.

ConfigureRLtotal(Double, Double)

Configure the value range for the RLtotal returned by ReadRLAsync(Int32, CancellationToken).

Declaration

```
public void ConfigureRLtotal(double rlTotalMin, double rlTotalMax)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Double	rlTotalMin	The minimum RLtotal value.
System.Double	rlTotalMax	The maximum RLtotal value.

Remarks

The value limits can be `double.NaN`. If either limit is `double.NaN`, ReadRLAsync(Int32, CancellationToken) will return `double.NaN` for the RLtotal reading.

GetActiveLaserAsync(CancellationToken)

Declaration

```
public async Task<int> GetActiveLaserAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Int32>	

GetDUTILAsync(Int32, CancellationToken)

Declaration

```
public async Task<double> GetDUTILAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

GetFactoryPowerAsync(Int32, Int32, CancellationToken)

Declaration

```
public async Task<FactoryPowerReading> GetFactoryPowerAsync(int channel, int wavelength, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	channel	
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<FactoryPowerReading>	

GetGainModeAsync(CancellationToken)

Declaration

```
public async Task<EGainMode> GetGainModeAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<EGainMode>	

GetILReferenceAsync(Int32, Int32, CancellationToken)

Declaration

```
public async Task<double> GetILReferenceAsync(int detector, int wavelength, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

GetILResolutionAsync(CancellationToken)

Declaration

```
public async Task<EResolution> GetILResolutionAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<EResolution>	

GetLengthModeAsync(CancellationToken)

Declaration

```
public async Task<ELengthMode> GetLengthModeAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<ELengthMode>	

GetOutputChannelAsync(CancellationToken)

Declaration

```
public async Task<int> GetOutputChannelAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Int32>	

GetPositionBModeAsync(CancellationToken)

Declaration

```
public async Task<EPositionBMode> GetPositionBModeAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<EPositionBMode>	

GetRLModeAsync(CancellationToken)

Declaration

```
public async Task<ERLMode> GetRLModeAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<ERLMode>	

GetRLReferenceAsync(CancellationToken)

Declaration

```
public async Task<double> GetRLReferenceAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

GetSwitchChannelAsync(Int32, CancellationToken)

Declaration

```
public async Task<int> GetSwitchChannelAsync(int switchIndex, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	switchIndex	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Int32>	

GetTraceDiagnosticsAsync(CancellationToken)

Declaration

```
public Task<TraceDiagnostics> GetTraceDiagnosticsAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<TraceDiagnostics>	

Remarks

This is currently not implemented.

ReadILAsync(Int32, Int32, CancellationToken)

Declaration

```
public async Task<double> ReadILAsync(int detector, int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

ReadMonitorPowerAsync(Int32, CancellationToken)

Declaration

```
public async Task<double> ReadMonitorPowerAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

ReadMultipleILsAsync(IEnumerable<Int32>, Int32, CancellationToken)

Declaration

```
public async Task<List<double>> ReadMultipleILsAsync(IEnumerable<int> detectors, int wavelength,
CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Collections.Generic.IEnumerable<System.Int32>	detectors	
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Collections.Generic.List<System.Double>>	

ReadMultiplePowersAsync(IEnumerable<Int32>, Int32, CancellationToken)

Declaration

```
public async Task<List<double>> ReadMultiplePowersAsync(IEnumerable<int> detectors, int wavelength,
CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Collections.Generic.IEnumerable<System.Int32>	detectors	
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Collections.Generic.List<System.Double>>	

ReadPowerAsync(Int32, Int32, CancellationToken)

Declaration

```
public async Task<double> ReadPowerAsync(int detector, int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

ReadRLAsync(Int32, Double, Double, CancellationToken)

Declaration

```
public async Task<RLReading> ReadRLAsync(int wavelength, double lengthReferenceA, double lengthReferenceB,
CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	
System.Double	lengthReferenceA	
System.Double	lengthReferenceB	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<RLReading>	

ReadRLAsync(Int32, Double, CancellationToken)

Declaration

```
public async Task<RLReading> ReadRLAsync(int wavelength, double lengthReference, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	
System.Double	lengthReference	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<RLReading>	

ReadRLAsync(Int32, CancellationToken)

Declaration

```
public async Task<RLReading> ReadRLAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<RLReading>	

ReferenceILAsync(Int32, Int32, Boolean, CancellationToken)

Declaration

```
public async Task<double> ReferenceILAsync(int detector, int wavelength, bool applyReferenceToAllDetectors,
CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Int32	wavelength	
System.Boolean	applyReferenceToAllDetectors	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

ReferenceILAsync(Int32, Int32, CancellationToken)

Declaration

```
public async Task<double> ReferenceILAsync(int detector, int wavelength, CancellationToken cancellationToken
= default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

ReferenceRLAsync(CancellationToken)

Declaration

```
public async Task<double> ReferenceRLAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	

SaveILReferencesAsync(CancellationToken)

Declaration

```
public async Task SaveILReferencesAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetDUTILAsync(Int32, Double, CancellationToken)

Declaration

```
public async Task SetDUTILAsync(int wavelength, double il, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	
System.Double	il	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetGainModeAsync(EGainMode, CancellationToken)

Declaration

```
public async Task SetGainModeAsync(EGainMode gainMode, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
EGainMode	gainMode	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetILReferenceAsync(Int32, Int32, Double, CancellationToken)

Declaration

```
public async Task SetILReferenceAsync(int detector, int wavelength, double reference, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	detector	
System.Int32	wavelength	
System.Double	reference	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetILResolutionAsync(EResolution, CancellationToken)

Declaration

```
public async Task SetILResolutionAsync(EResolution resolution, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
EResolution	resolution	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetLengthModeAsync(ELengthMode, CancellationToken)

Declaration

```
public async Task SetLengthModeAsync(ELengthMode lengthMode, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
ELengthMode	lengthMode	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetPositionBModeAsync(EPositionBMode, CancellationToken)

Declaration

```
public async Task SetPositionBModeAsync(EPositionBMode positionBMode, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
EPositionBMode	positionBMode	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetRLModeAsync(ERLMode, CancellationToken)

Declaration

```
public async Task SetRLModeAsync(ERLMode rlMode, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
ERLMode	rlMode	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

SetRLReferenceAsync(Double, CancellationToken)

Declaration

```
public async Task SetRLReferenceAsync(double reference, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Double	reference	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

TurnOffLaserAsync(CancellationToken)

Declaration

```
public async Task TurnOffLaserAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

TurnOnLaserAsync(Int32, CancellationToken)

Declaration

```
public async Task TurnOnLaserAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Explicit Interface Implementations

IDetectorDevice.Detectors

Declaration

```
IReadOnlyList<IModule> IDetectorDevice.Detectors { get; }
```

Returns

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<IModule>	

IPowerMeterWithReferencePersistence.RequiresILReferenceSave

Declaration

```
bool IPowerMeterWithReferencePersistence.RequiresILReferenceSave { get; }
```

Returns

TYPE	DESCRIPTION
System.Boolean	

Implements

ISimulatedDevice
IRLM
ISantecDevice
IDiscretePowerMeter
ISwitchController
IFiberDevice
IPowerMeterWithReferencePersistence
IDevice<TModelType>
IPowerMeter
IInsertionLossMeter
IDetectorDevice
ILaserSourceWithMonitor
ILaserSource
IWavelengthDevice
IDevice
System.IDisposable

Extension Methods

RLMExtensions.IsMultimodeRLM100(IRLM)

Namespace Santec.Hardware.Devices.ReturnLoss.Extensions

Classes

RLMExtensions

Provides extension methods for the IRLM interface.

Class RLMExtensions

Provides extension methods for the IRLM interface.

Inheritance

System.Object
RLMExtensions

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.ReturnLoss.Extensions

Assembly: Santec.Hardware.dll

Syntax

```
public static class RLMExtensions
```

Methods

IsMultimodeRLM100(IRLM)

Determines whether the specified RLM instance is a multimode RLM-100.

Declaration

```
public static bool IsMultimodeRLM100(this IRLM rlm)
```

Parameters

TYPE	NAME	DESCRIPTION
IRLM	rlm	The RLM instance to evaluate. Must not be null.

Returns

TYPE	DESCRIPTION
System.Boolean	true if the RLM instance has a fiber type of Multimode and a model of RLM100; otherwise, false.

Namespace Santec.Hardware.Devices.ReturnLoss.Trace

Classes

TraceDiagnosticInfo

Represents RLM diagnostic trace information from an RL measurement.

> **NOTE**
>
> This class should not be used directly. It should only be used when returned as part of a TraceDiagnostics by an RLM GetTraceDiagnosticsAsync(CancellationToken) call after an RL measurement has been performed.

TraceDiagnostics

Represents RLM diagnostic trace information and trace data from an RL measurement.

> **NOTE**
>
> This class should not be used directly. It should only be used when returned by an RLM GetTraceDiagnosticsAsync(CancellationToken) call after an RL measurement has been performed.

Enums

ERLMType

Specifies the type of an RLM.

Enum ERLMType

Specifies the type of an RLM.

Namespace: Santec.Hardware.Devices.ReturnLoss.Trace

Assembly: Santec.Hardware.dll

Syntax

```
public enum ERLMType
```

Fields

NAME	DESCRIPTION
Multimode	Multimode RLM.
SingleMode	Single mode RLM.

Class TraceDiagnosticInfo

Represents RLM diagnostic trace information from an RL measurement.

> **NOTE**
>
> This class should not be used directly. It should only be used when returned as part of a TraceDiagnostics by an RLM GetTraceDiagnosticsAsync(CancellationToken) call after an RL measurement has been performed.

Inheritance

System.Object
TraceDiagnosticInfo

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.ReturnLoss.Trace

Assembly: Santec.Hardware.dll

Syntax

```
public class TraceDiagnosticInfo
```

Properties

AcquisitionPostDelay

Get/Set the acquisition postDelay.

Declaration

```
[JsonPropertyName("acqPostDelay")]
public int AcquisitionPostDelay { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	The acquisition postDelay.

AcquisitionPreDelay

Get/Set the acquisition preDelay.

Declaration

```
[JsonPropertyName("acqPreDelay")]
public int AcquisitionPreDelay { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	The acquisition preDelay.

AcquisitionSample

Get/Set the acquisition sample.

Declaration

```
[JsonPropertyName("acqSample")]
public int AcquisitionSample { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	The acquisition sample.

APDBias

Get/Set the APD bias.

Declaration

```
public int APDBias { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	The APD bias.

BaseIOR

Get/Set the baseIOR.

Declaration

```
[JsonPropertyName("baseIOR")]
public double BaseIOR { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The baseIOR.

DeltaD

Get/Set the deltaD of the measurement.

Declaration

```
[JsonPropertyName("deltaD")]
public double DeltaD { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The deltaD.

DeltaT

Get/Set the deltaT of the measurement.

Declaration

```
[JsonPropertyName("deltaT")]
public double DeltaT { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The deltaT.

Dither

Get/Set the dither value.

Declaration

```
[JsonPropertyName("ditherVal")]
public int Dither { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	The dither value.

DitherMode

Get/Set the dither mode.

Declaration

```
public int DitherMode { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	The dither mode.

DUTil

Get/Set the DUT IL used in the measurement.

Declaration

```
public double DUTil { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The DUT IL.

EndMTJLength

Get/Set the length of the end MTJ used in the measurement.

Declaration

```
public double EndMTJLength { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The length of the end MTJ.

EndOfFiberBottomIndex

Get/Set the index of the bottom of the end of the fiber of the measurement .

Declaration

```
[JsonPropertyName("endOfFiberBottom")]
public int EndOfFiberBottomIndex { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	The index of the bottom of the end of the fiber.

EndOfFiberPeakHeight

Get/Set the peak height of the end of the fiber for the measurement.

Declaration

```
[JsonPropertyName("eofPeakHeight")]
public double EndOfFiberPeakHeight { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The peak height of the end of the fiber.

EndOfFiberPeakIndex

Get/Set the index of the end of fiber peak for the measurement.

Declaration

```
[JsonPropertyName("eofPeakPoint")]
[JsonConverter(typeof(JSONIndexConverter))]
public int EndOfFiberPeakIndex { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	The index of the end of fiber peak.

EndOfFiberTopIndex

Get/Set the index of the top of the end of the fiber of the measurement.

Declaration

```
[JsonPropertyName("endOfFiberTop")]
public int EndOfFiberTopIndex { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	The index of the top of the end of the fiber.

InjectionLevel

Get/Set the injection level of the measurement.

Declaration

```
public double InjectionLevel { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The injection level.

InjectionLevelEndIndex

Get/Set the index of the end of the injection level of the measurement.

Declaration

```
[JsonPropertyName("injectionLevelEnd")]
public int InjectionLevelEndIndex { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	The index of the end of the injection level.

InjectionLevelStartIndex

Get/Set the index of the start of the injection level of the measurement.

Declaration

```
[JsonPropertyName("injectionLevelStart")]
public int InjectionLevelStartIndex { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	The index of the start of the injection level.

InternalFiberLength

Get/Set the length of the internal fiber.

Declaration

```
[JsonPropertyName("internalLength")]
public double InternalFiberLength { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The length of the internal fiber.

LaserPulseWidth

Get/Set the laser pulse width used for the measurement.

Declaration

```
[JsonPropertyName("pulseWidth")]
public double LaserPulseWidth { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The laser pulse width in ns.

MeasuredFiberLength

Get/Set the measured length of the fiber.

Declaration

```
[JsonPropertyName("measuredLength")]
public double MeasuredFiberLength { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The measured fiber length.

MTJil

Get/Set the MTJ IL used in the measurement.

Declaration

```
[JsonPropertyName("ilMTJ")]
public double MTJil { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The MTJ IL.

NumberOfIterations

Get/Set the number of iterations used for the measurement.

Declaration

```
[JsonPropertyName("noIteration")]
public int NumberOfIterations { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	The number of iterations.

NumberOfPhases

Get/Set the number of phases used for the measurement.

Declaration

```
[JsonPropertyName("noPhase")]
public int NumberOfPhases { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	The number of phases.

OffPostDelay

Get/Set the offPostDelay.

Declaration

```
[JsonPropertyName("offPostDelay")]
public int OffPostDelay { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	The offPostDelay.

OffPreDelay

Get/Set the offPreDelay.

Declaration

```
[JsonPropertyName("offPreDelay")]
public int OffPreDelay { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	The offPreDelay.

OffSample

Get/Set the offSample.

Declaration

```
[JsonPropertyName("offSample")]
public int OffSample { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	The offSample.

PeakDetectionConfiguration

Get/Set the peak detection configuration used for the measurement.

Declaration

```
[JsonPropertyName("detectPeakConfig")]
public int PeakDetectionConfiguration { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	The peak detection configuration.

ReferencedEndOfFiberIndex

Get/Set the index of the referenced end of the fiber of the measurement.

Declaration

```
[JsonPropertyName("referencedEOF")]
public int ReferencedEndOfFiberIndex { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	The index of the referenced end of the fiber.

RefPeakHeightStart

Get/Set the refPeakHeightStart.

Declaration

```
[JsonPropertyName("refPeakHeightStart")]
public double RefPeakHeightStart { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The refPeakHeightStart.

RLa

Get/Set the RLa for the measurement.

Declaration

```
[JsonPropertyName("rlAValue")]
public double RLa { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The RLa value.

RLaH

Get/Set the RLa H value for the measurement.

Declaration

```
[JsonPropertyName("rlAHValue")]
public double RLaH { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The RLa H value.

RLaIndex

Get/Set the index of the RLa point for the measurement.

Declaration

```
[JsonPropertyName("rlAPoint")]
public int RLaIndex { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	The index of the RLa point.

RLaPeakHeight

Get/Set the RLa peak height for the measurement.

Declaration

```
public double RLaPeakHeight { get; set; }
```

Property Value

TYPE	DESCRIPTION

TYPE	DESCRIPTION
System.Double	The RLa peak height.

RLaReferenceIndex

Get/Set the index of the RLa reference point.

Declaration

```
[JsonPropertyName("rlARefPoint")]
public int RLaReferenceIndex { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	The index of the RLa reference point.

RLb

Get/Set the RLb for the measurement.

Declaration

```
[JsonPropertyName("rlBValue")]
public double RLb { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The RLb value.

RLbH

Get/Set the RLb H value for the measurement.

Declaration

```
[JsonPropertyName("rlBHValue")]
public double RLbH { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The RLb H value.

RLbIndex

Get/Set the index of the RLb point for the measurement.

Declaration

```
[JsonPropertyName("rlBPoint")]
public int RLbIndex { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	The index of the RLb point.

RLbPeakHeight

Get/Set the RLb peak height for the measurement.

Declaration

```
public double RLbPeakHeight { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The RLb peak height.

RLbReferenceIndex

Get/Set the index of the RLb reference point.

Declaration

```
[JsonPropertyName("rlBRefPoint")]
public int RLbReferenceIndex { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	The index of the RLb reference point.

RLMType

Get/Set the RLM type.

Declaration

```
[JsonPropertyName("moduleType")]
[JsonConverter(typeof(JSONRLMTypeConverter))]
public ERLMType RLMType { get; set; }
```

Property Value

TYPE	DESCRIPTION
ERLMType	The RLM type.

RLtotal

Get/Set the RLtotal for the measurement.

Declaration

```
[JsonPropertyName("rlTotalValue")]
public double RLtotal { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The RLtotal value.

RLtotalEndIndex

Get/Set the index of the end of the RLtotal for the measurement.

Declaration

```
[JsonPropertyName("rlTotalEnd")]
public int RLtotalEndIndex { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	The index of the end of the RLtotal.

RLtotalStartIndex

Get/Set the index of the start of the RLtotal for the measurement.

Declaration

```
[JsonPropertyName("rlTotalStart")]
public int RLtotalStartIndex { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	The index of the start of the RLtotal.

StartMTJLength

Get/Set the length of the start MTJ used in the measurement.

Declaration

```
public double StartMTJLength { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The length of the start MTJ.

StartOfFiberIndex

Get/Set the index of the start of the fiber of the measurement.

Declaration

```
[JsonPropertyName("startOfFiber")]
public int StartOfFiberIndex { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	The index of the start of the fiber.

StartPhase

Get/Set the start phase of the measurement.

Declaration

```
public int StartPhase { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	The start phase.

ZeroPositionIndex

Get/Set the index of the zero position.

Declaration

```
[JsonConverter(typeof(JSONIndexConverter))]
public int ZeroPositionIndex { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	The index of the zero position.

Class TraceDiagnostics

Represents RLM diagnostic trace information and trace data from an RL measurement.

> **NOTE**
>
> This class should not be used directly. It should only be used when returned by an RLM GetTraceDiagnosticsAsync(CancellationToken) call after an RL measurement has been performed.

Inheritance

System.Object
TraceDiagnostics

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.ReturnLoss.Trace

Assembly: Santec.Hardware.dll

Syntax

```
public class TraceDiagnostics
```

Properties

DiagnosticInfo

Get/Set the diagnostic info for the measurement.

Declaration

```
[JsonPropertyName("debugInfo")]
public TraceDiagnosticInfo DiagnosticInfo { get; set; }
```

Property Value

TYPE	DESCRIPTION
TraceDiagnosticInfo	The diagnostic information of the trace.

TraceData

Get/Set the trace data for the measurement.

Declaration

```
public List<double> TraceData { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.List<System.Double>	The trace data.

Methods

DeepCopy()

Create a deep copy of the trace diagnostics.

Declaration

```
public TraceDiagnostics DeepCopy()
```

Returns

TYPE	DESCRIPTION
TraceDiagnostics	A deep copy of the trace diagnostics.

Namespace Santec.Hardware.Devices.Source

Classes

FactoryPowerReading

Represents the results from a factory power reading.

> **NOTE**
>
> This class should not be used directly. It should only be used when returned by a laser source's factory power reading.

Interfaces

IDiscreteSource

Defines the properties and operations of a basic source with a single discrete wavelength that is part of a device.

ILaserSource

Defines the properties of a laser source and the operations that can be performed with a connection to the device.

ILaserSourceWithMonitor

Defines the properties of a laser source with a power monitor and defines the operations that can be performed with a connection to the device.

ISource

Defines the properties and operations of a basic source that is part of a device.

Enums

ESourceState

Specifies the source state of a source

Enum ESourceState

Specifies the source state of a source

Namespace: Santec.Hardware.Devices.Source

Assembly: Santec.Hardware.dll

Syntax

```
public enum ESourceState
```

Fields

NAME	DESCRIPTION
Off	Source is off.
On	Source is on.

Class FactoryPowerReading

Represents the results from a factory power reading.

> **NOTE**
>
> This class should not be used directly. It should only be used when returned by a laser source's factory power reading.

Inheritance

System.Object
FactoryPowerReading

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.Source

Assembly: Santec.Hardware.dll

Syntax

```
public class FactoryPowerReading
```

Constructors

FactoryPowerReading(Double, Double)

Initializes a new factory power reading.

Declaration

```
public FactoryPowerReading(double detectorPower, double monitorPower)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Double	detectorPower	The detector power reading.
System.Double	monitorPower	The monitor power reading.

Properties

DetectorPower

The detector power reading.

Declaration

```
public double DetectorPower { get; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The detector power reading.

MonitorPower

The monitor power reading.

Declaration

```
public double MonitorPower { get; }
```

Property Value

TYPE	DESCRIPTION
System.Double	The monitor power reading.

Interface IDiscreteSource

Defines the properties and operations of a basic source with a single discrete wavelength that is part of a device.

Inherited Members

ISource.GetSourceStateAsync(CancellationToken)
ISource.TurnOnSourceAsync(CancellationToken)
ISource.TurnOffSourceAsync(CancellationToken)
IFiberDevice.Fiber

Namespace: Santec.Hardware.Devices.Source

Assembly: Santec.Hardware.dll

Syntax

```
public interface IDiscreteSource : ISource, IFiberDevice
```

Properties

Wavelength

Get the wavelength of the source.

> **NOTE**
>
> The device must be initialized before this property contains the proper values.

Declaration

```
int Wavelength { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	The wavelength of the source.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the property is accessed before the device has been initialized.

Interface ILaserSource

Defines the properties of a laser source and the operations that can be performed with a connection to the device.

Inherited Members

IWavelengthDevice.Wavelengths
IDevice.Name
IDevice.SerialNumber
IDevice.FirmwareVersion
IDevice.Address
IDevice.IsInitialized
IDevice.InitializeAsync()
IDevice.Uninitialize()
IDevice.GetTimeout()
IDevice.SetTimeout(ETimeout)
IDevice.ResetAsync()
IDevice.RefreshAsync()
IDevice.PingAsync()
IDevice.QueryAsync(String, CancellationToken)
IDevice.WriteAsync(String, CancellationToken)
IDevice.ReadAsync(CancellationToken)
System.IDisposable.Dispose()

Namespace: Santec.Hardware.Devices.Source

Assembly: Santec.Hardware.dll

Syntax

```
public interface ILaserSource : IWavelengthDevice, IDevice, IDisposable
```

Properties

NumberOfOutputs

Get the number of laser outputs.

> **NOTE**
>
> The device must be initialized before this property contains the proper value.

Declaration

```
int NumberOfOutputs { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	The number of device outputs.

Exceptions

TYPE	CONDITION

TYPE	CONDITION
System.InvalidOperationException	Thrown when the property is accessed before the device has been initialized.

SupportsTurningOffLaser

Get whether the laser source supports turning off the laser.

> **NOTE**
>
> The device must be initialized before this property contains the proper value.

Declaration

```
bool SupportsTurningOffLaser { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	`true` if the laser source supports turning off the laser; otherwise, `false`.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the property is accessed before the device has been initialized.

Methods

ChangeOutputChannelAsync(Int32, CancellationToken)

Asynchronously change the output channel of the laser source.

Declaration

```
Task ChangeOutputChannelAsync(int output, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	output	The output channel to switch to.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	The task representing the operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the specified output channel is out of range for the laser source.
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device switch output channel response does not match the expected response.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the RLM.

GetActiveLaserAsync(CancellationToken)

Asynchronously get the active laser.

Declaration

```
Task<int> GetActiveLaserAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Int32>	The task representing the asynchronous operation.

Remarks

If no laser is active, the result of the returned task will be `0`.

> **NOTE**

This operation will not block.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device response does not correspond to a supported wavelength or to no active laser.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

GetOutputChannelAsync(CancellationToken)

Asynchronously get the current output channel of the laser source.

Declaration

```
Task<int> GetOutputChannelAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Int32>	The task representing the operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device switch channel response is not a valid switch channel.

TYPE	CONDITION
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

TurnOffLaserAsync(CancellationToken)

Asynchronously turn off the active laser.

Declaration

```
Task TurnOffLaserAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	The task representing the operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
System.InvalidOperationException	Thrown if the laser source does not support turning off lasers.
UnexpectedDeviceResponseException	Thrown when the device laser response does not match the expected response.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

TurnOnLaserAsync(Int32, CancellationToken)

Asynchronously turn on the specified laser.

Declaration

```
Task TurnOnLaserAsync(int wavelength, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	The wavelength of the laser to turn on.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the device does not support the specified wavelength.
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device laser response does not match the expected response.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

Interface ILaserSourceWithMonitor

Defines the properties of a laser source with a power monitor and defines the operations that can be performed with a connection to the device.

Inherited Members

ILaserSource.NumberOfOutputs
ILaserSource.SupportsTurningOffLaser
ILaserSource.GetActiveLaserAsync(CancellationToken)
ILaserSource.TurnOnLaserAsync(Int32, CancellationToken)
ILaserSource.TurnOffLaserAsync(CancellationToken)
ILaserSource.GetOutputChannelAsync(CancellationToken)
ILaserSource.ChangeOutputChannelAsync(Int32, CancellationToken)
IWavelengthDevice.Wavelengths
IDevice.Name
IDevice.SerialNumber
IDevice.FirmwareVersion
IDevice.Address
IDevice.IsInitialized
IDevice.InitializeAsync()
IDevice.Uninitialize()
IDevice.GetTimeout()
IDevice.SetTimeout(ETimeout)
IDevice.ResetAsync()
IDevice.RefreshAsync()
IDevice.PingAsync()
IDevice.QueryAsync(String, CancellationToken)
IDevice.WriteAsync(String, CancellationToken)
IDevice.ReadAsync(CancellationToken)
System.IDisposable.Dispose()

Namespace: Santec.Hardware.Devices.Source

Assembly: Santec.Hardware.dll

Syntax

```
public interface ILaserSourceWithMonitor : ILaserSource, IWavelengthDevice, IDevice, IDisposable
```

Methods

GetFactoryPowerAsync(Int32, Int32, CancellationToken)

Asynchronously get the factory power for the specified output channel and wavelength.

Declaration

```
Task<FactoryPowerReading> GetFactoryPowerAsync(int output, int wavelength, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	output	The output channel to get the factory power for.

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	The wavelength to get the factory power for.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<FactoryPowerReading>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the specified output channel is out of range.
System.ArgumentException	Thrown when the device does not support the specified wavelength.
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device factory power response does not match the expected format.
UnexpectedDeviceResponseException	Thrown when any of the device factory power reading values is not a valid number.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

ReadMonitorPowerAsync(Int32, CancellationToken)

Read the monitor power at the specified wavelength.

Declaration

```
Task<double> ReadMonitorPowerAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	wavelength	The wavelength to read the monitor power at.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Double>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the device does not support the specified wavelength.
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device monitor power response is not a valid number.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

Interface ISource

Defines the properties and operations of a basic source that is part of a device.

Inherited Members

IFiberDevice.Fiber

Namespace: Santec.Hardware.Devices.Source

Assembly: Santec.Hardware.dll

Syntax

```
public interface ISource : IFiberDevice
```

Methods

GetSourceStateAsync(CancellationToken)

Asynchronously get the state of the source.

Declaration

```
Task<ESourceState> GetSourceStateAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<ESourceState>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device response does not correspond to a source state.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

TurnOffSourceAsync(CancellationToken)

Asynchronously turn off the source.

Declaration

```
Task TurnOffSourceAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	The task representing the operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device source response does not match the expected response.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

TurnOnSourceAsync(CancellationToken)

Asynchronously turn on the source.

Declaration

```
Task TurnOnSourceAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION

TYPE	NAME	DESCRIPTION
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device source response does not match the expected response.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

Namespace Santec.Hardware.Devices.Switching

Classes

DeviceSwitch

Represents a switch connected to a ISwitchController.

DeviceSwitchAndChannelPair

Represents a device switch and a channel on the switch.

ExternalDeviceSwitch

Represents an external switch connected to a ISwitchController.

OSX

Provides a connection to an OSX along with properties of the device and operations that can be performed using it.

> **NOTE**
>
> This class cannot not be used directly. It can only be used when returned:
>
> 1. As part of a list of devices by a DetectHardwareAsync(EConnectionTypesToDetect) or DetectHardwareAsync<T>(EConnectionTypesToDetect) call.
> 2. As a device by a TryDetectIPDeviceAsync(IPAddress) or TryDetectIPDeviceAsync<T>(IPAddress) call.

SimulatedExternalDeviceSwitchInformation

Represents the properties of a simulated external device switch.

> **NOTE**
>
> This class is intended for testing/development purposes. It supports the creation of simulated external device switches for simulated RLMs.

SimulatedOSX

Provides a simulated OSX with configurable switch modules.

> **NOTE**
>
> This class is intended for testing/development purposes. It operates much quicker than an actual OSX. It can only be created using a SimulatedDeviceFactory. It should only be used when returned as part of a list of devices by a DetectHardwareAsync(EConnectionTypesToDetect) or DetectHardwareAsync<T>(EConnectionTypesToDetect) call to a SimulatedHardwareDetectionService.

Interfaces

IOSX

Defines the properties of an OSX and the operations that can be performed with a connection to the device.

ISwitchController

Defines the properties of a device that controls one or more switches and the operations that can be performed with a connection to the device.

Enums

EOSXModel

Specifies the model of an OSX.

Class DeviceSwitch

Represents a switch connected to a ISwitchController.

Inheritance

System.Object
DeviceSwitch
ExternalDeviceSwitch

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.Switching

Assembly: Santec.Hardware.dll

Syntax

```
public class DeviceSwitch
```

Properties

Index

Get the switch index.

Declaration

```
public int Index { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	The switch index.

NumberOfChannels

Get the number of the channels the switch has.

Declaration

```
public int NumberOfChannels { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	The number of channels the switch has.

Class DeviceSwitchAndChannelPair

Represents a device switch and a channel on the switch.

Inheritance

System.Object
DeviceSwitchAndChannelPair

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.Switching

Assembly: Santec.Hardware.dll

Syntax

```
public class DeviceSwitchAndChannelPair
```

Constructors

DeviceSwitchAndChannelPair(Int32, Int32)

Initializes a new device switch and channel pair.

Declaration

```
public DeviceSwitchAndChannelPair(int deviceSwitchIndex, int channel)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	deviceSwitchIndex	
System.Int32	channel	

Properties

Channel

Get the switch channel.

Declaration

```
public int Channel { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	The switch channel.

DeviceSwitchIndex

Get the switch index of the switch.

Declaration

```
public int DeviceSwitchIndex { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	The switch index of the switch.

Enum EOSXModel

Specifies the model of an OSX.

Namespace: Santec.Hardware.Devices.Switching

Assembly: Santec.Hardware.dll

Syntax

```
public enum EOSXModel
```

Fields

NAME	DESCRIPTION
OSX100	The OSX-100 model.

Class ExternalDeviceSwitch

Represents an external switch connected to a ISwitchController.

Inheritance

System.Object
DeviceSwitch
ExternalDeviceSwitch

Inherited Members

DeviceSwitch.Index
DeviceSwitch.NumberOfChannels
System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.Switching

Assembly: Santec.Hardware.dll

Syntax

```
public class ExternalDeviceSwitch : DeviceSwitch
```

Properties

Fiber

Get the switch fiber information.

> **NOTE**
>
> The fiber type and core size can be unknown if the parent switch controller does not support retrieving the switch fiber information.

Declaration

```
public FiberInformation Fiber { get; }
```

Property Value

TYPE	DESCRIPTION
FiberInformation	The switch fiber information.

FirmwareVersion

Get the switch firmware version.

Declaration

```
public Version FirmwareVersion { get; }
```

Property Value

TYPE	DESCRIPTION
Version	The switch firmware version.

SerialNumber

Get the switch serial number.

Declaration

```
public string SerialNumber { get; }
```

Property Value

TYPE	DESCRIPTION
System.String	The switch serial number.

Interface IOSX

Defines the properties of an OSX and the operations that can be performed with a connection to the device.

Inherited Members

ISantecDevice.GetIPAddressAsync(CancellationToken)
ISantecDevice.SetIPAddressAsync(IPAddress, CancellationToken)
ISantecDevice.EnableDHCPAsync(CancellationToken)
ISantecDevice.GetGatewayAsync(CancellationToken)
ISantecDevice.SetGatewayAsync(IPAddress, CancellationToken)
ISantecDevice.GetSubnetPrefixLengthAsync(CancellationToken)
ISantecDevice.SetSubnetPrefixLengthAsync(Int32, CancellationToken)
ISantecDevice.GetHostnameAsync(CancellationToken)
ISantecDevice.SetHostnameAsync(String, CancellationToken)
ISantecDevice.GetInteractionModeAsync(CancellationToken)
ISantecDevice.SetInteractionModeAsync(EInteractionMode, CancellationToken)
IModuleController.NumberOfModules
IDevice<EOSXModel>.Model
IDevice.Name
IDevice.SerialNumber
IDevice.FirmwareVersion
IDevice.Address
IDevice.IsInitialized
IDevice.InitializeAsync()
IDevice.Uninitialize()
IDevice.GetTimeout()
IDevice.SetTimeout(ETimeout)
IDevice.ResetAsync()
IDevice.RefreshAsync()
IDevice.PingAsync()
IDevice.QueryAsync(String, CancellationToken)
IDevice.WriteAsync(String, CancellationToken)
IDevice.ReadAsync(CancellationToken)
System.IDisposable.Dispose()

Namespace: Santec.Hardware.Devices.Switching

Assembly: Santec.Hardware.dll

Syntax

```
public interface IOSX : ISantecDevice, IModuleController, IDevice<EOSXModel>, IDevice, IDisposable
```

Properties

Modules

Get the modules of the switch.

> **NOTE**
>
> The OSX must be initialized before this property contains the proper values.

Declaration

```
IReadOnlyList<ISwitchModule> Modules { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<ISwitchModule>	A list of the modules the switch contains.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the property is accessed before the OSX has been initialized.

Methods

ChangeAllModuleChannelsAsync(Int32, CancellationToken)

Asynchronously change the channel of all modules.

Declaration

```
Task ChangeAllModuleChannelsAsync(int channel, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	channel	The channel to switch all modules to.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	The task representing the asynchronous operation.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the specified channel is out of range for any of the OSX modules.
System.InvalidOperationException	Thrown when the method is called on an uninitialized OSX.
UnexpectedDeviceResponseException	Thrown when the OSX change all channels response does not match the expected response.

TYPE	CONDITION
System.TimeoutException	Thrown when a communication failure causes no response to be received from the OSX.

ChangeModuleChannelAsync(Int32, Int32, CancellationToken)

Asynchronously change the channel of a given switch module.

Declaration

```
Task ChangeModuleChannelAsync(int moduleIndex, int channel, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	moduleIndex	The index of the module to change the channel of.
System.Int32	channel	The channel to switch to.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the specified module is out of range for the OSX.
System.ArgumentException	Thrown when the specified channel is out of range for the OSX module.
System.InvalidOperationException	Thrown when the method is called on an uninitialized OSX.

TYPE	CONDITION
UnexpectedDeviceResponseException	Thrown when the OSX module channel response does not match the expected response.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the OSX.

ChangeModuleCommonAsync(Int32, Int32, CancellationToken)

Asynchronously change the common channel of a given switch module.

Declaration

```
Task ChangeModuleCommonAsync(int moduleIndex, int common, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	moduleIndex	The index of the module to change the common channel of.
System.Int32	common	The common channel to switch to.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the specified module is out of range for the OSX.
System.ArgumentException	Thrown when the specified common channel is out of range for the OSX module.

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized OSX.
UnexpectedDeviceResponseException	Thrown when the OSX module common channel response does not match the expected response.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the OSX.

GetModuleChannelAsync(Int32, CancellationToken)

Asynchronously get the current channel of a given switch module.

Declaration

```
Task<int> GetModuleChannelAsync(int moduleIndex, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	moduleIndex	The index of the module to get the current channel for.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Int32>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the specified module is out of range for the OSX.
System.InvalidOperationException	Thrown when the method is called on an uninitialized OSX.

TYPE	CONDITION
UnexpectedDeviceResponseException	Thrown when the OSX module channel response is not a valid module channel.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the OSX.

GetModuleCommonAsync(Int32, CancellationToken)

Asynchronously get the current common channel of a given switch module.

Declaration

```
Task<int> GetModuleCommonAsync(int moduleIndex, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	moduleIndex	The index of the module to get the current common channel for.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Int32>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the specified module is out of range for the OSX.
System.InvalidOperationException	Thrown when the method is called on an uninitialized OSX.
UnexpectedDeviceResponseException	Thrown when the OSX module common channel response is not a valid module channel.

TYPE	CONDITION
System.TimeoutException	Thrown when a communication failure causes no response to be received from the OSX.

Interface ISwitchController

Defines the properties of a device that controls one or more switches and the operations that can be performed with a connection to the device.

Inherited Members

IDevice.Name
IDevice.SerialNumber
IDevice.FirmwareVersion
IDevice.Address
IDevice.IsInitialized
IDevice.InitializeAsync()
IDevice.Uninitialize()
IDevice.GetTimeout()
IDevice.SetTimeout(ETimeout)
IDevice.ResetAsync()
IDevice.RefreshAsync()
IDevice.PingAsync()
IDevice.QueryAsync(String, CancellationToken)
IDevice.WriteAsync(String, CancellationToken)
IDevice.ReadAsync(CancellationToken)
System.IDisposable.Dispose()

Namespace: Santec.Hardware.Devices.Switching

Assembly: Santec.Hardware.dll

Syntax

```
public interface ISwitchController : IDevice, IDisposable
```

Properties

Switches

Get the switches connected to the device.

> **NOTE**
>
> The device must be initialized before this property contains the proper value.

Declaration

```
List<DeviceSwitch> Switches { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.List<DeviceSwitch>	The list of the switches connected to the device.

Exceptions

TYPE	CONDITION

TYPE	CONDITION
System.InvalidOperationException	Thrown when the property is accessed before the device has been initialized.

Methods

ChangeMultipleSwitchesChannelsAsync(IEnumerable<DeviceSwitchAndChannelPair>, CancellationToken)

Asynchronously change the channels of multiple connected switches.

Declaration

```
Task ChangeMultipleSwitchesChannelsAsync(IEnumerable<DeviceSwitchAndChannelPair> deviceSwitchesAndChannels,
CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Collections.Generic.IEnumerable<DeviceSwitchAndChannelPair>	deviceSwitchesAndChannels	The set of device switches and corresponding channels to change them to. Switch index `0` for the internal switch; Switch indexes `1` or `2` for external switches attached to the device.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	The task representing the operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the device has no switch at any of the specified switch indexes.
System.ArgumentException	Thrown when any the specified channels is out of range for the corresponding device switch.

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device switch channel response does not match the expected response.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

ChangeSwitchChannelAsync(Int32, Int32, CancellationToken)

Asynchronously change the channel of a connected switch.

Declaration

```
Task ChangeSwitchChannelAsync(int switchIndex, int channel, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	switchIndex	The switch number of the switch. `0` for the internal switch; `1` or `2` for external switches attached to the device.
System.Int32	channel	The channel to switch to.
System.Threading.CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	The task representing the operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the device has no switch at the specified switch index.

TYPE	CONDITION
System.ArgumentException	Thrown when the specified channel is out of range for the device switch.
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device switch channel response does not match the expected response.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

GetSwitchChannelAsync(Int32, CancellationToken)

Asynchronously get the channel of a connected switch.

Declaration

```
Task<int> GetSwitchChannelAsync(int switchIndex, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	switchIndex	The switch number of the switch. The cancellation token for the asynchronous operation. The default value is `None`. `0` for the internal switch; `1` or `2` for external switches attached to the device.
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Int32>	The task representing the operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the device has no switch at the specified switch index.

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device switch channel response is not a valid switch channel.
System.TimeoutException	Thrown when a communication failure causes no response to be received from the device.

Class OSX

Provides a connection to an OSX along with properties of the device and operations that can be performed using it.

> **NOTE**
>
> This class cannot not be used directly. It can only be used when returned:
>
> 1. As part of a list of devices by a DetectHardwareAsync(EConnectionTypesToDetect) or DetectHardwareAsync<T>(EConnectionTypesToDetect) call.
> 2. As a device by a TryDetectIPDeviceAsync(IPAddress) or TryDetectIPDeviceAsync<T>(IPAddress) call.

Inheritance

System.Object
PhysicalDevice
SantecDevice
OSX

Implements

IOSX
ISantecDevice
IModuleController
IDevice<EOSXModel>
IDevice
System.IDisposable

Inherited Members

SantecDevice.HandleIDNResponse(String[])
SantecDevice.GetIPAddressAsync(CancellationToken)
SantecDevice.SetIPAddressAsync(IPAddress, CancellationToken)
SantecDevice.EnableDHCPAsync(CancellationToken)
SantecDevice.GetGatewayAsync(CancellationToken)
SantecDevice.SetGatewayAsync(IPAddress, CancellationToken)
SantecDevice.GetSubnetPrefixLengthAsync(CancellationToken)
SantecDevice.SetSubnetPrefixLengthAsync(Int32, CancellationToken)
SantecDevice.GetHostnameAsync(CancellationToken)
SantecDevice.SetHostnameAsync(String, CancellationToken)
SantecDevice.GetInteractionModeAsync(CancellationToken)
SantecDevice.SetInteractionModeAsync(EInteractionMode, CancellationToken)
PhysicalDevice.Address
PhysicalDevice.SerialNumber
PhysicalDevice.FirmwareVersion
PhysicalDevice.IsInitialized
PhysicalDevice.IsInitializing
PhysicalDevice.SetTimeout(ETimeout)
PhysicalDevice.GetTimeout()
PhysicalDevice.Dispose()
PhysicalDevice.Dispose(Boolean)
PhysicalDevice.PingAsync()
PhysicalDevice.ResetAsync()
PhysicalDevice.QueryAsync(String, CancellationToken)
PhysicalDevice.WriteAsync(String, CancellationToken)
PhysicalDevice.ReadAsync(CancellationToken)

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.Switching

Assembly: Santec.Hardware.dll

Syntax

```
public class OSX : SantecDevice, IOSX, ISantecDevice, IModuleController, IDevice<EOSXModel>, IDevice,
IDisposable
```

Properties

Model

Declaration

```
public EOSXModel Model { get; }
```

Property Value

TYPE	DESCRIPTION
EOSXModel	

Modules

Declaration

```
public IReadOnlyList<ISwitchModule> Modules { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<ISwitchModule>	

Name

Declaration

```
public override string Name { get; }
```

Property Value

TYPE	DESCRIPTION
System.String	

Overrides

PhysicalDevice.Name

Remarks

> **NOTE**
>
> This property will always return "OSX".

NumberOfModules

Declaration

```
public int NumberOfModules { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

Methods

ChangeAllModuleChannelsAsync(Int32, CancellationToken)

Declaration

```
public async Task ChangeAllModuleChannelsAsync(int channel, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	channel	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

ChangeModuleChannelAsync(Int32, Int32, CancellationToken)

Declaration

```
public Task ChangeModuleChannelAsync(int moduleIndex, int channel, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	moduleIndex	
System.Int32	channel	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

ChangeModuleCommonAsync(Int32, Int32, CancellationToken)

Declaration

```
public Task ChangeModuleCommonAsync(int moduleIndex, int common, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	moduleIndex	
System.Int32	common	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

GetModuleChannelAsync(Int32, CancellationToken)

Declaration

```
public Task<int> GetModuleChannelAsync(int moduleIndex, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	moduleIndex	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Int32>	

GetModuleCommonAsync(Int32, CancellationToken)

Declaration

```
public Task<int> GetModuleCommonAsync(int moduleIndex, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	moduleIndex	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Int32>	

InitializeAsync()

Declaration

```
public override async Task InitializeAsync()
```

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Overrides

SantecDevice.InitializeAsync()

RefreshAsync()

Declaration

```
public override async Task RefreshAsync()
```

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

Overrides

PhysicalDevice.RefreshAsync()

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Thrown when the method is called on an uninitialized OSX.
UnexpectedDeviceResponseException	Thrown when the OSX modules response is not a valid module count.
UnexpectedDeviceResponseException	Thrown when any OSX module information response contains invalid channel or common channel counts.

Uninitialize()

Declaration

```
public override void Uninitialize()
```

Overrides

PhysicalDevice.Uninitialize()

Explicit Interface Implementations

IModuleController.Modules

Declaration

```
IReadOnlyList<IModule> IModuleController.Modules { get; }
```

Returns

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<IModule>	

Implements

IOSX
ISantecDevice
IModuleController
IDevice<TModelType>
IDevice
System.IDisposable

Class SimulatedExternalDeviceSwitchInformation

Represents the properties of a simulated external device switch.

> **NOTE**
>
> This class is intended for testing/development purposes. It supports the creation of simulated external device switches for simulated RLMs.

Inheritance

System.Object
SimulatedExternalDeviceSwitchInformation

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.Switching

Assembly: Santec.Hardware.dll

Syntax

```
public class SimulatedExternalDeviceSwitchInformation
```

Constructors

SimulatedExternalDeviceSwitchInformation(Int32, String, Version, FiberInformation)

Initialize a new record of simulated external device switch information.

Declaration

```
public SimulatedExternalDeviceSwitchInformation(int numberOfChannels, string serialNumber, Version
firmwareVersion, FiberInformation fiber)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	numberOfChannels	The number of channels of the simulated external device switch.
System.String	serialNumber	The serial number of the simulated external device switch.
Version	firmwareVersion	The firmware version of the simulated external device switch.
FiberInformation	fiber	The fiber information of the simulated external device switch.

Exceptions

TYPE	CONDITION
System.ArgumentException	Thrown when the serial number is empty or whitespace.
System.ArgumentException	Thrown when the number of channels is less than 1.
System.ArgumentNullException	Thrown when the serial number, firmware version, or fiber information is null.

Properties

Fiber

Gets the fiber information of the simulated external device switch.

Declaration

```
public FiberInformation Fiber { get; }
```

Property Value

TYPE	DESCRIPTION
FiberInformation	The fiber information of the simulated external device switch.

FirmwareVersion

Gets the firmware version of the simulated external device switch.

Declaration

```
public Version FirmwareVersion { get; }
```

Property Value

TYPE	DESCRIPTION
Version	The firmware version of the simulated external device switch.

NumberOfChannels

Gets the number of channels of the simulated external device switch

Declaration

```
public int NumberOfChannels { get; }
```

Property Value

TYPE	DESCRIPTION

TYPE	DESCRIPTION
System.Int32	The number of channels of the simulated external device switch.

SerialNumber

Gets the serial number of the simulated external device switch.

Declaration

```
public string SerialNumber { get; }
```

Property Value

TYPE	DESCRIPTION
System.String	The serial number of the simulated external device switch.

Class SimulatedOSX

Provides a simulated OSX with configurable switch modules.

> **NOTE**
>
> This class is intended for testing/development purposes. It operates much quicker than an actual OSX. It can only be created using a SimulatedDeviceFactory. It should only be used when returned as part of a list of devices by a DetectHardwareAsync(EConnectionTypesToDetect) or DetectHardwareAsync<T>(EConnectionTypesToDetect) call to a SimulatedHardwareDetectionService.

Inheritance

System.Object
SimulatedSantecDevice
SimulatedOSX

Implements

ISimulatedDevice
IOSX
ISantecDevice
IModuleController
IDevice<EOSXModel>
IDevice
System.IDisposable

Inherited Members

SimulatedSantecDevice.creatingDevice
SimulatedSantecDevice.Dispose()
SimulatedSantecDevice.SerialNumber
SimulatedSantecDevice.FirmwareVersion
SimulatedSantecDevice.Address
SimulatedSantecDevice.IsInitialized
SimulatedSantecDevice.QueryResponses
SimulatedSantecDevice.ConfigureQueryResponses(IEnumerable<QueryResponsePair>)
SimulatedSantecDevice.GetTimeout()
SimulatedSantecDevice.SetTimeout(ETimeout)
SimulatedSantecDevice.GetInteractionModeAsync(CancellationToken)
SimulatedSantecDevice.GetIPAddressAsync(CancellationToken)
SimulatedSantecDevice.SetIPAddressAsync(IPAddress, CancellationToken)
SimulatedSantecDevice.EnableDHCPAsync(CancellationToken)
SimulatedSantecDevice.GetGatewayAsync(CancellationToken)
SimulatedSantecDevice.SetGatewayAsync(IPAddress, CancellationToken)
SimulatedSantecDevice.GetSubnetPrefixLengthAsync(CancellationToken)
SimulatedSantecDevice.SetSubnetPrefixLengthAsync(Int32, CancellationToken)
SimulatedSantecDevice.GetHostnameAsync(CancellationToken)
SimulatedSantecDevice.SetHostnameAsync(String, CancellationToken)
SimulatedSantecDevice.SetInteractionModeAsync(EInteractionMode, CancellationToken)
SimulatedSantecDevice.InitializeAsync()
SimulatedSantecDevice.PingAsync()
SimulatedSantecDevice.Uninitialize()
SimulatedSantecDevice.ResetAsync()
SimulatedSantecDevice.RefreshAsync()
SimulatedSantecDevice.QueryAsync(String, CancellationToken)
SimulatedSantecDevice.WriteAsync(String, CancellationToken)

SimulatedSantecDevice.ReadAsync(CancellationToken)
System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.Switching

Assembly: Santec.Hardware.dll

Syntax

```
public class SimulatedOSX : SimulatedSantecDevice, ISimulatedDevice, IOSX, ISantecDevice, IModuleController,
IDevice<EOSXModel>, IDevice, IDisposable
```

Properties

Model

Declaration

```
public EOSXModel Model { get; }
```

Property Value

TYPE	DESCRIPTION
EOSXModel	

Modules

Declaration

```
public IReadOnlyList<ISwitchModule> Modules { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<ISwitchModule>	

Name

Declaration

```
public override string Name { get; }
```

Property Value

TYPE	DESCRIPTION
System.String	

Overrides

SimulatedSantecDevice.Name

NumberOfModules

Declaration

```
public int NumberOfModules { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

Methods

ChangeAllModuleChannelsAsync(Int32, CancellationToken)

Declaration

```
public async Task ChangeAllModuleChannelsAsync(int channel, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	channel	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

ChangeModuleChannelAsync(Int32, Int32, CancellationToken)

Declaration

```
public Task ChangeModuleChannelAsync(int moduleIndex, int channel, CancellationToken cancellationToken)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	moduleIndex	
System.Int32	channel	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

ChangeModuleCommonAsync(Int32, Int32, CancellationToken)

Declaration

```
public Task ChangeModuleCommonAsync(int moduleIndex, int common, CancellationToken cancellationToken)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	moduleIndex	
System.Int32	common	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task	

GetModuleChannelAsync(Int32, CancellationToken)

Declaration

```
public Task<int> GetModuleChannelAsync(int moduleIndex, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	moduleIndex	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Int32>	

GetModuleCommonAsync(Int32, CancellationToken)

Declaration

```
public Task<int> GetModuleCommonAsync(int moduleIndex, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	moduleIndex	
System.Threading.CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Int32>	

Explicit Interface Implementations

IModuleController.Modules

Declaration

```
IReadOnlyList<IModule> IModuleController.Modules { get; }
```

Returns

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<IModule>	

Implements

ISimulatedDevice
IOSX
ISantecDevice
IModuleController
IDevice<TModelType>
IDevice
System.IDisposable

Namespace Santec.Hardware.Exceptions

Classes

AmbientLightDetectedException

The exception that is thrown when a device cannot determine a mapping of outputs to inputs.

CameraNotFoundException

The exception that is thrown when a RD-P camera cannot be found.

CompositeDeviceHardwareException

The exception that is thrown when a problem occurs with a composite device's component hardware.

DeviceTimeoutException

The exception that is thrown when a timeout occurs during a device operation.

PolarityMappingFailedException

The exception that is thrown when a device cannot determine a mapping of outputs to inputs.

UnableToPersistILReferenceException

Exception thrown when unable to persist IL reference for a device.

UnexpectedDeviceResponseException

The exception that is thrown when a response from a device does not match the expected format for the response.

VisaNotFoundException

The exception that is thrown when trying to detect USB devices and no VISA implementation can be found.

Class AmbientLightDetectedException

The exception that is thrown when a device cannot determine a mapping of outputs to inputs.

Inheritance

System.Object
System.Exception
AmbientLightDetectedException

Implements

System.Runtime.Serialization.ISerializable
System.Runtime.InteropServices._Exception

Inherited Members

System.Exception.GetBaseException()
System.Exception.ToString()
System.Exception.GetObjectData(System.Runtime.Serialization.SerializationInfo, System.Runtime.Serialization.StreamingContext)
System.Exception.GetType()
System.Exception.Message
System.Exception.Data
System.Exception.InnerException
System.Exception.TargetSite
System.Exception.StackTrace
System.Exception.HelpLink
System.Exception.Source
System.Exception.HResult
System.Exception.SerializeObjectState
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Exceptions

Assembly: Santec.Hardware.dll

Syntax

```
public class AmbientLightDetectedException : Exception, ISerializable, _Exception
```

Constructors

AmbientLightDetectedException(Int32, Mat)

Initializes a new instance of the AmbientLightFailedException class.

Declaration

```
public AmbientLightDetectedException(int fiber, Mat ambientLightImage)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	fiber	The fiber that ambient light was detected on.

TYPE	NAME	DESCRIPTION
OpenCvSharp.Mat	ambientLightImage	A composite image of the fiber.

AmbientLightDetectedException(String, Int32, Mat)

Initializes a new instance of the PolarityMappingFailedException class with the specified error message.

Declaration

```
public AmbientLightDetectedException(string message, int fiber, Mat ambientLightImage)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	message	The error message.
System.Int32	fiber	The fiber that ambient light was detected on.
OpenCvSharp.Mat	ambientLightImage	A composite image of the fiber.

Properties

AmbientLightImage

Get the composite image of the polarity mapping operation.

Declaration

```
public Mat AmbientLightImage { get; }
```

Property Value

TYPE	DESCRIPTION
OpenCvSharp.Mat	The composite image of the polarity mapping operation.

Remarks

This can be converted to a bitmap or bitmap source using the appropriate converter from OpenCvSharp4.Extensions or OpenCvSharp4.WpfExtensions.

Fiber

Get the fiber that was detected with ambient light during the mapping operation

Declaration

```
public int Fiber { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	The fiber that was detected with ambient light during the mapping operation.

Methods

Finalize()

Free the resources of the PolarityMappingFailedException.

Declaration

```
protected void Finalize()
```

Implements

System.Runtime.Serialization.ISerializable
System.Runtime.InteropServices._Exception

Class CameraNotFoundException

The exception that is thrown when a RD-P camera cannot be found.

Inheritance

System.Object
System.Exception
CameraNotFoundException

Implements

System.Runtime.Serialization.ISerializable
System.Runtime.InteropServices._Exception

Inherited Members

System.Exception.GetBaseException()
System.Exception.ToString()
System.Exception.GetObjectData(System.Runtime.Serialization.SerializationInfo, System.Runtime.Serialization.StreamingContext)
System.Exception.GetType()
System.Exception.Message
System.Exception.Data
System.Exception.InnerException
System.Exception.TargetSite
System.Exception.StackTrace
System.Exception.HelpLink
System.Exception.Source
System.Exception.HResult
System.Exception.SerializeObjectState
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Exceptions

Assembly: Santec.Hardware.dll

Syntax

```
public class CameraNotFoundException : Exception, ISerializable, _Exception
```

Constructors

CameraNotFoundException()

Initializes a new instance of the CameraNotFoundException class.

Declaration

```
public CameraNotFoundException()
```

CameraNotFoundException(String)

Initializes a new instance of the CameraNotFoundException class with the specified error message.

Declaration

```
public CameraNotFoundException(string message)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	message	The error message.

CameraNotFoundException(String, Exception)

Initializes a new instance of the CameraNotFoundException class with the specified error message and the inner exception that caused the CameraNotFoundException.

Declaration

```
public CameraNotFoundException(string message, Exception innerException)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	message	The error message.
System.Exception	innerException	The inner exception.

Implements

System.Runtime.Serialization.ISerializable
System.Runtime.InteropServices._Exception

Class CompositeDeviceHardwareException

The exception that is thrown when a problem occurs with a composite device's component hardware.

Inheritance

System.Object
System.Exception
CompositeDeviceHardwareException

Implements

System.Runtime.Serialization.ISerializable
System.Runtime.InteropServices._Exception

Inherited Members

System.Exception.GetBaseException()
System.Exception.ToString()
System.Exception.GetObjectData(System.Runtime.Serialization.SerializationInfo, System.Runtime.Serialization.StreamingContext)
System.Exception.GetType()
System.Exception.Message
System.Exception.Data
System.Exception.InnerException
System.Exception.TargetSite
System.Exception.StackTrace
System.Exception.HelpLink
System.Exception.Source
System.Exception.HResult
System.Exception.SerializeObjectState
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Exceptions

Assembly: Santec.Hardware.dll

Syntax

```
public class CompositeDeviceHardwareException : Exception, ISerializable, _Exception
```

Constructors

CompositeDeviceHardwareException()

Initializes a new instance of the CompositeDeviceHardwareException class.

Declaration

```
public CompositeDeviceHardwareException()
```

CompositeDeviceHardwareException(String)

Initializes a new instance of the CompositeDeviceHardwareException class with the specified error message.

Declaration

```
public CompositeDeviceHardwareException(string message)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	message	The error message.

CompositeDeviceHardwareException(String, Exception)

Initializes a new instance of the CameraNotFoundECompositeDeviceHardwareExceptionxception class with the specified error message and the inner exception that caused the CompositeDeviceHardwareException.

Declaration

```
public CompositeDeviceHardwareException(string message, Exception innerException)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	message	The error message.
System.Exception	innerException	The inner exception.

Implements

System.Runtime.Serialization.ISerializable
System.Runtime.InteropServices._Exception

Class DeviceTimeoutException

The exception that is thrown when a timeout occurs during a device operation.

Inheritance

System.Object
System.Exception
DeviceTimeoutException

Implements

System.Runtime.Serialization.ISerializable
System.Runtime.InteropServices._Exception

Inherited Members

System.Exception.GetBaseException()
System.Exception.ToString()
System.Exception.GetObjectData(System.Runtime.Serialization.SerializationInfo, System.Runtime.Serialization.StreamingContext)
System.Exception.GetType()
System.Exception.Message
System.Exception.Data
System.Exception.InnerException
System.Exception.TargetSite
System.Exception.StackTrace
System.Exception.HelpLink
System.Exception.Source
System.Exception.HResult
System.Exception.SerializeObjectState
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Exceptions

Assembly: Santec.Hardware.dll

Syntax

```
public class DeviceTimeoutException : Exception, ISerializable, _Exception
```

Constructors

DeviceTimeoutException(IDevice)

Initializes a new instance of the DeviceTimeoutException class.

Declaration

```
public DeviceTimeoutException(IDevice device)
```

Parameters

TYPE	NAME	DESCRIPTION
IDevice	device	The device that caused the exception.

DeviceTimeoutException(IDevice, String)

Initializes a new instance of the DeviceTimeoutException class with the specified error message.

Declaration

```
public DeviceTimeoutException(IDevice device, string message)
```

Parameters

TYPE	NAME	DESCRIPTION
IDevice	device	The device that caused the exception.
System.String	message	The error message.

DeviceTimeoutException(IDevice, String, Exception)

Initializes a new instance of the DeviceTimeoutException class with the specified error message and the inner exception that caused the DeviceTimeoutException.

Declaration

```
public DeviceTimeoutException(IDevice device, string message, Exception innerException)
```

Parameters

TYPE	NAME	DESCRIPTION
IDevice	device	The device that caused the exception.
System.String	message	The error message.
System.Exception	innerException	The inner exception.

Properties

Device

Gets the device that caused the exception.

Declaration

```
public IDevice Device { get; }
```

Property Value

TYPE	DESCRIPTION
IDevice	The device that caused the exception.

Implements

System.Runtime.Serialization.ISerializable
System.Runtime.InteropServices._Exception

Class PolarityMappingFailedException

The exception that is thrown when a device cannot determine a mapping of outputs to inputs.

Inheritance

System.Object
System.Exception
PolarityMappingFailedException

Implements

System.Runtime.Serialization.ISerializable
System.Runtime.InteropServices._Exception

Inherited Members

System.Exception.GetBaseException()
System.Exception.ToString()
System.Exception.GetObjectData(System.Runtime.Serialization.SerializationInfo, System.Runtime.Serialization.StreamingContext)
System.Exception.GetType()
System.Exception.Message
System.Exception.Data
System.Exception.InnerException
System.Exception.TargetSite
System.Exception.StackTrace
System.Exception.HelpLink
System.Exception.Source
System.Exception.HResult
System.Exception.SerializeObjectState
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Exceptions

Assembly: Santec.Hardware.dll

Syntax

```
public class PolarityMappingFailedException : Exception, ISerializable, _Exception
```

Constructors

PolarityMappingFailedException(IEnumerable<Int32>, List<Mat>)

Initializes a new instance of the PolarityMappingFailedException class.

Declaration

```
public PolarityMappingFailedException(IEnumerable<int> detectedFibers, List<Mat>
polarityMappingDetectorImages)
```

Parameters

TYPE	NAME	DESCRIPTION

TYPE	NAME	DESCRIPTION
System.Collections.Generic.IEnumerable<System.Int32>	detectedFibers	The fibers that were detected during the polarity mapping operation.
System.Collections.Generic.List<OpenCvSharp.Mat>	polarityMappingDetectorImages	A list of the composite images of the polarity mapping operation for each detector.

PolarityMappingFailedException(String, IEnumerable<Int32>, List<Mat>)

Initializes a new instance of the PolarityMappingFailedException class with the specified error message.

Declaration

```
public PolarityMappingFailedException(string message, IEnumerable<int> detectedFibers, List<Mat>
polarityMappingDetectorImages)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	message	The error message.
System.Collections.Generic.IEnumerable<System.Int32>	detectedFibers	The fibers that were detected during the polarity mapping operation.
System.Collections.Generic.List<OpenCvSharp.Mat>	polarityMappingDetectorImages	A list of the composite images of the polarity mapping operation for each detector.

PolarityMappingFailedException(String, Exception, IEnumerable<Int32>, List<Mat>)

Initializes a new instance of the PolarityMappingFailedException class with the specified error message and the inner exception that caused the PolarityMappingFailedException.

Declaration

```
public PolarityMappingFailedException(string message, Exception innerException, IEnumerable<int>
detectedFibers, List<Mat> polarityMappingDetectorImages)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	message	The error message.
System.Exception	innerException	The inner exception.

TYPE	NAME	DESCRIPTION
System.Collections.Generic.IEnumerable<System.Int32>	detectedFibers	The fibers that were detected during the polarity mapping operation.
System.Collections.Generic.List<OpenCvSharp.Mat>	polarityMappingDetectorImages	A list of the composite images of the polarity mapping operation for each detector.

Properties

DetectedFibers

Get the fibers that were detected during the polarity mapping operation.

Declaration

```
public IReadOnlyList<int> DetectedFibers { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<System.Int32>	A list of the fibers that were detected during the polarity mapping operation.

PolarityMappingDetectorImages

Get the list of composite images of the polarity mapping operation for each detector.

Declaration

```
public IReadOnlyList<Mat> PolarityMappingDetectorImages { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<OpenCvSharp.Mat>	A list of the composite images of the polarity mapping operation for eah detector.

Remarks

These can be converted to bitmaps or bitmap sources using the appropriate converter from OpenCvSharp4.Extensions or OpenCvSharp4.WpfExtensions.

Methods

Finalize()

Free the resources of the PolarityMappingFailedException.

Declaration

```
protected void Finalize()
```

Implements

System.Runtime.Serialization.ISerializable
System.Runtime.InteropServices._Exception

Class UnableToPersistILReferenceException

Exception thrown when unable to persist IL reference for a device.

Inheritance

System.Object
System.Exception
UnableToPersistILReferenceException

Implements

System.Runtime.Serialization.ISerializable
System.Runtime.InteropServices._Exception

Inherited Members

System.Exception.GetBaseException()
System.Exception.ToString()
System.Exception.GetObjectData(System.Runtime.Serialization.SerializationInfo, System.Runtime.Serialization.StreamingContext)
System.Exception.GetType()
System.Exception.Message
System.Exception.Data
System.Exception.InnerException
System.Exception.TargetSite
System.Exception.StackTrace
System.Exception.HelpLink
System.Exception.Source
System.Exception.HResult
System.Exception.SerializeObjectState
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Exceptions

Assembly: Santec.Hardware.dll

Syntax

```
public class UnableToPersistILReferenceException : Exception, ISerializable, _Exception
```

Constructors

UnableToPersistILReferenceException(IDevice)

Initializes a new instance of the UnableToPersistILReferenceException class.

Declaration

```
public UnableToPersistILReferenceException(IDevice device)
```

Parameters

TYPE	NAME	DESCRIPTION
IDevice	device	The device associated with the exception.

UnableToPersistILReferenceException(IDevice, String)

Initializes a new instance of the UnableToPersistILReferenceException class with a specified error message.

Declaration

```
public UnableToPersistILReferenceException(IDevice device, string message)
```

Parameters

TYPE	NAME	DESCRIPTION
IDevice	device	The device associated with the exception.
System.String	message	The message that describes the error.

UnableToPersistILReferenceException(IDevice, String, Exception)

Initializes a new instance of the UnableToPersistILReferenceException class with a specified error message and a reference to the inner exception that is the cause of this exception.

Declaration

```
public UnableToPersistILReferenceException(IDevice device, string message, Exception innerException)
```

Parameters

TYPE	NAME	DESCRIPTION
IDevice	device	The device associated with the exception.
System.String	message	The message that describes the error.
System.Exception	innerException	The exception that is the cause of the current exception.

Properties

Device

Gets the device associated with the exception.

Declaration

```
public IDevice Device { get; }
```

Property Value

TYPE	DESCRIPTION
IDevice	

Implements

System.Runtime.Serialization.ISerializable
System.Runtime.InteropServices._Exception

Class UnexpectedDeviceResponseException

The exception that is thrown when a response from a device does not match the expected format for the response.

Inheritance

System.Object
System.Exception
UnexpectedDeviceResponseException

Implements

System.Runtime.Serialization.ISerializable
System.Runtime.InteropServices._Exception

Inherited Members

System.Exception.GetBaseException()
System.Exception.ToString()
System.Exception.GetObjectData(System.Runtime.Serialization.SerializationInfo, System.Runtime.Serialization.StreamingContext)
System.Exception.GetType()
System.Exception.Message
System.Exception.Data
System.Exception.InnerException
System.Exception.TargetSite
System.Exception.StackTrace
System.Exception.HelpLink
System.Exception.Source
System.Exception.HResult
System.Exception.SerializeObjectState
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Exceptions

Assembly: Santec.Hardware.dll

Syntax

```
public class UnexpectedDeviceResponseException : Exception, ISerializable, _Exception
```

Constructors

UnexpectedDeviceResponseException(IDevice)

Initializes a new instance of the UnexpectedDeviceResponseException class.

Declaration

```
public UnexpectedDeviceResponseException(IDevice device)
```

Parameters

TYPE	NAME	DESCRIPTION
IDevice	device	The device that caused the exception.

UnexpectedDeviceResponseException(IDevice, String)

Initializes a new instance of the UnexpectedDeviceResponseException class with the specified error message.

Declaration

```
public UnexpectedDeviceResponseException(IDevice device, string message)
```

Parameters

TYPE	NAME	DESCRIPTION
IDevice	device	The device that caused the exception.
System.String	message	The error message.

UnexpectedDeviceResponseException(IDevice, String, Exception)

Initializes a new instance of the UnexpectedDeviceResponseException class with the specified error message and the inner exception that caused the UnexpectedDeviceResponseException.

Declaration

```
public UnexpectedDeviceResponseException(IDevice device, string message, Exception innerException)
```

Parameters

TYPE	NAME	DESCRIPTION
IDevice	device	The device that caused the exception.
System.String	message	The error message.
System.Exception	innerException	The inner exception.

Properties

Device

Gets the device that caused the exception.

Declaration

```
public IDevice Device { get; }
```

Property Value

TYPE	DESCRIPTION
IDevice	The device that caused the exception.

Implements

System.Runtime.Serialization.ISerializable
System.Runtime.InteropServices._Exception

Class VisaNotFoundException

The exception that is thrown when trying to detect USB devices and no VISA implementation can be found.

Inheritance

System.Object
System.Exception
VisaNotFoundException

Implements

System.Runtime.Serialization.ISerializable
System.Runtime.InteropServices._Exception

Inherited Members

System.Exception.GetBaseException()
System.Exception.ToString()
System.Exception.GetObjectData(System.Runtime.Serialization.SerializationInfo, System.Runtime.Serialization.StreamingContext)
System.Exception.GetType()
System.Exception.Message
System.Exception.Data
System.Exception.InnerException
System.Exception.TargetSite
System.Exception.StackTrace
System.Exception.HelpLink
System.Exception.Source
System.Exception.HResult
System.Exception.SerializeObjectState
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Exceptions

Assembly: Santec.Hardware.dll

Syntax

```
public class VisaNotFoundException : Exception, ISerializable, _Exception
```

Constructors

VisaNotFoundException()

Initializes a new instance of the UnexpectedDeVisaNotFoundExceptionviceResponseException class.

Declaration

```
public VisaNotFoundException()
```

VisaNotFoundException(String)

Initializes a new instance of the VisaNotFoundException class with the specified error message.

Declaration

```
public VisaNotFoundException(string message)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	message	The error message.

VisaNotFoundException(String, Exception)

Initializes a new instance of the VisaNotFoundException class with the specified error message and the inner exception that caused the VisaNotFoundException.

Declaration

```
public VisaNotFoundException(string message, Exception innerException)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	message	The error message.
System.Exception	innerException	The inner exception.

Implements

System.Runtime.Serialization.ISerializable
System.Runtime.InteropServices._Exception

Namespace Santec.Hardware.Factories

Classes

ConverterFactory

Provides a factory for creating converters.

Class ConverterFactory

Provides a factory for creating converters.

Inheritance

System.Object
ConverterFactory

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Factories

Assembly: Santec.Hardware.dll

Syntax

```
public class ConverterFactory
```

Constructors

ConverterFactory()

Initializes a new converter factory.

Declaration

```
public ConverterFactory()
```

ConverterFactory(ICultureService)

Initializes a new converter factory.

Declaration

```
public ConverterFactory(ICultureService cultureService)
```

Parameters

TYPE	NAME	DESCRIPTION
Santec.Core.Globalization.ICultureService	cultureService	The culture service for localization.

Methods

CreateFiberTypeToStringConverter()

Creates new test block action to string converter.

Declaration

```
public FiberTypeToStringConverter CreateFiberTypeToStringConverter()
```

Returns

TYPE	DESCRIPTION
FiberTypeToStringConverter	

Namespace Santec.Hardware.Services

Classes

HardwareDetectionService

Provides a service to detect Santec Canada hardware devices.

SimulatedHardwareDetectionService

Provides a service to detect real and simulated Santec Canada hardware devices.

> **NOTE**
>
> This class is intended for testing/development purposes. The ConfigureDevices(IEnumerable<IDevice>) method must be called to setup simulated devices. The ConfigureRealHardwareDetection(Boolean) method can be used to enable or disable the detection of real devices.

Interfaces

IHardwareDetectionService

Defines a service to detect Santec Canada hardware devices

Enums

EConnectionTypesToDetect

Specifies the type(s) of connections to detect devices on.

Enum EConnectionTypesToDetect

Specifies the type(s) of connections to detect devices on.

Namespace: Santec.Hardware.Services

Assembly: Santec.Hardware.dll

Syntax

```
public enum EConnectionTypesToDetect
```

Fields

NAME	DESCRIPTION
All	All connection types.
IP	IP connections.
USB	USB connections.

Class HardwareDetectionService

Provides a service to detect Santec Canada hardware devices.

Inheritance

System.Object
HardwareDetectionService

Implements

IHardwareDetectionService

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Services

Assembly: Santec.Hardware.dll

Syntax

```
public class HardwareDetectionService : IHardwareDetectionService
```

Constructors

HardwareDetectionService()

Initialize a new hardware detection service.

Declaration

```
public HardwareDetectionService()
```

HardwareDetectionService(ICultureService)

Initialize a new hardware detection service.

Declaration

```
public HardwareDetectionService(ICultureService cultureService)
```

Parameters

TYPE	NAME	DESCRIPTION
Santec.Core.Globalization.ICultureService	cultureService	The culture service for localization.

Properties

CanDetectUSBDevices

Declaration

```
public bool CanDetectUSBDevices { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	

Methods

CheckForVISA()

Declaration

```
public bool CheckForVISA()
```

Returns

TYPE	DESCRIPTION
System.Boolean	

DetectHardwareAsync(EConnectionTypesToDetect)

Declaration

```
public async Task<List<IDevice>> DetectHardwareAsync(EConnectionTypesToDetect connectionTypesToDetect =
EConnectionTypesToDetect.All)
```

Parameters

TYPE	NAME	DESCRIPTION
EConnectionTypesToDetect	connectionTypesToDetect	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Collections.Generic.List<IDevice>>	

DetectHardwareAsync<T>(EConnectionTypesToDetect)

Declaration

```
public async Task<List<T>> DetectHardwareAsync<T>(EConnectionTypesToDetect connectionTypesToDetect =
EConnectionTypesToDetect.All)

    where T : IDevice
```

Parameters

TYPE	NAME	DESCRIPTION
EConnectionTypesToDetect	connectionTypesToDetect	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Collections.Generic.List<T>>	

Type Parameters

NAME	DESCRIPTION
T	

DetectmDNSDevices(String)

Detect all mDNS hosts.

Declaration

```
protected virtual IReadOnlyList<IZeroconfHost> DetectmDNSDevices(string serviceType)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	serviceType	The service type to detect mDNS devices for.

Returns

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<Zeroconf.IZeroconfHost>	A list of detected mDNS hosts for the specified service type.

DetectmDNSDevicesAsync(String)

Asynchronously detect all mDNS hosts.

Declaration

```
protected virtual Task<IReadOnlyList<IZeroconfHost>> DetectmDNSDevicesAsync(string serviceType)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	serviceType	The service type to detect mDNS devices for.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Collections.Generic.IReadOnlyList<Zeroconf.IZeroconfHost>>	A list of detected mDNS hosts for the specified service type.

DetectUSBDevicesAsync()

Detect all Santec Canada USB devices.

Declaration

```
protected virtual async Task<List<IDevice>> DetectUSBDevicesAsync()
```

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Collections.Generic.List<IDevice>>	A list of all detected Santec Canada USB devices.

DetectUSBResources()

Detect all USB resources.

Declaration

```
protected virtual IReadOnlyList<string> DetectUSBResources()
```

Returns

TYPE	DESCRIPTION
System.Collections.Generic.IReadOnlyList<System.String>	A list of all detected USB resource identifiers.

TryDetectIPDeviceAsync(IPAddress)

Declaration

```
public async Task<(bool Success, IDevice DetectedDevice)> TryDetectIPDeviceAsync(IPAddress ipAddress)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Net.IPAddress	ipAddress	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.ValueTuple<System.Boolean, IDevice>>	

TryDetectIPDeviceAsync<T>(IPAddress)

Declaration

```
public async Task<(bool Success, T DetectedDevice)> TryDetectIPDeviceAsync<T>(IPAddress ipAddress)
    where T : class, IDevice
```

Parameters

TYPE	NAME	DESCRIPTION
System.Net.IPAddress	ipAddress	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.ValueTuple<System.Boolean, T>>	

Type Parameters

NAME	DESCRIPTION
T	

WrappedVISACall()

A wrapped VISA call that will properly determine if a VISA implementation is installed on the system.

Declaration

```
protected virtual void WrappedVISACall()
```

Implements

IHardwareDetectionService

Interface IHardwareDetectionService

Defines a service to detect Santec Canada hardware devices

Namespace: Santec.Hardware.Services

Assembly: Santec.Hardware.dll

Syntax

```
public interface IHardwareDetectionService
```

Properties

CanDetectUSBDevices

Get whether the hardware detection service can find a VISA implementation and detect USB devices.

Declaration

```
bool CanDetectUSBDevices { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	`true` if the service can detect USB devices; otherwise, `false`.

Methods

CheckForVISA()

Check for a VISA implementation on the system and update CanDetectUSBDevices accordingly.

Declaration

```
bool CheckForVISA()
```

Returns

TYPE	DESCRIPTION
System.Boolean	`true` if a VISA implementation is found; otherwise, `false`.

DetectHardwareAsync(EConnectionTypesToDetect)

Asynchronously detect Santec Canada hardware devices.

Declaration

```
Task<List<IDevice>> DetectHardwareAsync(EConnectionTypesToDetect connectionTypesToDetect =
EConnectionTypesToDetect.All)
```

Parameters

TYPE	NAME	DESCRIPTION
EConnectionTypesToDetect	connectionTypesToDetect	The connection types to detect devices on. Default is All.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Collections.Generic.List<IDevice>>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
VisaNotFoundException	Thrown when trying to detect USB devices or all connection types and no VISA implementation can be found on the system.

DetectHardwareAsync<T>(EConnectionTypesToDetect)

Asynchronously detect all Santec Canada devices of a specific type.

Declaration

```
Task<List<T>> DetectHardwareAsync<T>(EConnectionTypesToDetect connectionTypesToDetect =
EConnectionTypesToDetect.All)

    where T : IDevice
```

Parameters

TYPE	NAME	DESCRIPTION
EConnectionTypesToDetect	connectionTypesToDetect	The connection types to detect devices on. Default is All.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Collections.Generic.List<T>>	The task representing the asynchronous operation.

Type Parameters

NAME	DESCRIPTION

NAME	DESCRIPTION
T	The type of the devices to detect.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
VisaNotFoundException	Thrown when trying to detect USB devices or all connection types and no VISA implementation can be found on the system.

TryDetectIPDeviceAsync(IPAddress)

Asynchronously try to detect a Santec Canada device at a specific IP address.

Declaration

```
Task<(bool Success, IDevice DetectedDevice)> TryDetectIPDeviceAsync(IPAddress ipAddress)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Net.IPAddress	ipAddress	The IP address to probe for the device.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.ValueTuple<System.Boolean, IDevice>>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

TryDetectIPDeviceAsync<T>(IPAddress)

Asynchronously try to detect a Santec Canada device of a specific type at a specific IP address.

Declaration

```
Task<(bool Success, T DetectedDevice)> TryDetectIPDeviceAsync<T>(IPAddress ipAddress)

    where T : class, IDevice
```

Parameters

TYPE	NAME	DESCRIPTION
System.Net.IPAddress	ipAddress	The IP address to probe for the device.

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.ValueTuple<System.Boolean, T>>	The task representing the asynchronous operation.

Type Parameters

NAME	DESCRIPTION
T	The type of the device to detect.

Remarks

NOTE

This operation will not block.

Class SimulatedHardwareDetectionService

Provides a service to detect real and simulated Santec Canada hardware devices.

> **NOTE**
>
> This class is intended for testing/development purposes. The ConfigureDevices(IEnumerable<IDevice>) method must be called to setup simulated devices. The ConfigureRealHardwareDetection(Boolean) method can be used to enable or disable the detection of real devices.

Inheritance

System.Object
SimulatedHardwareDetectionService

Implements

IHardwareDetectionService

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Santec.Hardware.Services

Assembly: Santec.Hardware.dll

Syntax

```
public class SimulatedHardwareDetectionService : IHardwareDetectionService
```

Constructors

SimulatedHardwareDetectionService()

Initialize a new simulated hardware detection service.

Declaration

```
public SimulatedHardwareDetectionService()
```

SimulatedHardwareDetectionService(ICultureService)

Initialize a new simulated hardware detection service.

Declaration

```
public SimulatedHardwareDetectionService(ICultureService cultureService)
```

Parameters

TYPE	NAME	DESCRIPTION
Santec.Core.Globalization.ICultureService	cultureService	The culture service for localization.

Properties

CanDetectUSBDevices

Declaration

```
public bool CanDetectUSBDevices { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	

DetectRealHardware

Get whether the service will detect real devices.

The default value is `true`. Use ConfigureRealHardwareDetection(Boolean) to set this value.

Declaration

```
public bool DetectRealHardware { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	`true` if the service will detect real devices; otherwise, `false`.

Methods

CheckForVISA()

Declaration

```
public bool CheckForVISA()
```

Returns

TYPE	DESCRIPTION
System.Boolean	

ConfigureDevices(IEnumerable<IDevice>)

Configure the service with a set of devices.

> **NOTE**
>
> The devices will be returned by the DetectHardwareAsync(EConnectionTypesToDetect) and DetectHardwareAsync<T>(EConnectionTypesToDetect) methods, as appropriate.

Declaration

```
public void ConfigureDevices(IEnumerable<IDevice> devices)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Collections.Generic.IEnumerable<IDevice>	devices	The set of devices to configure the service with.

ConfigureRealHardwareDetection(Boolean)

Configure whether the service will detect real devices.

Declaration

```
public void ConfigureRealHardwareDetection(bool detectRealHardware)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Boolean	detectRealHardware	If `true`, the service will be configured to detect real devices.

ConfigureUSBDeviceDetectionOverride(Boolean, Boolean)

Configure whether the service will override the CanDetectUSBDevices and CheckForVISA() responses.

Declaration

```
public void ConfigureUSBDeviceDetectionOverride(bool overrideUSBDetection, bool overrideUSBDetectionValue =
false)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Boolean	overrideUSBDetection	If `true`, the service will be configured to override the USB detection responses; otherwise, USB detection will occur normally.
System.Boolean	overrideUSBDetectionValue	The value to override the USB detection responses with. If `true`, the service will always report it can detect USB devices; otherwise, the service will always report it cannot detect USB devices.

DetectHardwareAsync(EConnectionTypesToDetect)

Declaration

```
public async Task<List<IDevice>> DetectHardwareAsync(EConnectionTypesToDetect connectionTypesToDetect =
EConnectionTypesToDetect.All)
```

Parameters

TYPE	NAME	DESCRIPTION
EConnectionTypesToDetect	connectionTypesToDetect	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Collections.Generic.List<IDevice>>	

DetectHardwareAsync<T>(EConnectionTypesToDetect)

Declaration

```
public async Task<List<T>> DetectHardwareAsync<T>(EConnectionTypesToDetect connectionTypesToDetect =
EConnectionTypesToDetect.All)

    where T : IDevice
```

Parameters

TYPE	NAME	DESCRIPTION
EConnectionTypesToDetect	connectionTypesToDetect	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.Collections.Generic.List<T>>	

Type Parameters

NAME	DESCRIPTION
T	

TryDetectIPDeviceAsync(IPAddress)

Declaration

```
public async Task<(bool Success, IDevice DetectedDevice)> TryDetectIPDeviceAsync(IPAddress ipAddress)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Net.IPAddress	ipAddress	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.ValueTuple<System.Boolean, IDevice>>	

Remarks

NOTE

This operation will not block.

NOTE

This method will never return success if DetectRealHardware is `false`.

TryDetectIPDeviceAsync<T>(IPAddress)

Declaration

```
public async Task<(bool Success, T DetectedDevice)> TryDetectIPDeviceAsync<T>(IPAddress ipAddress)

    where T : class, IDevice
```

Parameters

TYPE	NAME	DESCRIPTION
System.Net.IPAddress	ipAddress	

Returns

TYPE	DESCRIPTION
System.Threading.Tasks.Task<System.ValueTuple<System.Boolean, T>>	

Type Parameters

NAME	DESCRIPTION
T	

Remarks

NOTE

This operation will not block.

NOTE

This method will never return success if DetectRealHardware is `false`.

Implements

IHardwareDetectionService