Santec Hardware Library Programming Guide

Version 0.20.0.0

Table of Contents

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

- 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/)

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);

            rdp.SetLossCompensation(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);

            rdp.SetLossCompensation(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);

                await rdp.CollectOutputMappingDataAsync(i + 1);
            }

            // 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);

rdp.SetLossCompensation(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);

rdp.SetLossCompensation(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);

    await rdp.CollectOutputMappingDataAsync(i + 1);
}

// 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. The output mapping data can be collected multiple times for each fiber if necessary. 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.

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 Hardware.Tests

Classes

RDPProcessingTests

SimulatedPowerMeterTests

Class RDPProcessingTests

Inheritance

Object

RDPProcessingTests

Inherited Members

Object.ToString()

Object.Equals(Object)

Object.Equals(Object, Object)

Object.ReferenceEquals(Object, Object)

Object.GetHashCode()

Object.GetType()

Object.MemberwiseClone()

Namespace: Hardware.Tests

Assembly: Hardware.Tests.dll

Syntax

```
[TestClass]
public class RDPProcessingTests
```

Constructors

RDPProcessingTests()

Declaration

```
public RDPProcessingTests()
```

Properties

TestContext

Declaration

```
public TestContext TestContext { get; set; }
```

Property Value

TYPE	DESCRIPTION
TestContext	

Methods

ProcessDetectionData_NoExpectedMapping_ReturnMapping(String, String)

Declaration

```
[DataTestMethod]
[DataRow(new object[]{"ProcessDetectionData_ReturnMapping_TestCases\\12A-AnalysisData",
"ProcessDetectionData_ReturnMapping_TestCases\\12A.csv"}, DisplayName = "12A MPO")]
[DataRow(new object[]{"ProcessDetectionData_ReturnMapping_TestCases\\12B-AnalysisData",
"ProcessDetectionData_ReturnMapping_TestCases\\12B.csv"}, DisplayName = "12B MPO")]
[DataRow(new object[]{"ProcessDetectionData_ReturnMapping_TestCases\\12C-AnalysisData",
"ProcessDetectionData_ReturnMapping_TestCases\\12C.csv"}, DisplayName = "12C MPO")]
[DataRow(new object[]{"ProcessDetectionData_ReturnMapping_TestCases\\24A-AnalysisData",
"ProcessDetectionData_ReturnMapping_TestCases\\24A.csv"}, DisplayName = "24A MPO")]
[DataRow(new object[]{"ProcessDetectionData_ReturnMapping_TestCases\\24B-AnalysisData",
```

```
"ProcessDetectionData_ReturnMapping_TestCases\\24B.csv"}, DisplayName = "24B MPO")]
[DataRow(new object[]{"ProcessDetectionData_ReturnMapping_TestCases\\12 Fiber-2nd Detection Pass-
AnalysisData", "ProcessDetectionData_ReturnMapping_TestCases\\12 Fiber-2nd Detection Pass.csv"}, DisplayName
= "12 Fiber MPO-2nd Detection Pass")]
[DataRow(new object[]{"ProcessDetectionData_ReturnMapping_TestCases\\8 Fiber-4 Light-4 Dark-4 Light-
AnalysisData", "ProcessDetectionData_ReturnMapping_TestCases\\8 Fiber-4 Light-4 Dark-4 Light.csv"},
DisplayName = "8 Fiber MPO-4 Light-4 Dark-4 Light")]
[DataRow(new object[]{"ProcessDetectionData_ReturnMapping_TestCases\\72 Fiber-AnalysisData",
"ProcessDetectionData_ReturnMapping_TestCases\\72 Fiber.csv"}, DisplayName = "72 Fiber MPO")]
[DataRow(new object[]{"ProcessDetectionData_ReturnMapping_TestCases\\6 Fiber-2-4-Offset-AnalysisData",
"ProcessDetectionData_ReturnMapping_TestCases\\6 Fiber-2-4-Offset.csv"}, DisplayName = "6 Fiber-2-4 Offset
MPO")]
[DataRow(new object[]{"ProcessDetectionData_ReturnMapping_TestCases\\No Fibers Detected-AnalysisData",
"ProcessDetectionData_ReturnMapping_TestCases\\No Fibers Detected.csv"}, DisplayName = "No Fibers
Detected")]
[DataRow(new object[]{"ProcessDetectionData_ReturnMapping_TestCases\\8 Fiber-1 Input Location-AnalysisData",
"ProcessDetectionData_ReturnMapping_TestCases\\8 Fiber-1 Input Location.csv"}, DisplayName = "8 Fiber MPO-1
Input Location")]
[DataRow(new object[]{"ProcessDetectionData_ReturnMapping_TestCases\\24 Fiber-Cable 1-AnalysisData",
"ProcessDetectionData_ReturnMapping_TestCases\\24 Fiber-Cable 1.csv"}, DisplayName = "24 Fiber Multi-Cable
MPO-Cable 1")]
[DataRow(new object[]{"ProcessDetectionData_ReturnMapping_TestCases\\24 Fiber-Cable 2-AnalysisData",
"ProcessDetectionData_ReturnMapping_TestCases\\24 Fiber-Cable 2.csv"}, DisplayName = "24 Fiber Multi-Cable
MPO-Cable 2")]
[DataRow(new object[]{"ProcessDetectionData_ReturnMapping_TestCases\\24 Fiber-Different Polarities-Cable 1-
AnalysisData", "ProcessDetectionData_ReturnMapping_TestCases\\24 Fiber-Different Polarities-Cable 1.csv"},
DisplayName = "24 Fiber Multi-Cable MPO-Different Polarities-Cable 1")]
[DataRow(new object[]{"ProcessDetectionData_ReturnMapping_TestCases\\24 Fiber-Different Polarities-Cable 2-
AnalysisData", "ProcessDetectionData_ReturnMapping_TestCases\\24 Fiber-Different Polarities-Cable 2.csv"},
DisplayName = "24 Fiber Multi-Cable MPO-Different Polarities-Cable 2")]
[DataRow(new object[]{"ProcessDetectionData_ReturnMapping_TestCases\\36 Fiber-Different Cable Lengths-Cable
1-AnalysisData", "ProcessDetectionData_ReturnMapping_TestCases\\36 Fiber-Different Cable Lengths-Cable
1.csv"}, DisplayName = "36 Fiber Multi-Cable MPO-Different Cable Lengths-Cable 1")]
[DataRow(new object[]{"ProcessDetectionData_ReturnMapping_TestCases\\36 Fiber-Different Cable Lengths-Cable
2-AnalysisData", "ProcessDetectionData_ReturnMapping_TestCases\\36 Fiber-Different Cable Lengths-Cable 2-No
Expected.csv"}, DisplayName = "36 Fiber Multi-Cable MPO-Different Cable Lengths-Cable 2")]
[DataRow(new object[]{"ProcessDetectionData_ReturnMapping_TestCases\\36 Fiber-Different Cable Lengths-Cable
3-AnalysisData", "ProcessDetectionData_ReturnMapping_TestCases\\36 Fiber-Different Cable Lengths-Cable
3.csv"}, DisplayName = "36 Fiber Multi-Cable MPO-Different Cable Lengths-Cable 3")]
[DataRow(new object[]{"ProcessDetectionData_ReturnMapping_TestCases\\12A Ribbon-AnalysisData",
"ProcessDetectionData_ReturnMapping_TestCases\\12A Ribbon.csv"}, DisplayName = "12A Ribbon")]
[DataRow(new object[]{"ProcessDetectionData_ReturnMapping_TestCases\\12B Ribbon-AnalysisData",
"ProcessDetectionData_ReturnMapping_TestCases\\12B Ribbon.csv"}, DisplayName = "12B Ribbon")]
[DataRow(new object[]{"ProcessDetectionData_ReturnMapping_TestCases\\12C Ribbon-AnalysisData",
"ProcessDetectionData_ReturnMapping_TestCases\\12C Ribbon.csv"}, DisplayName = "12C Ribbon")]
[DataRow(new object[]{"ProcessDetectionData_ReturnMapping_TestCases\\Duplex-1-AnalysisData",
"ProcessDetectionData_ReturnMapping_TestCases\\Duplex-1.csv"}, DisplayName = "Duplex LC-1")]
[DataRow(new object[]{"ProcessDetectionData_ReturnMapping_TestCases\\Duplex-2-AnalysisData",
"ProcessDetectionData_ReturnMapping_TestCases\\Duplex-2.csv"}, DisplayName = "Duplex LC-2")]
[DataRow(new object[]{"ProcessDetectionData_ReturnMapping_TestCases\\Duplex-Vertical Spread-AnalysisData",
"ProcessDetectionData_ReturnMapping_TestCases\\Duplex-Vertical Spread.csv"}, DisplayName = "Duplex LC-
Vertical Spread")]
[DataRow(new object[]{"ProcessDetectionData_ReturnMapping_TestCases\\3 Fiber-Multi Row-AnalysisData",
"ProcessDetectionData_ReturnMapping_TestCases\\3 Fiber-Multi Row.csv"}, DisplayName = "3 Fiber MPO-Multi
Row")]
public void ProcessDetectionData_NoExpectedMapping_ReturnMapping(string analysisDataPath, string
expectedMappingOperationMappingPath)
```

Parameters

TYPE	NAME	DESCRIPTION
String	analysisDataPath	

TYPE	NAME	DESCRIPTION
String	expectedMappingOperationMappingPath	

ProcessDetectionData_NoExpectedMapping_ThrowPolarityMappingFailedException(String)

Declaration

```
[DataTestMethod]
[DataRow(new object[]{"ProcessDetectionData_ThrowPolarityMappingFailedException_TestCases\\12 Fiber-A
Expected-B with 1 Dark Fiber-AnalysisData"}, DisplayName = "12 Fiber MPO-B with 1 Dark Fiber")]
[DataRow(new object[]{"ProcessDetectionData_ReturnMapping_TestCases\\12 Fiber-4 Light-4 Dark-4 Light-
AnalysisData"}, DisplayName = "12 Fiber MPO-4 Light-4 Dark-4 Light")]
[DataRow(new object[]{"ProcessDetectionData_ReturnMapping_TestCases\\Sparse 24 Fiber-AnalysisData"},
DisplayName = "Sparse 24 Fiber MPO")]
[DataRow(new object[]{"ProcessDetectionData_ReturnMapping_TestCases\\24 Fiber-Vertical-AnalysisData"},
DisplayName = "24 Fiber MPO-Vertical")]
public void ProcessDetectionData_NoExpectedMapping_ThrowPolarityMappingFailedException(string
analysisDataPath)
```

Parameters

TYPE	NAME	DESCRIPTION
String	analysisDataPath	

ProcessDetectionData_ReturnMapping(String, String)

Declaration

```
[DataTestMethod]
[DataRow(new object[]{"ProcessDetectionData_ReturnMapping_TestCases\\12A-AnalysisData",
"ProcessDetectionData_ReturnMapping_TestCases\\12A.csv"}, DisplayName = "12A MPO")]
[DataRow(new object[]{"ProcessDetectionData_ReturnMapping_TestCases\\12B-AnalysisData",
"ProcessDetectionData_ReturnMapping_TestCases\\12B.csv"}, DisplayName = "12B MPO")]
[DataRow(new object[]{"ProcessDetectionData_ReturnMapping_TestCases\\12C-AnalysisData",
"ProcessDetectionData_ReturnMapping_TestCases\\12C.csv"}, DisplayName = "12C MPO")]
[DataRow(new object[]{"ProcessDetectionData_ReturnMapping_TestCases\\24A-AnalysisData",
"ProcessDetectionData_ReturnMapping_TestCases\\24A.csv"}, DisplayName = "24A MPO")]
[DataRow(new object[]{"ProcessDetectionData_ReturnMapping_TestCases\\24B-AnalysisData",
"ProcessDetectionData_ReturnMapping_TestCases\\24B.csv"}, DisplayName = "24B MPO")]
[DataRow(new object[]{"ProcessDetectionData_ReturnMapping_TestCases\\12 Fiber-2nd Detection Pass-
AnalysisData", "ProcessDetectionData_ReturnMapping_TestCases\\12 Fiber-2nd Detection Pass.csv"}, DisplayName
= "12 Fiber MPO-2nd Detection Pass")]
[DataRow(new object[]{"ProcessDetectionData_ReturnMapping_TestCases\\8 Fiber-4 Light-4 Dark-4 Light-
AnalysisData", "ProcessDetectionData_ReturnMapping_TestCases\\8 Fiber-4 Light-4 Dark-4 Light.csv"},
DisplayName = "8 Fiber MPO-4 Light-4 Dark-4 Light")]
[DataRow(new object[]{"ProcessDetectionData_ReturnMapping_TestCases\\12 Fiber-4 Light-4 Dark-4 Light-
AnalysisData", "ProcessDetectionData_ReturnMapping_TestCases\\12 Fiber-4 Light-4 Dark-4 Light.csv"},
DisplayName = "12 Fiber MPO-4 Light-4 Dark-4 Light")]
[DataRow(new object[]{"ProcessDetectionData_ReturnMapping_TestCases\\12 Fiber-A Expected-B Detected-
AnalysisData", "ProcessDetectionData_ReturnMapping_TestCases\\12 Fiber-A Expected-B Detected.csv"},
DisplayName = "12 Fiber MPO-A Expected-B Detected")]
[DataRow(new object[]{"ProcessDetectionData_ReturnMapping_TestCases\\Sparse 24 Fiber-AnalysisData",
"ProcessDetectionData_ReturnMapping_TestCases\\Sparse 24 Fiber.csv"}, DisplayName = "Sparse 24 Fiber MPO")]
[DataRow(new object[]{"ProcessDetectionData_ReturnMapping_TestCases\\72 Fiber-AnalysisData",
"ProcessDetectionData_ReturnMapping_TestCases\\72 Fiber.csv"}, DisplayName = "72 Fiber MPO")]
[DataRow(new object[]{"ProcessDetectionData_ReturnMapping_TestCases\\6 Fiber-2-4-Offset-AnalysisData",
"ProcessDetectionData_ReturnMapping_TestCases\\6 Fiber-2-4-Offset.csv"}, DisplayName = "6 Fiber-2-4 Offset
MPO")]
[DataRow(new object[]{"ProcessDetectionData_ReturnMapping_TestCases\\All Fibers Detected-Detected does not
match Expected-AnalysisData", "ProcessDetectionData_ReturnMapping_TestCases\\All Fibers Detected-Detected
does not match Expected.csv"}, DisplayName = "All Fibers Detected-Detected does not match Expected")]
```

```
does not match Expected.csv"}, DisplayName = "All Fibers Detected-Detected does not match Expected")]
[DataRow(new object[]{"ProcessDetectionData_ReturnMapping_TestCases\\No Fibers Detected-AnalysisData",
"ProcessDetectionData_ReturnMapping_TestCases\\No Fibers Detected.csv"}, DisplayName = "No Fibers
Detected")]
[DataRow(new object[]{"ProcessDetectionData_ReturnMapping_TestCases\\8 Fiber-1 Input Location-AnalysisData",
"ProcessDetectionData_ReturnMapping_TestCases\\8 Fiber-1 Input Location.csv"}, DisplayName = "8 Fiber MPO-1
Input Location")]
[DataRow(new object[]{"ProcessDetectionData_ReturnMapping_TestCases\\8 Fiber-2-9-AnalysisData",
"ProcessDetectionData_ReturnMapping_TestCases\\8 Fiber-2-9.csv"}, DisplayName = "8 Fiber MPO-2-9")]
[DataRow(new object[]{"ProcessDetectionData_ReturnMapping_TestCases\\24 Fiber-Cable 1-AnalysisData",
"ProcessDetectionData_ReturnMapping_TestCases\\24 Fiber-Cable 1.csv"}, DisplayName = "24 Fiber Multi-Cable
MPO-Cable 1")]
[DataRow(new object[]{"ProcessDetectionData_ReturnMapping_TestCases\\24 Fiber-Cable 2-AnalysisData",
"ProcessDetectionData_ReturnMapping_TestCases\\24 Fiber-Cable 2.csv"}, DisplayName = "24 Fiber Multi-Cable
MPO-Cable 2")]
[DataRow(new object[]{"ProcessDetectionData_ReturnMapping_TestCases\\24 Fiber-Different Polarities-Cable 1-
AnalysisData", "ProcessDetectionData_ReturnMapping_TestCases\\24 Fiber-Different Polarities-Cable 1.csv"},
DisplayName = "24 Fiber Multi-Cable MPO-Different Polarities-Cable 1")]
[DataRow(new object[]{"ProcessDetectionData_ReturnMapping_TestCases\\24 Fiber-Different Polarities-Cable 2-
AnalysisData", "ProcessDetectionData_ReturnMapping_TestCases\\24 Fiber-Different Polarities-Cable 2.csv"},
DisplayName = "24 Fiber Multi-Cable MPO-Different Polarities-Cable 2")]
[DataRow(new object[]{"ProcessDetectionData_ReturnMapping_TestCases\\36 Fiber-Different Cable Lengths-Cable
1-AnalysisData", "ProcessDetectionData_ReturnMapping_TestCases\\36 Fiber-Different Cable Lengths-Cable
1.csv"}, DisplayName = "36 Fiber Multi-Cable MPO-Different Cable Lengths-Cable 1")]
[DataRow(new object[]{"ProcessDetectionData_ReturnMapping_TestCases\\36 Fiber-Different Cable Lengths-Cable
2-AnalysisData", "ProcessDetectionData_ReturnMapping_TestCases\\36 Fiber-Different Cable Lengths-Cable
2.csv"}, DisplayName = "36 Fiber Multi-Cable MPO-Different Cable Lengths-Cable 2")]
[DataRow(new object[]{"ProcessDetectionData_ReturnMapping_TestCases\\36 Fiber-Different Cable Lengths-Cable
3-AnalysisData", "ProcessDetectionData_ReturnMapping_TestCases\\36 Fiber-Different Cable Lengths-Cable
3.csv"}, DisplayName = "36 Fiber Multi-Cable MPO-Different Cable Lengths-Cable 3")]
[DataRow(new object[]{"ProcessDetectionData_ReturnMapping_TestCases\\Sparse 24 Fiber-Cable 1-AnalysisData",
"ProcessDetectionData_ReturnMapping_TestCases\\Sparse 24 Fiber-Cable 1.csv"}, DisplayName = "Sparse 24 Fiber
Multi-Cable MPO-Cable 1")]
[DataRow(new object[]{"ProcessDetectionData_ReturnMapping_TestCases\\Sparse 24 Fiber-Cable 2-AnalysisData",
"ProcessDetectionData_ReturnMapping_TestCases\\Sparse 24 Fiber-Cable 2.csv"}, DisplayName = "Sparse 24 Fiber
Multi-Cable MPO-Cable 2")]
[DataRow(new object[]{"ProcessDetectionData_ReturnMapping_TestCases\\24 Fiber-End Offset-AnalysisData",
"ProcessDetectionData_ReturnMapping_TestCases\\24 Fiber-End Offset.csv"}, DisplayName = "24 Fiber MPO-End
Offset")]
[DataRow(new object[]{"ProcessDetectionData_ReturnMapping_TestCases\\12A Ribbon-AnalysisData",
"ProcessDetectionData_ReturnMapping_TestCases\\12A Ribbon.csv"}, DisplayName = "12A Ribbon")]
[DataRow(new object[]{"ProcessDetectionData_ReturnMapping_TestCases\\12B Ribbon-AnalysisData",
"ProcessDetectionData_ReturnMapping_TestCases\\12B Ribbon.csv"}, DisplayName = "12B Ribbon")]
[DataRow(new object[]{"ProcessDetectionData_ReturnMapping_TestCases\\12C Ribbon-AnalysisData",
"ProcessDetectionData_ReturnMapping_TestCases\\12C Ribbon.csv"}, DisplayName = "12C Ribbon")]
[DataRow(new object[]{"ProcessDetectionData_ReturnMapping_TestCases\\Sparse 12 Fiber Ribbon-AnalysisData",
"ProcessDetectionData_ReturnMapping_TestCases\\Sparse 12 Fiber Ribbon.csv"}, DisplayName = "Sparse 12 Fiber
Ribbon")]
[DataRow(new object[]{"ProcessDetectionData_ReturnMapping_TestCases\\12 Fiber-Same Input-AnalysisData",
"ProcessDetectionData_ReturnMapping_TestCases\\12 Fiber-Same Input.csv"}, DisplayName = "12 Fiber Ribbon-
Same Input")]
[DataRow(new object[]{"ProcessDetectionData_ReturnMapping_TestCases\\12 Fiber-Coupled Light-AnalysisData",
"ProcessDetectionData_ReturnMapping_TestCases\\12 Fiber-Coupled Light.csv"}, DisplayName = "12 Fiber Ribbon-
Coupled Light")]
[DataRow(new object[]{"ProcessDetectionData_ReturnMapping_TestCases\\12 Fiber-Uneven Spacing-AnalysisData",
"ProcessDetectionData_ReturnMapping_TestCases\\12 Fiber-Uneven Spacing.csv"}, DisplayName = "12 Fiber
Ribbon-Uneven Spacing")]
[DataRow(new object[]{"ProcessDetectionData_ReturnMapping_TestCases\\12 Fiber-Uneven Spacing-2-
AnalysisData", "ProcessDetectionData_ReturnMapping_TestCases\\12 Fiber-Uneven Spacing-2.csv"}, DisplayName =
"12 Fiber Ribbon-Uneven Spacing 2")]
[DataRow(new object[]{"ProcessDetectionData_ReturnMapping_TestCases\\12 Fiber-Vertical Spread-AnalysisData",
"ProcessDetectionData_ReturnMapping_TestCases\\12 Fiber-Vertical Spread.csv"}, DisplayName = "12 Fiber
Ribbon-Vertical Spread")]
[DataRow(new object[]{"ProcessDetectionData_ReturnMapping_TestCases\\Duplex-1-AnalysisData",
"ProcessDetectionData_ReturnMapping_TestCases\\Duplex-1.csv"}, DisplayName = "Duplex LC-1")]
[DataRow(new object[]{"ProcessDetectionData_ReturnMapping_TestCases\\Duplex-2-AnalysisData",
```

```
"ProcessDetectionData_ReturnMapping_TestCases\\Duplex-2.csv"}, DisplayName = "Duplex LC-2")]
[DataRow(new object[]{"ProcessDetectionData_ReturnMapping_TestCases\\Duplex-Different from Expected-
AnalysisData", "ProcessDetectionData_ReturnMapping_TestCases\\Duplex-Different from Expected.csv"},
DisplayName = "Duplex LC-Different from Expected")]
[DataRow(new object[]{"ProcessDetectionData_ReturnMapping_TestCases\\Duplex-Vertical Spread-AnalysisData",
"ProcessDetectionData_ReturnMapping_TestCases\\Duplex-Vertical Spread.csv"}, DisplayName = "Duplex LC-
Vertical Spread")]
[DataRow(new object[]{"ProcessDetectionData_ReturnMapping_TestCases\\3 Fiber-Multi Row-AnalysisData",
"ProcessDetectionData_ReturnMapping_TestCases\\3 Fiber-Multi Row.csv"}, DisplayName = "3 Fiber MPO-Multi
Row")]
[DataRow(new object[]{"ProcessDetectionData_ReturnMapping_TestCases\\24 Fiber-Vertical-AnalysisData",
"ProcessDetectionData_ReturnMapping_TestCases\\24 Fiber-Vertical.csv"}, DisplayName = "24 Fiber MPO-
Vertical")]
public void ProcessDetectionData_ReturnMapping(string analysisDataPath, string
expectedMappingOperationMappingPath)
```

Parameters

TYPE	NAME	DESCRIPTION
String	analysisDataPath	
String	expectedMappingOperationMappingPath	

ProcessDetectionData_ThrowPolarityMappingFailedException(String)

Declaration

```
[DataTestMethod]
[DataRow(new object[]{"ProcessDetectionData_ThrowPolarityMappingFailedException_TestCases\\12 Fiber-A
Expected-B with 1 Dark Fiber-AnalysisData"}, DisplayName = "12 Fiber MPO-A Expected-B with 1 Dark Fiber")]
[DataRow(new object[]{"ProcessDetectionData_ThrowPolarityMappingFailedException_TestCases\\Sparse 24 Fiber-1
Fiber Short of Mapping-AnalysisData"}, DisplayName = "Sparse 24 Fiber MPO-1 Fiber Short of Mapping")]
[DataRow(new object[]{"ProcessDetectionData_ThrowPolarityMappingFailedException_TestCases\\Sparse 12 Fiber
Ribbon-1 Fiber Short of Mapping-AnalysisData"}, DisplayName = "Sparse 12 Fiber Ribbon-1 Fiber Short of
Mapping")]
[DataRow(new object[]{"ProcessDetectionData_ThrowPolarityMappingFailedException_TestCases\\Duplex-Missing
Fiber-AnalysisData"}, DisplayName = "Duplex LC-Missing Fiber")]
public void ProcessDetectionData_ThrowPolarityMappingFailedException(string analysisDataPath)
```

Parameters

TYPE	NAME	DESCRIPTION
String	analysisDataPath	

Class SimulatedPowerMeterTests

Inheritance

Object

SimulatedPowerMeterTests

Inherited Members

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

Namespace: Hardware.Tests

Assembly: Hardware.Tests.dll

Syntax

```
[TestClass]
public class SimulatedPowerMeterTests
```

Methods

CheckILRefValuesForAllSimulatedBRMDetectors()

Declaration

```
[TestMethod]
public async Task CheckILRefValuesForAllSimulatedBRMDetectors()
```

Returns

TYPE	DESCRIPTION
Task	

CheckILRefValuesForAllSimulatedPLMDetectors()

Declaration

```
[TestMethod]
public async Task CheckILRefValuesForAllSimulatedPLMDetectors()
```

Returns

TYPE	DESCRIPTION
Task	

CheckILRefValuesForAllSimulatedRLMDetectors()

Declaration

```
[TestMethod]
public async Task CheckILRefValuesForAllSimulatedRLMDetectors()
```

Returns

TYPE	DESCRIPTION
Task	

Namespace Santec

Classes

Version

Represents a Santec version number.

Class Version

Represents a Santec version number.

Inheritance

Object
Version

Implements

IComparable<Version>
IEquatable<Version>

Inherited Members

Object.Equals(Object, Object)
Object.ReferenceEquals(Object, Object)
Object.GetType()
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
Int32	major	The major version number.
Int32	minor	The minor version number.
Int32	revision	The revision number.

Exceptions

TYPE	CONDITION
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
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
Int32	The major version number.

Minor

Get the minor version number.

Declaration

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

Property Value

TYPE	DESCRIPTION
Int32	The minor version number.

Revision

Get the revision number.

Declaration

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

Property Value

TYPE	DESCRIPTION
Int32	The revision number.

Methods

CompareTo(Version)

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
Version	other	

Returns

TYPE	DESCRIPTION
Int32	

Equals(Version)

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
Version	other	

Returns

TYPE	DESCRIPTION
Boolean	

Equals(Object)

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
Object	o	

Returns

TYPE	DESCRIPTION
Boolean	

Overrides

Object.Equals(Object)

GetHashCode()

Declaration

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

Returns

TYPE	DESCRIPTION
Int32	

Overrides

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
String	input	The string representation of the version number.

Returns

TYPE	DESCRIPTION
Version	A version number of the string.

Exceptions

TYPE	CONDITION
ArgumentNullException	Thrown if the input is `null`.
ArgumentException	Thrown if the input does not have three components.
ArgumentException	Thrown if one or more input components is not an integer.
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
String	The string representation of the version number.

Overrides

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

Returns

TYPE	DESCRIPTION
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
Boolean	

GreaterThan(Version, Version)

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
Version	operand1	
Version	operand2	

Returns

TYPE	DESCRIPTION
Boolean	

GreaterThanOrEqual(Version, Version)

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
Version	operand1	
Version	operand2	

Returns

TYPE	DESCRIPTION
Boolean	

Inequality(Version, Version)

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
Version	operand1	
Version	operand2	

Returns

TYPE	DESCRIPTION
Boolean	

LessThan(Version, Version)

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
Version	operand1	
Version	operand2	

Returns

TYPE	DESCRIPTION
Boolean	

LessThanOrEqual(Version, Version)

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
Version	operand1	
Version	operand2	

Returns

TYPE	DESCRIPTION
Boolean	

Implements

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

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

Object

FiberTypeToStringConverter

Inherited Members

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

Convert the fiber type.

Declaration

```
public string Convert(EFiberType fiberType, bool capitalize)
```

Parameters

TYPE	NAME	DESCRIPTION
EFiberType	fiberType	The fiber type to convert.

TYPE	NAME	DESCRIPTION
Boolean	capitalize	Whether to capitalize the string.

Returns

TYPE	DESCRIPTION
String	The string representation of the fiber type.

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.

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

Object
PhysicalDevice
SantecDevice
BRM

Implements

IBRM
ISantecDevice
IDiscretePowerMeter
IPowerMeter
IBackreflectionMeter
IDetectorDeviceWithDarking
IDetectorDevice
ILaserSource
IWavelengthDevice
ISwitchController
IDevice
IDisposable
IFiberDevice

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.Dispose()
PhysicalDevice.PingAsync()
PhysicalDevice.ResetAsync()

PhysicalDevice.QueryAsync(String, CancellationToken)
PhysicalDevice.WriteAsync(String, CancellationToken)
PhysicalDevice.ReadAsync(CancellationToken)
Object.ToString()
Object.Equals(Object)
Object.Equals(Object, Object)
Object.ReferenceEquals(Object, Object)
Object.GetHashCode()
Object.GetType()
Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.Backreflection

Assembly: Santec.Hardware.dll

Syntax

```
public class BRM : SantecDevice, IBRM, ISantecDevice, IDiscretePowerMeter, IPowerMeter,
IBackreflectionMeter, IDetectorDeviceWithDarking, IDetectorDevice, ILaserSource, IWavelengthDevice,
ISwitchController, IDevice, IDisposable, IFiberDevice
```

Properties

Detectors

Get the detectors connected to the device.

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

Declaration

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

Property Value

TYPE	DESCRIPTION
IReadOnlyList<DeviceDetector>	The list of the detectors connected to the device.

Exceptions

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

Fiber

Get the fiber information of the device.

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

Declaration

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

Property Value

TYPE	DESCRIPTION
FiberInformation	The fiber information of the device.

Exceptions

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

Name

Get the name of the device.

Declaration

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

Property Value

TYPE	DESCRIPTION
String	The name of the device.

Overrides

PhysicalDevice.Name

Remarks

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

NumberOfDetectors

Get the number of detectors connected to the device.

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

Declaration

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

Property Value

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

Exceptions

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

NumberOfOutputs

Get the number of laser outputs.

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

Declaration

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

Property Value

TYPE	DESCRIPTION
Int32	The number of device outputs.

Exceptions

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

Switches

Get the switches connected to the device.

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

Declaration

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

Property Value

TYPE	DESCRIPTION

TYPE	DESCRIPTION
List<DeviceSwitch>	The list of the switches connected to the device.

Exceptions

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

Wavelengths

Get the wavelengths supported by the device.

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

Declaration

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

Property Value

TYPE	DESCRIPTION
List<Int32>	A list of the device's wavelengths.

Exceptions

TYPE	CONDITION
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

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

Parameters

TYPE	NAME	DESCRIPTION

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

Returns

TYPE	DESCRIPTION
Task	The task representing the operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device has no switch at any of the specified switch indexes.
ArgumentException	Thrown when any the specified channels is out of range for the corresponding device switch.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

ChangeOutputChannelAsync(Int32, CancellationToken)

Asynchronously change the output channel of the laser source.

Declaration

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

Parameters

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

Returns

TYPE	DESCRIPTION
Task	The task representing the operation.

Remarks

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

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified output channel is out of range for the laser source.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the RLM.

ChangeSwitchChannelAsync(Int32, Int32, CancellationToken)

Asynchronously change the channel of a connected switch.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
Int32	switchIndex	The switch number of the switch. `0` for the internal switch; `1` or `2` for external switches attached to the device.

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

Returns

TYPE	DESCRIPTION
Task	The task representing the operation.

Remarks

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

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device has no switch at the specified switch index.
ArgumentException	Thrown when the specified channel is out of range for the device switch.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

DarkAllDetectorsAsync(CancellationToken)

Asynchronously dark all of the detectors that support darking.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION

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

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

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

Exceptions

TYPE	CONDITION
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.
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.

DarkBRDetectorAsync(CancellationToken)

Asynchronously dark the BR detector.

Declaration

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

Parameters

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

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

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

Exceptions

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

DarkDetectorAsync(Int32, CancellationToken)

Asynchronously dark the specified detector.

Declaration

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

Parameters

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

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

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

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified detector is out of range for the device.
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.
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.

GetActiveLaserAsync(CancellationToken)

Asynchronously get the active laser.

Declaration

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

Parameters

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

Returns

TYPE	DESCRIPTION
Task<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

TYPE	CONDITION
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.
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

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

Parameters

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

Returns

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

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
InvalidOperationException	Thrown when the method is called on an uninitialized device.

TYPE	CONDITION
UnexpectedDeviceResponseException	Thrown when the BR reference query response is not a valid number.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

GetDetectorDarkAsync(Int32, CancellationToken)

Asynchronously get whether the specified detector has a dark value.

Declaration

```
public Task<bool> GetDetectorDarkAsync(int detector, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	The detector to determine if it has a dark value.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

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

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified detector is out of range for the detector.
InvalidOperationException	Thrown when the method is called on an uninitialized detector.
UnexpectedDeviceResponseException	Thrown when the device detector dark query response is not a valid boolean value.

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

GetFactoryPowerAsync(Int32, Int32, CancellationToken)

Asynchronously get the factory power for the specified output channel and wavelength.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
Int32	output	The output channel to get the factory power for.
Int32	wavelength	The wavelength to get the factory power for.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

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

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified output channel is out of range.
ArgumentException	Thrown when the device does not support the specified wavelength.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device factory power response is not a valid number.

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

GetILCompensationEnabledAsync(CancellationToken)

Asynchronously get whether IL compensation is enabled.

Declaration

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

Parameters

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

Returns

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

Remarks

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

Exceptions

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

GetILReferenceAsync(Int32, Int32, CancellationToken)

Asynchronously get the stored IL reference for a detector at a given wavelength.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	The detector to get the reference for.
Int32	wavelength	The wavelength to get the reference for.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

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

Remarks

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

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
ArgumentException	Thrown when the specified detector is out of range for the device.
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.
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

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

Parameters

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

Returns

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

Remarks

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

Exceptions

TYPE	CONDITION
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.
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

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

Parameters

TYPE	NAME	DESCRIPTION
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.
CancellationToken	cancellationToken	

Returns

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

Remarks

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

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device has no switch at the specified switch index.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

InitializeAsync()

Asynchronously open the device connection and initialize the device settings.

Declaration

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

Returns

TYPE	DESCRIPTION
Task	The task representing the operation.

Overrides

SantecDevice.InitializeAsync()

Remarks

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

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.
TimeoutException	Thrown when a communication failure causes no response to be received from the BRM.

ReadBRAsync(Int32, CancellationToken)

Asynchronously read the backreflection at a given wavelength.

Declaration

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

Parameters

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

Returns

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

Remarks

NOTE

This operation will not block.

Exceptions

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

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device read BR response is not a valid number.
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

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

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	The detector number of the detector to read the insertion loss from.
Int32	wavelength	The wavelength to read the insertion loss at.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

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

Remarks

NOTE

This operation will not block.

Exceptions

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

TYPE	CONDITION
ArgumentException	Thrown when the specified detector is out of range for the device.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device read IL response is not a valid number.
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

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

Parameters

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

Returns

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

Remarks

NOTE

This operation will not block.

Exceptions

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

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

ReadMultipleILsAsync(IEnumerable<Int32>, Int32, CancellationToken)

Asynchronously read the insertion loss from multiple detectors at a given wavelength.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
IEnumerable<Int32>	detectors	The detectors to read the insertion loss from.
Int32	wavelength	The wavelength to read the insertion loss at.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<List<Double>>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

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

TYPE	CONDITION
ArgumentException	Thrown when any of the specified detectors are out of range for the device.
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.
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

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

Parameters

TYPE	NAME	DESCRIPTION
IEnumerable<Int32>	detectors	The detectors to read from.
Int32	wavelength	The wavelength to read the power at.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<List<Double>>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
ArgumentException	Thrown when any of the specified detectors are out of range for the device.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

ReadPowerAsync(Int32, Int32, CancellationToken)

Asynchronously read the power from an device detector at a given wavelength.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	The detector number of the detector to read from.
Int32	wavelength	The wavelength to read the power at.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

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

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
ArgumentException	Thrown when the specified detector is out of range for the device.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device read power response is not a valid number.
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

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

Parameters

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

Returns

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

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device reference BR response is not a valid number.
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

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

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	The detector number of the detector to perform the reference for.
Int32	wavelength	The wavelength to perform the reference at.
Boolean	applyReferenceToAllDetectors	Indicate if the reference read should be applied to all detectors for the current channel.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

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

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
ArgumentException	Thrown when the specified detector is out of range for the device.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device reference IL response is not a valid number.
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

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

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	The detector number of the detector to perform the reference for.
Int32	wavelength	The wavelength to perform the reference at.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

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

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
ArgumentException	Thrown when the specified detector is out of range for the device.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device reference IL response is not a valid number.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

RefreshAsync()

Asynchronously refresh the device settings.

Declaration

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

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Overrides

PhysicalDevice.RefreshAsync()

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
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.

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

SetBRReferenceAsync(Int32, Double, CancellationToken)

Asynchronously set the BR zero reference at a given wavelength.

Declaration

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

Parameters

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

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
InvalidOperationException	Thrown when the method is called on an uninitialized device.

TYPE	CONDITION
UnexpectedDeviceResponseException	Thrown when the BR reference query response does not match the expected response.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

SetILCompensationEnabledAsync(Boolean, CancellationToken)

Asynchronously set whether IL compensation is enabled.

Declaration

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

Parameters

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

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

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

SetILReferenceAsync(Int32, Int32, Double, CancellationToken)

Asynchronously set the IL reference for a detector at a given wavelength.

Declaration

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

Parameters

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

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
ArgumentException	Thrown when the specified detector is out of range for the device.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

TurnOffLaserAsync(CancellationToken)

Asynchronously turn off the active laser.

Declaration

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

Parameters

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

Returns

TYPE	DESCRIPTION
Task	The task representing the operation.

Remarks

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

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device laser response does not match the expected response.
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

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

Parameters

TYPE	NAME	DESCRIPTION
Int32	wavelength	The wavelength of the laser to turn on.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

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

Exceptions

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

Uninitialize()

Synchronously uninitialize device settings and close the connection.

Declaration

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

Overrides

PhysicalDevice.Uninitialize()

Implements

IBRM
ISantecDevice
IDiscretePowerMeter
IPowerMeter
IBackreflectionMeter
IDetectorDeviceWithDarking
IDetectorDevice
ILaserSource
IWavelengthDevice
ISwitchController
IDevice
System.IDisposable
IFiberDevice

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.GetDetectorDarkAsync(Int32, CancellationToken)
IDetectorDevice.NumberOfDetectors
IDetectorDevice.Detectors
IDevice.Name
IDevice.SerialNumber
IDevice.FirmwareVersion
IDevice.Address
IDevice.IsInitialized
IDevice.InitializeAsync()
IDevice.Uninitialize()
IDevice.ResetAsync()
IDevice.RefreshAsync()
IDevice.PingAsync()
IDevice.QueryAsync(String, CancellationToken)
IDevice.WriteAsync(String, CancellationToken)
IDevice.ReadAsync(CancellationToken)
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
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

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

Exceptions

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

Returns

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

Remarks

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

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the BR reference query response is not a valid number.
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<double> ReadBRAsync(int wavelength, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

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

Returns

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

Remarks

NOTE

This operation will not block.

Exceptions

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

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device read BR response is not a valid number.
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
Int32	wavelength	The wavelength to perform the BR zero reference at.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

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

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
InvalidOperationException	Thrown when the method is called on an uninitialized device.

TYPE	CONDITION
UnexpectedDeviceResponseException	Thrown when the device reference BR response is not a valid number.
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
Int32	wavelength	The wavelength to set the BR zero for.
Double	reference	The BR zero value.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
InvalidOperationException	Thrown when the method is called on an uninitialized device.

TYPE	CONDITION
UnexpectedDeviceResponseException	Thrown when the BR reference query response does not match the expected response.
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)
IPowerMeter.ReadPowerAsync(Int32, Int32, CancellationToken)
IPowerMeter.GetILReferenceAsync(Int32, Int32, CancellationToken)
IPowerMeter.SetILReferenceAsync(Int32, Int32, Double, CancellationToken)
IPowerMeter.ReferenceILAsync(Int32, Int32, CancellationToken)
IPowerMeter.ReferenceILAsync(Int32, Int32, Boolean, CancellationToken)
IPowerMeter.ReadILAsync(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.GetDetectorDarkAsync(Int32, CancellationToken)
IDetectorDevice.NumberOfDetectors
IDetectorDevice.Detectors
ILaserSource.NumberOfOutputs
ILaserSource.GetActiveLaserAsync(CancellationToken)
ILaserSource.TurnOnLaserAsync(Int32, CancellationToken)
ILaserSource.TurnOffLaserAsync(CancellationToken)
ILaserSource.GetOutputChannelAsync(CancellationToken)
ILaserSource.ChangeOutputChannelAsync(Int32, CancellationToken)
ILaserSource.ReadMonitorPowerAsync(Int32, CancellationToken)
ILaserSource.GetFactoryPowerAsync(Int32, Int32, CancellationToken)
IWavelengthDevice.Wavelengths
ISwitchController.Switches
ISwitchController.GetSwitchChannelAsync(Int32, CancellationToken)
ISwitchController.ChangeSwitchChannelAsync(Int32, Int32, CancellationToken)
ISwitchController.ChangeMultipleSwitchesChannelsAsync(IEnumerable<DeviceSwitchAndChannelPair>, CancellationToken)
IDevice.Name
IDevice.SerialNumber
IDevice.FirmwareVersion
IDevice.Address
IDevice.IsInitialized

IDevice.InitializeAsync()
IDevice.Uninitialize()
IDevice.ResetAsync()
IDevice.RefreshAsync()
IDevice.PingAsync()
IDevice.QueryAsync(String, CancellationToken)
IDevice.WriteAsync(String, CancellationToken)
IDevice.ReadAsync(CancellationToken)
IDisposable.Dispose()
IFiberDevice.Fiber

Namespace: Santec.Hardware.Devices.Backreflection

Assembly: Santec.Hardware.dll

Syntax

```
public interface IBRM : ISantecDevice, IDiscretePowerMeter, IPowerMeter, IBackreflectionMeter,
IDetectorDeviceWithDarking, IDetectorDevice, ILaserSource, IWavelengthDevice, ISwitchController, IDevice,
IDisposable, IFiberDevice
```

Methods

GetILCompensationEnabledAsync(CancellationToken)

Asynchronously get whether IL compensation is enabled.

Declaration

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

Parameters

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

Returns

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

Remarks

NOTE

This operation will not block.

Exceptions

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

TYPE	CONDITION
UnexpectedDeviceResponseException	Thrown when the BRM IL compensation enabled response is not a valid boolean value.
TimeoutException	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
Boolean	enabled	Whether to enable IL compensation or not.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
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.
TimeoutException	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

Object
SimulatedSantecDevice
SimulatedBRM

Implements

IBRM
ISantecDevice
IDiscretePowerMeter
ISwitchController
IFiberDevice
IBackreflectionMeter
IDetectorDeviceWithDarking
IPowerMeter
IDetectorDevice
ILaserSource
IWavelengthDevice
IDevice
IDisposable

Inherited Members

SimulatedSantecDevice.creatingDevice
SimulatedSantecDevice.Dispose()
SimulatedSantecDevice.SerialNumber
SimulatedSantecDevice.FirmwareVersion
SimulatedSantecDevice.Address
SimulatedSantecDevice.IsInitialized
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)
Object.ToString()
Object.Equals(Object)
Object.Equals(Object, Object)
Object.ReferenceEquals(Object, Object)
Object.GetHashCode()
Object.GetType()
Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.Backreflection

Assembly: Santec.Hardware.dll

Syntax

```
public class SimulatedBRM : SimulatedSantecDevice, IBRM, ISantecDevice, IDiscretePowerMeter,
ISwitchController, IFiberDevice, IBackreflectionMeter, IDetectorDeviceWithDarking, IPowerMeter,
IDetectorDevice, ILaserSource, IWavelengthDevice, IDevice, IDisposable
```

Properties

BRMax

Declaration

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

Property Value

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

BRMin

Declaration

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

Property Value

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

BRReferenceMax

Declaration

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

Property Value

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

BRReferenceMin

Declaration

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

Property Value

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

Detectors

Get the detectors connected to the device.

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

Declaration

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

Property Value

TYPE	DESCRIPTION
IReadOnlyList<DeviceDetector>	The list of the detectors connected to the device.

Exceptions

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

Fiber

Get the fiber information of the device.

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

Declaration

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

Property Value

TYPE	DESCRIPTION
FiberInformation	The fiber information of the device.

Exceptions

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

ILMax

Declaration

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

Property Value

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

ILMin

Declaration

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

Property Value

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

ILReferenceMax

Declaration

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

Property Value

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

ILReferenceMin

Declaration

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

Property Value

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

MonitorPowerMax

Declaration

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

Property Value

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

MonitorPowerMin

Declaration

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

Property Value

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

Name

Get the name of the device.

Declaration

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

Property Value

TYPE	DESCRIPTION
String	The name of the device.

Overrides

SimulatedSantecDevice.Name

Remarks

NOTE

This property will always return "BRM".

NumberOfDetectors

Get the number of detectors connected to the device.

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

Declaration

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

Property Value

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

Exceptions

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

NumberOfOutputs

Get the number of laser outputs.

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

Declaration

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

Property Value

TYPE	DESCRIPTION
Int32	The number of device outputs.

Exceptions

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

PowerMax

Declaration

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

Property Value

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

PowerMin

Declaration

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

Property Value

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

Switches

Get the switches connected to the device.

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

Declaration

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

Property Value

TYPE	DESCRIPTION
List<DeviceSwitch>	The list of the switches connected to the device.

Exceptions

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

Wavelengths

Get the wavelengths supported by the device.

> **NOTE**

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

Declaration

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

Property Value

TYPE	DESCRIPTION
List<Int32>	A list of the device's wavelengths.

Exceptions

TYPE	CONDITION
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

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

Parameters

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

Returns

TYPE	DESCRIPTION
Task	The task representing the operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device has no switch at any of the specified switch indexes.
ArgumentException	Thrown when any the specified channels is out of range for the corresponding device switch.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

ChangeOutputChannelAsync(Int32, CancellationToken)

Asynchronously change the output channel of the laser source.

Declaration

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

Parameters

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

Returns

TYPE	DESCRIPTION
Task	The task representing the operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified output channel is out of range for the laser source.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the RLM.

ChangeSwitchChannelAsync(Int32, Int32, CancellationToken)

Asynchronously change the channel of a connected switch.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
Int32	switchIndex	The switch number of the switch. `0` for the internal switch; `1` or `2` for external switches attached to the device.
Int32	channel	The channel to switch to.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION

TYPE	CONDITION
ArgumentException	Thrown when the device has no switch at the specified switch index.
ArgumentException	Thrown when the specified channel is out of range for the device switch.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

ConfigureBR(Double, Double)

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
Double	brMin	
Double	brMax	

ConfigureBRReference(Double, Double)

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
Double	brReferenceMin	
Double	brReferenceMax	

ConfigureIL(Double, Double)

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
Double	ilMin	
Double	ilMax	

ConfigureILReference(Double, Double)

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
Double	ilReferenceMin	
Double	ilReferenceMax	

ConfigureMonitorPower(Double, Double)

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
Double	monitorPowerMin	
Double	monitorPowerMax	

ConfigurePower(Double, Double)

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
Double	powerMin	
Double	powerMax	

DarkAllDetectorsAsync(CancellationToken)

Asynchronously dark all of the detectors that support darking.

Declaration

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

Parameters

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

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

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

Exceptions

TYPE	CONDITION
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.
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.

DarkBRDetectorAsync(CancellationToken)

Asynchronously dark the BR detector.

Declaration

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

Parameters

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

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

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

Exceptions

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

DarkDetectorAsync(Int32, CancellationToken)

Asynchronously dark the specified detector.

Declaration

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

Parameters

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

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

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

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified detector is out of range for the device.
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.
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.

GetActiveLaserAsync(CancellationToken)

Asynchronously get the active laser.

Declaration

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

Parameters

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

Returns

TYPE	DESCRIPTION
Task<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

TYPE	CONDITION
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.
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

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

Parameters

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

Returns

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

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
InvalidOperationException	Thrown when the method is called on an uninitialized device.

TYPE	CONDITION
UnexpectedDeviceResponseException	Thrown when the BR reference query response is not a valid number.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

GetDetectorDarkAsync(Int32, CancellationToken)

Asynchronously get whether the specified detector has a dark value.

Declaration

```
public Task<bool> GetDetectorDarkAsync(int detector, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	The detector to determine if it has a dark value.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

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

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified detector is out of range for the detector.
InvalidOperationException	Thrown when the method is called on an uninitialized detector.
UnexpectedDeviceResponseException	Thrown when the device detector dark query response is not a valid boolean value.

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

GetFactoryPowerAsync(Int32, Int32, CancellationToken)

Asynchronously get the factory power for the specified output channel and wavelength.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
Int32	output	The output channel to get the factory power for.
Int32	wavelength	The wavelength to get the factory power for.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

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

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified output channel is out of range.
ArgumentException	Thrown when the device does not support the specified wavelength.
InvalidOperationException	Thrown when the method is called on an uninitialized device.

TYPE	CONDITION
UnexpectedDeviceResponseException	Thrown when the device factory power response is not a valid number.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

GetILCompensationEnabledAsync(CancellationToken)

Asynchronously get whether IL compensation is enabled.

Declaration

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

Parameters

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

Returns

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

Remarks

NOTE

This operation will not block.

Exceptions

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

GetILReferenceAsync(Int32, Int32, CancellationToken)

Asynchronously get the stored IL reference for a detector at a given wavelength.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	The detector to get the reference for.
Int32	wavelength	The wavelength to get the reference for.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

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

Remarks

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

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
ArgumentException	Thrown when the specified detector is out of range for the device.
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.
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

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

Parameters

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

Returns

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

Remarks

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

Exceptions

TYPE	CONDITION
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.
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

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

Parameters

TYPE	NAME	DESCRIPTION
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.
CancellationToken	cancellationToken	

Returns

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

Remarks

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

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device has no switch at the specified switch index.
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.
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

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

Parameters

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

Returns

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

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device read BR response is not a valid number.
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

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

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	The detector number of the detector to read the insertion loss from.
Int32	wavelength	The wavelength to read the insertion loss at.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

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

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
ArgumentException	Thrown when the specified detector is out of range for the device.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device read IL response is not a valid number.
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

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

Parameters

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

Returns

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

Remarks

NOTE

This operation will not block.

Exceptions

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

ReadMultipleILsAsync(IEnumerable<Int32>, Int32, CancellationToken)

Asynchronously read the insertion loss from multiple detectors at a given wavelength.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
IEnumerable<Int32>	detectors	The detectors to read the insertion loss from.
Int32	wavelength	The wavelength to read the insertion loss at.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<List<Double>>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
ArgumentException	Thrown when any of the specified detectors are out of range for the device.
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.
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

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

Parameters

TYPE	NAME	DESCRIPTION
IEnumerable<Int32>	detectors	The detectors to read from.
Int32	wavelength	The wavelength to read the power at.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<List<Double>>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
ArgumentException	Thrown when any of the specified detectors are out of range for the device.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

ReadPowerAsync(Int32, Int32, CancellationToken)

Asynchronously read the power from an device detector at a given wavelength.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	The detector number of the detector to read from.
Int32	wavelength	The wavelength to read the power at.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

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

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
ArgumentException	Thrown when the specified detector is out of range for the device.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device read power response is not a valid number.
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

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

Parameters

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

Returns

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

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device reference BR response is not a valid number.
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

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

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	The detector number of the detector to perform the reference for.
Int32	wavelength	The wavelength to perform the reference at.
Boolean	applyReferenceToAllDetectors	Indicate if the reference read should be applied to all detectors for the current channel.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

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

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
ArgumentException	Thrown when the specified detector is out of range for the device.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device reference IL response is not a valid number.
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

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

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	The detector number of the detector to perform the reference for.
Int32	wavelength	The wavelength to perform the reference at.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

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

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
ArgumentException	Thrown when the specified detector is out of range for the device.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device reference IL response is not a valid number.
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

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

Parameters

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

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

SetILCompensationEnabledAsync(Boolean, CancellationToken)

Asynchronously set whether IL compensation is enabled.

Declaration

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

Parameters

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

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

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

TYPE	CONDITION
UnexpectedDeviceResponseException	Thrown when the BRM IL compensation enabled response does not match the expected response.
TimeoutException	Thrown when a communication failure causes no response to be received from the BRM.

SetILReferenceAsync(Int32, Int32, Double, CancellationToken)

Asynchronously set the IL reference for a detector at a given wavelength.

Declaration

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

Parameters

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

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
ArgumentException	Thrown when the specified detector is out of range for the device.
InvalidOperationException	Thrown when the method is called on an uninitialized device.

TYPE	CONDITION
UnexpectedDeviceResponseException	Thrown when the device IL reference query response does not match the expected response.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

TurnOffLaserAsync(CancellationToken)

Asynchronously turn off the active laser.

Declaration

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

Parameters

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

Returns

TYPE	DESCRIPTION
Task	The task representing the operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device laser response does not match the expected response.
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

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

Parameters

TYPE	NAME	DESCRIPTION
Int32	wavelength	The wavelength of the laser to turn on.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

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

Exceptions

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

Implements

IBRM
ISantecDevice
IDiscretePowerMeter
ISwitchController
IFiberDevice
IBackreflectionMeter
IDetectorDeviceWithDarking
IPowerMeter
IDetectorDevice
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 NotSupportedException.

DeviceDetector

Represents a detector of a IDetectorDevice.

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.

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.

IFiberDevice

Defines the properties of a fiber device.

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.

ISantecDevice

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

IWavelengthDevice

Defines the properties of a discrete wavelength device.

Enums

ECoreSize

Specifies the core size of a 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.

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 NotSupportedException.

Inheritance

Object
CompositeSantecDevice
RDP
RDSP
SimulatedRDP
SimulatedRDSP

Implements

ISantecDevice
ICompositeDevice
IDevice
IDisposable

Inherited Members

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

Namespace: Santec.Hardware.Devices.General

Assembly: Santec.Hardware.dll

Syntax

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

Properties

Address

Get the connection address of the device.

Declaration

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

Property Value

TYPE	DESCRIPTION
String	The connection address of the device.

FirmwareVersion

Get the firmware version of the device.

Declaration

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

Property Value

TYPE	DESCRIPTION
Version	The firmware version of the device.

IsInitialized

Get whether the device and connection are initialized.

Declaration

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

Property Value

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

IsInitializing

Get whether the device and connection are initializing.

Declaration

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

Property Value

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

Name

Get the name of the device.

Declaration

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

Property Value

TYPE	DESCRIPTION
String	The name of the device.

SerialNumber

Get the serial number of the device.

Declaration

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

Property Value

TYPE	DESCRIPTION
String	The serial number of the device.

Subdevices

Get the subdevices that make up the composite device.

Declaration

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

Property Value

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

Methods

Dispose()

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

Declaration

```
public void Dispose()
```

EnableDHCPAsync(CancellationToken)

⚠ CAUTION

This operation is not supported for composite devices. This operation will always throw a NotSupportedException.

Declaration

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

Parameters

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

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

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

GetGatewayAsync(CancellationToken)

CAUTION

This operation is not supported for composite devices. This operation will always throw a NotSupportedException.

Declaration

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

Parameters

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

Returns

TYPE	DESCRIPTION
Task<IPAddress>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

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

GetHostnameAsync(CancellationToken)

> **⚠ CAUTION**
>
> This operation is not supported for composite devices. This operation will always throw a NotSupportedException.

Declaration

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

Parameters

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

Returns

TYPE	DESCRIPTION
Task<String>	The task representing the asynchronous operation.

Remarks

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

Exceptions

TYPE	CONDITION
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 NotSupportedException.

Declaration

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

Parameters

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

Returns

TYPE	DESCRIPTION
Task<EInteractionMode>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

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

GetIPAddressAsync(CancellationToken)

CAUTION

This operation is not supported for composite devices. This operation will always throw a NotSupportedException.

Declaration

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

Parameters

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

Returns

TYPE	DESCRIPTION
Task<IPAddress>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

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

GetSubnetPrefixLengthAsync(CancellationToken)

⚠ CAUTION

This operation is not supported for composite devices. This operation will always throw a NotSupportedException.

Declaration

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

Parameters

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

Returns

TYPE	DESCRIPTION
Task<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
NotSupportedException	Thrown when the operation is not supported by the device.

InitializeAsync()

Asynchronously open the device connection and initialize the device settings.

Declaration

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

Returns

TYPE	DESCRIPTION
Task	The task representing the operation.

Remarks

NOTE

This operation will not block.

PingAsync()

Declaration

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

Returns

TYPE	DESCRIPTION
Task<Boolean>	

QueryAsync(String, CancellationToken)

⚠ CAUTION

This operation is not supported for composite devices. This operation will always throw a NotSupportedException.

Declaration

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

Parameters

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

Returns

TYPE	DESCRIPTION
Task<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
NotSupportedException	Thrown when the operation is not supported by the device.

ReadAsync(CancellationToken)

> **⚠ CAUTION**
>
> This operation is not supported for composite devices. This operation will always throw a NotSupportedException.

Declaration

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

Parameters

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

Returns

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

Remarks

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

Exceptions

TYPE	CONDITION
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

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

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

Exceptions

TYPE	CONDITION
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.
TimeoutException	Thrown when a communication failure causes no response to be received from a subdevice during its refresh operation.

ResetAsync()

Asynchronously reset the device.

Declaration

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

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

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

SetGatewayAsync(IPAddress, CancellationToken)

> **CAUTION**
>
> This operation is not supported for composite devices. This operation will always throw a NotSupportedException.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
IPAddress	address	The gateway IP address.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

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

SetHostnameAsync(String, CancellationToken)

CAUTION

This operation is not supported for composite devices. This operation will always throw a NotSupportedException.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
String	hostname	The mDNS hostname.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
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 NotSupportedException.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
EInteractionMode	interactionMode	The interaction mode to set the device to.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

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

SetIPAddressAsync(IPAddress, CancellationToken)

> **⚠ CAUTION**
>
> This operation is not supported for composite devices. This operation will always throw a NotSupportedException.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
IPAddress	address	The IP address.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

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

Exceptions

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

SetSubnetPrefixLengthAsync(Int32, CancellationToken)

> **⚠ CAUTION**
>
> This operation is not supported for composite devices. This operation will always throw a NotSupportedException.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
Int32	prefixLength	The subnet prefix length.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

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

Uninitialize()

Synchronously uninitialize device settings and close the connection.

Declaration

```
public virtual void Uninitialize()
```

WriteAsync(String, CancellationToken)

> **CAUTION**
>
> This operation is not supported for composite devices. This operation will always throw a NotSupportedException.

Declaration

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

Parameters

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

Returns

TYPE	DESCRIPTION
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
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

Object
DeviceDetector
OPMDetector

Inherited Members

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

Namespace: Santec.Hardware.Devices.General

Assembly: Santec.Hardware.dll

Syntax

```
public class DeviceDetector
```

Properties

Index

Get the detector index.

Declaration

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

Property Value

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

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

Object
DeviceFactory

Inherited Members

Object.ToString()
Object.Equals(Object)
Object.Equals(Object, Object)
Object.ReferenceEquals(Object, Object)
Object.GetHashCode()
Object.GetType()
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
IPAddress	ipAddress	The IP address of the RLM.
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
IPAddress	ipAddress	The IP address of the RLM.
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.
Unknown	Unknown core size.

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
Multimode	Multimode 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.

Class FiberInformation

Represents information about the fiber of a device

Inheritance

Object

FiberInformation

Inherited Members

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

Namespace: Santec.Hardware.Devices.General

Assembly: Santec.Hardware.dll

Syntax

```
public class FiberInformation
```

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.ResetAsync()
IDevice.RefreshAsync()
IDevice.PingAsync()
IDevice.QueryAsync(String, CancellationToken)
IDevice.WriteAsync(String, CancellationToken)
IDevice.ReadAsync(CancellationToken)
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
IReadOnlyList<IDevice>	A list of the subdevices that make up the composite device.

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.ResetAsync()
IDevice.RefreshAsync()
IDevice.PingAsync()
IDevice.QueryAsync(String, CancellationToken)
IDevice.WriteAsync(String, CancellationToken)
IDevice.ReadAsync(CancellationToken)
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<DeviceDetector> Detectors { get; }
```

Property Value

TYPE	DESCRIPTION
IReadOnlyList<DeviceDetector>	The list of the detectors connected to the device.

Exceptions

TYPE	CONDITION
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
Int32	The number of detectors connected to the device.

Exceptions

TYPE	CONDITION
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.ResetAsync()
IDevice.RefreshAsync()
IDevice.PingAsync()
IDevice.QueryAsync(String, CancellationToken)
IDevice.WriteAsync(String, CancellationToken)
IDevice.ReadAsync(CancellationToken)
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
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

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

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified detector is out of range for the device.
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.
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.

GetDetectorDarkAsync(Int32, CancellationToken)

Asynchronously get whether the specified detector has a dark value.

Declaration

```
Task<bool> GetDetectorDarkAsync(int detector, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	The detector to determine if it has a dark value.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

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

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified detector is out of range for the detector.
InvalidOperationException	Thrown when the method is called on an uninitialized detector.
UnexpectedDeviceResponseException	Thrown when the device detector dark query response is not a valid boolean value.
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

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

Name

Get the name of the device.

Declaration

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

Property Value

TYPE	DESCRIPTION
String	The name of the device.

SerialNumber

Get the serial number of the device.

Declaration

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

Property Value

TYPE	DESCRIPTION
String	The serial number of the device.

Methods

InitializeAsync()

Asynchronously open the device connection and initialize the device settings.

Declaration

```
Task InitializeAsync()
```

Returns

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

Returns

TYPE	DESCRIPTION
Task<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
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
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

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

Remarks

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

Exceptions

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

RefreshAsync()

Asynchronously refresh the device settings.

Declaration

```
Task RefreshAsync()
```

Returns

TYPE	DESCRIPTION
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
Task	The task representing the asynchronous operation.

Remarks

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

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

Returns

TYPE	DESCRIPTION
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
InvalidOperationException	Thrown when the method is called on an uninitialized 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
InvalidOperationException	Thrown when the property is accessed before the device has been initialized.

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.ResetAsync()
IDevice.RefreshAsync()
IDevice.PingAsync()
IDevice.QueryAsync(String, CancellationToken)
IDevice.WriteAsync(String, CancellationToken)
IDevice.ReadAsync(CancellationToken)
IDisposable.Dispose()

Namespace: Santec.Hardware.Devices.General

Assembly: Santec.Hardware.dll

Syntax

```
public interface IModuleController : IDevice, IDisposable
```

Properties

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
Int32	The number of modules in the device.

Exceptions

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

Methods

GetSelectedModuleAsync(CancellationToken)

Asynchronously get the currently selected module.

Declaration

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

Parameters

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

Returns

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

Remarks

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

Exceptions

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

SetSelectedModuleAsync(Int32, CancellationToken)

Asynchronously set the currently selected module.

Declaration

```
Task SetSelectedModuleAsync(int moduleIndex, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION

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

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

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

Exceptions

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

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.ResetAsync()
IDevice.RefreshAsync()
IDevice.PingAsync()
IDevice.QueryAsync(String, CancellationToken)
IDevice.WriteAsync(String, CancellationToken)
IDevice.ReadAsync(CancellationToken)
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
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
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.
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
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<IPAddress>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
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.
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
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<String>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device mDNS hostname response is not a valid hostname.
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
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<EInteractionMode>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device response does not correspond to an interaction mode.
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
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<IPAddress>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
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.
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
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<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
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.

TYPE	CONDITION
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
IPAddress	address	The gateway IP address.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
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.
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
String	hostname	The mDNS hostname.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

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

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified mDNS hostname is null, empty, or whitespace.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the mDNS hostname response does not match the expected response.
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.

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

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

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

Exceptions

TYPE	CONDITION
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.
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
IPAddress	address	The IP address.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

TYPE	DESCRIPTION

Remarks

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

Exceptions

TYPE	CONDITION
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.
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
Int32	prefixLength	The subnet prefix length.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
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
ArgumentException	Thrown when the subnet prefix length is not valid (less than `0` or greater than `31`).
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

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.ResetAsync()
IDevice.RefreshAsync()
IDevice.PingAsync()
IDevice.QueryAsync(String, CancellationToken)
IDevice.WriteAsync(String, CancellationToken)
IDevice.ReadAsync(CancellationToken)
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
List<Int32>	A list of the device's wavelengths.

Exceptions

TYPE	CONDITION
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

Object
PhysicalDevice
SantecDevice

Implements

IDevice
IDisposable

Inherited Members

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

Namespace: Santec.Hardware.Devices.General

Assembly: Santec.Hardware.dll

Syntax

```
public abstract class PhysicalDevice : IDevice, IDisposable
```

Properties

Address

Get the connection address of the device.

Declaration

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

Property Value

TYPE	DESCRIPTION
String	The connection address of the device.

FirmwareVersion

Get the firmware version of the device.

Declaration

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

Property Value

TYPE	DESCRIPTION
Version	The firmware version of the device.

IsInitialized

Get whether the device and connection are initialized.

Declaration

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

Property Value

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

IsInitializing

Get whether the device and connection are initializing.

Declaration

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

Property Value

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

Name

Get the name of the device.

Declaration

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

Property Value

TYPE	DESCRIPTION
String	The name of the device.

SerialNumber

Get the serial number of the device.

Declaration

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

Property Value

TYPE	DESCRIPTION
String	The serial number of the device.

Methods

Dispose()

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

Declaration

```
public void Dispose()
```

InitializeAsync()

Asynchronously open the device connection and initialize the device settings.

Declaration

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

Returns

TYPE	DESCRIPTION
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

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

Returns

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

Remarks

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

QueryAsync(String, CancellationToken)

Asynchronously send a query to the device.

Declaration

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

Parameters

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

Returns

TYPE	DESCRIPTION
Task<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
InvalidOperationException	Thrown when the method is called on an uninitialized device.

ReadAsync(CancellationToken)

Asynchronously read from the device.

Declaration

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

Parameters

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

Returns

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

Remarks

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

Exceptions

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

RefreshAsync()

Asynchronously refresh the device settings.

Declaration

```
public abstract Task RefreshAsync()
```

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

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

ResetAsync()

Asynchronously reset the device.

Declaration

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

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

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

Exceptions

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

Uninitialize()

Synchronously uninitialize device settings and close the connection.

Declaration

```
public virtual void Uninitialize()
```

WriteAsync(String, CancellationToken)

Asynchronously write a command to the device.

Declaration

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

Parameters

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

Returns

TYPE	DESCRIPTION

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

Implements

IDevice
System.IDisposable

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

Object

ReadingValue

Inherited Members

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

Object
PhysicalDevice
SantecDevice
BRM
PTM
PLM
OPM
RLM
OSX

Implements

ISantecDevice
IDevice
IDisposable

Inherited Members

PhysicalDevice.Address
PhysicalDevice.Name
PhysicalDevice.SerialNumber
PhysicalDevice.FirmwareVersion
PhysicalDevice.IsInitialized
PhysicalDevice.IsInitializing
PhysicalDevice.Uninitialize()
PhysicalDevice.Dispose()
PhysicalDevice.PingAsync()
PhysicalDevice.ResetAsync()
PhysicalDevice.RefreshAsync()
PhysicalDevice.QueryAsync(String, CancellationToken)
PhysicalDevice.WriteAsync(String, CancellationToken)
PhysicalDevice.ReadAsync(CancellationToken)
Object.ToString()
Object.Equals(Object)
Object.Equals(Object, Object)
Object.ReferenceEquals(Object, Object)
Object.GetHashCode()
Object.GetType()
Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.General

Assembly: Santec.Hardware.dll

Syntax

```
public abstract class SantecDevice : PhysicalDevice, ISantecDevice, IDevice, IDisposable
```

Methods

EnableDHCPAsync(CancellationToken)

Asynchronously enable the device to use DHCP to acquire its IP address.

Declaration

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

Parameters

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

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

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

Exceptions

TYPE	CONDITION
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.
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

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

Parameters

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

Returns

TYPE	DESCRIPTION
Task<IPAddress>	The task representing the asynchronous operation.

Remarks

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

Exceptions

TYPE	CONDITION
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.
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

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

Parameters

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

Returns

TYPE	DESCRIPTION
Task<String>	The task representing the asynchronous operation.

Remarks

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

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device mDNS hostname response is not a valid hostname.
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

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

Parameters

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

Returns

TYPE	DESCRIPTION
Task<EInteractionMode>	The task representing the asynchronous operation.

Remarks

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

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device response does not correspond to an interaction mode.

TYPE	CONDITION
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

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

Parameters

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

Returns

TYPE	DESCRIPTION
Task<IPAddress>	The task representing the asynchronous operation.

Remarks

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

Exceptions

TYPE	CONDITION
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.
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

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

Parameters

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

Returns

TYPE	DESCRIPTION
Task<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
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

InitializeAsync()

Asynchronously open the device connection and initialize the device settings.

Declaration

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

Returns

TYPE	DESCRIPTION
Task	The task representing the operation.

Overrides

PhysicalDevice.InitializeAsync()

Remarks

> **NOTE**

This operation will not block.

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.
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

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

Parameters

TYPE	NAME	DESCRIPTION
IPAddress	address	The gateway IP address.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

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

TYPE	CONDITION
UnexpectedDeviceResponseException	Thrown when the device gateway address response does not match the expected response.
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

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

Parameters

TYPE	NAME	DESCRIPTION
String	hostname	The mDNS hostname.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified mDNS hostname is null, empty, or whitespace.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the mDNS hostname response does not match the expected response.

TYPE	CONDITION
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

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

Parameters

TYPE	NAME	DESCRIPTION
EInteractionMode	interactionMode	The interaction mode to set the device to.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
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.
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

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

Parameters

TYPE	NAME	DESCRIPTION
IPAddress	address	The IP address.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

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

Exceptions

TYPE	CONDITION
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.
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

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

Parameters

TYPE	NAME	DESCRIPTION
Int32	prefixLength	The subnet prefix length.

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

Returns

TYPE	DESCRIPTION
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
ArgumentException	Thrown when the subnet prefix length is not valid (less than `0` or greater than `31`).
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

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

Object
SimulatedDeviceFactory

Inherited Members

Object.ToString()
Object.Equals(Object)
Object.Equals(Object, Object)
Object.ReferenceEquals(Object, Object)
Object.GetHashCode()
Object.GetType()
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
String	serialNumber	The serial number of the BRM.
IEnumerable<Int32>	wavelengths	The wavelengths of the BRM.
Int32	numberOfDetectors	The number of detectors for the BRM.
IEnumerable<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
ArgumentException	Thrown when the simulated BRM is given an invalid number of wavelengths or detectors.
ArgumentException	Thrown when the simulated BRM is provided an invalid number of switches.

TYPE	CONDITION
ArgumentException	Thrown when any simulated BRM switch is given an invalid number of channels.

CreateSimulatedOSX(String, IEnumerable<SwitchModule>, Version)

Creates a new simulated OSX.

Declaration

```
public SimulatedOSX CreateSimulatedOSX(string serialNumber, IEnumerable<SwitchModule> modules, Version
firmwareVersion = null)
```

Parameters

TYPE	NAME	DESCRIPTION
String	serialNumber	The serial number of the OSX.
IEnumerable<SwitchModule>	modules	The switch modules of the OSX.
Version	firmwareVersion	The firmware version of the OSX.

Returns

TYPE	DESCRIPTION
SimulatedOSX	

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the simulated OSX is given an invalid number of modules.
ArgumentException	Thrown when any OSX module is given an invalid number of channels or commons.

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
String	serialNumber	The serial number of the PLM.
IEnumerable<Int32>	wavelengths	The wavelengths of the PLM.
Int32	numberOfDetectors	The number of detectors for the PLM.
IEnumerable<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
ArgumentException	Thrown when the simulated PLM is given an invalid number of wavelengths or detectors.
ArgumentException	Thrown when the simulated PLM is provided an invalid number of switches.
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
String	serialNumber	The serial number of the PTM.
Int32	numberOfChannels	The number of channels on the PTM.
Version	firmwareVersion	The firmware version of the PTM.

Returns

TYPE	DESCRIPTION
SimulatedPTM	

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the simulated PTM is given an invalid number of channels.

CreateSimulatedRLM(String, IEnumerable<Int32>, Int32, IEnumerable<Int32>, Version, EFiberType, ECoreSize)

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)
```

Parameters

TYPE	NAME	DESCRIPTION
String	serialNumber	The serial number of the RLM.
IEnumerable<Int32>	wavelengths	The wavelengths of the RLM.
Int32	numberOfDetectors	The number of detectors for the RLM.
IEnumerable<Int32>	switchChannelCounts	The channel count for RLM switches.

TYPE	NAME	DESCRIPTION
Version	firmwareVersion	The firmware version of the RLM.
EFiberType	fiberType	The fiber type of the RLM.
ECoreSize	coreSize	The core size 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
ArgumentException	Thrown when the simulated RLM is given an invalid number of wavelengths or detectors.
ArgumentException	Thrown when the simulated RLM is provided an invalid number of switches.
ArgumentException	Thrown when any simulated RLM switch is given an invalid number of channels.

Class SimulatedSantecDevice

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

Inheritance

Object
SimulatedSantecDevice
SimulatedBRM
SimulatedPTM
SimulatedPLM
SimulatedRLM
SimulatedOSX

Implements

ISantecDevice
IDevice
IDisposable

Inherited Members

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

Namespace: Santec.Hardware.Devices.General

Assembly: Santec.Hardware.dll

Syntax

```
public abstract class SimulatedSantecDevice : ISantecDevice, IDevice, IDisposable
```

Fields

creatingDevice

Declaration

```
protected bool creatingDevice
```

Field Value

TYPE	DESCRIPTION
Boolean	

Properties

Address

Get the connection address of the device.

Declaration

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

Property Value

TYPE	DESCRIPTION
String	The connection address of the device.

Remarks

> **NOTE**
>
> This property returns "N/A" for the simulated device unless an IP address has been explicitly set.

FirmwareVersion

Get the firmware version of the device.

Declaration

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

Property Value

TYPE	DESCRIPTION
Version	The firmware version of the device.

IsInitialized

Get whether the device and connection are initialized.

Declaration

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

Property Value

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

Name

Get the name of the device.

Declaration

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

Property Value

TYPE	DESCRIPTION
String	The name of the device.

SerialNumber

Get the serial number of the device.

Declaration

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

Property Value

TYPE	DESCRIPTION
String	The serial number of the device.

Methods

Dispose()

Declaration

```
public void Dispose()
```

EnableDHCPAsync(CancellationToken)

Asynchronously enable the device to use DHCP to acquire its IP address.

Declaration

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

Parameters

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

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

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

TYPE	CONDITION
UnexpectedDeviceResponseException	Thrown when the device enable DHCP response does not match the expected response.
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

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

Parameters

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

Returns

TYPE	DESCRIPTION
Task<IPAddress>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
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.
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

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

Parameters

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

Returns

TYPE	DESCRIPTION
Task<String>	The task representing the asynchronous operation.

Remarks

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

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device mDNS hostname response is not a valid hostname.
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

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

Parameters

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

Returns

TYPE	DESCRIPTION
Task<EInteractionMode>	The task representing the asynchronous operation.

Remarks

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

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device response does not correspond to an interaction mode.
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

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

Parameters

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

Returns

TYPE	DESCRIPTION
Task<IPAddress>	The task representing the asynchronous operation.

Remarks

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

Exceptions

TYPE	CONDITION
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.
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

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

Parameters

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

Returns

TYPE	DESCRIPTION
Task<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
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

InitializeAsync()

Asynchronously open the device connection and initialize the device settings.

Declaration

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

Returns

TYPE	DESCRIPTION
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

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

Returns

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

Remarks

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

QueryAsync(String, CancellationToken)

Asynchronously send a query to the device.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
String	command	The SCPI command to send to the device.

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

Returns

TYPE	DESCRIPTION
Task<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
InvalidOperationException	Thrown when the method is called on an uninitialized device.

ReadAsync(CancellationToken)

Asynchronously read from the device.

Declaration

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

Parameters

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

Returns

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

Remarks

NOTE

This operation will not block.

Exceptions

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

RefreshAsync()

Asynchronously refresh the device settings.

Declaration

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

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

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

ResetAsync()

Asynchronously reset the device.

Declaration

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

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

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

SetGatewayAsync(IPAddress, CancellationToken)

Asynchronously set the gateway address of the device.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
IPAddress	address	The gateway IP address.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

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

Exceptions

TYPE	CONDITION
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.
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

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

Parameters

TYPE	NAME	DESCRIPTION
String	hostname	The mDNS hostname.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

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

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified mDNS hostname is null, empty, or whitespace.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the mDNS hostname response does not match the expected response.
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

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

Parameters

TYPE	NAME	DESCRIPTION
EInteractionMode	interactionMode	The interaction mode to set the device to.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
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.
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

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

Parameters

TYPE	NAME	DESCRIPTION
IPAddress	address	The IP address.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

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

TYPE	CONDITION
UnexpectedDeviceResponseException	Thrown when the device IP address response does not match the expected response.
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

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

Parameters

TYPE	NAME	DESCRIPTION
Int32	prefixLength	The subnet prefix length.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
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
ArgumentException	Thrown when the subnet prefix length is not valid (less than `0` or greater than `31`).
InvalidOperationException	Thrown when the method is called on an uninitialized device.

TYPE	CONDITION
UnexpectedDeviceResponseException	Thrown when the device subnet prefix length response does not match the expected response.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

Uninitialize()

Synchronously uninitialize device settings and close the connection.

Declaration

```
public void Uninitialize()
```

WriteAsync(String, CancellationToken)

Asynchronously write a command to the device.

Declaration

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

Parameters

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

Returns

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

Implements

ISantecDevice
IDevice
System.IDisposable

Namespace Santec.Hardware.Devices.Polarity

Classes

CableInformation

Represents cable information about a cable in a multi-cable polarity mapping operation.

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 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 NotSupportedException.

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

Interfaces

IPTM

Defines the properties of a PTM 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.

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

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.

Specifies the mapping sensitivity of an RD-1.

This corresponds to the strictness of the threshold used when differentiating rows/columns during the mapping data processing.

Class CableInformation

Represents cable information about a cable in a multi-cable polarity mapping operation.

Inheritance

Object
CableInformation

Inherited Members

Object.ToString()
Object.Equals(Object)
Object.Equals(Object, Object)
Object.ReferenceEquals(Object, Object)
Object.GetHashCode()
Object.GetType()
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
Int32	numberOfFibers	The number of fibers in the cable.
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
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
Int32	The mapping offset of the cable.

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.

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.ResetAsync()
IDevice.RefreshAsync()
IDevice.PingAsync()
IDevice.QueryAsync(String, CancellationToken)
IDevice.WriteAsync(String, CancellationToken)
IDevice.ReadAsync(CancellationToken)
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
Int32	The number of channels on the device.

Exceptions

TYPE	CONDITION
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
Int32	channel	The output channel to map the detector for.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Nullable<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
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device map detector response is not a valid response.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

TYPE	CONDITION
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
Int32	numberOfChannels	The number of output channels to map.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<IReadOnlyList<Nullable<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
ArgumentException	Thrown when the specified number of channels is out of range for the device.
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.

TYPE	CONDITION
UnexpectedDeviceResponseException	Thrown when any of the mapping values are not valid response values.
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
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device laser response does not match the expected response.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.
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
Int32	channel	The output channel to turn on the laser for.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified number of channels is out of range for the device.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device flashing VFL response does not match the expected response.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.
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
Int32	channel	The output channel to turn on the laser for.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified number of channels is out of range for the device.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device VFL response does not match the expected response.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.
NotSupportedException	Thrown when the operation is not supported by the device.

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
IDevice.Name
IDevice.SerialNumber
IDevice.FirmwareVersion
IDevice.Address
IDevice.IsInitialized
IDevice.InitializeAsync()
IDevice.Uninitialize()
IDevice.ResetAsync()
IDevice.RefreshAsync()
IDevice.PingAsync()
IDevice.QueryAsync(String, CancellationToken)
IDevice.WriteAsync(String, CancellationToken)
IDevice.ReadAsync(CancellationToken)
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
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 CollectOutputMappingDataAsync(int cable, int fiber, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	cable	The cable to collect the mapping data for.
Int32	fiber	The fiber of the cable to collect the mapping data for.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

The fiber 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
ArgumentException	Thrown when the specified cable is out of range for the polarity mapping operation.
ArgumentException	Thrown when the specified fiber is out of range for the specified cable.

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized device.
InvalidOperationException	Thrown when a polarity mapping operation is not in progress.
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 CollectOutputMappingDataAsync(int channel, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	channel	The output channel to collect the mapping data for.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	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
ArgumentException	Thrown when the specified channel is out of range for the polarity mapping operation.
InvalidOperationException	Thrown when the method is called on an uninitialized device.

TYPE	CONDITION
InvalidOperationException	Thrown when a polarity mapping operation is not in progress.
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
Task	The task representing the asynchronous operation.

Remarks

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

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized device.
InvalidOperationException	Thrown when a polarity mapping operation is not in progress.
InvalidOperationException	Thrown when a segmented polarity mapping operation is not in progress.

GetLossCompensation()

Get the loss compensation of the RD-P.

Declaration

```
int GetLossCompensation()
```

Returns

TYPE	DESCRIPTION

TYPE	DESCRIPTION
Int32	The loss compensation input power in dBm.

Exceptions

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

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
Int32	numberOfChannels	The number of output channels to map.
IEnumerable<Nullable<Int32>>	expectedMapping	The expected mapping result.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<IReadOnlyList<Nullable<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
ArgumentException	Thrown when the specified number of channels is out of range for the device.
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.
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
IEnumerable<Nullable<Int32>>	expectedMapping	Optional: The expected mapping results.

Returns

TYPE	DESCRIPTION
IReadOnlyList<Nullable<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
InvalidOperationException	Thrown when the method is called on an uninitialized device.

TYPE	CONDITION
InvalidOperationException	Thrown when a polarity mapping operation is not in progress.
InvalidOperationException	Thrown when a segmented polarity mapping operation is not in progress.
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
Int32	cable	The 1-based index of the cable to process the mapping data for.
IEnumerable<Nullable<Int32>>	expectedMapping	Optional: The expected mapping results.

Returns

TYPE	DESCRIPTION
IReadOnlyList<Nullable<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
ArgumentException	Thrown when the specified cable index is out of range for the polarity mapping operation.

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized device.
InvalidOperationException	Thrown when a polarity mapping operation is not in progress.
InvalidOperationException	Thrown when a segmented polarity mapping operation is not in progress.
InvalidOperationException	Thrown when a multi-cable polarity mapping operation is not in progress.
InvalidOperationException	Thrown when data is missing for one or more outputs.
PolarityMappingFailedException	Thrown when the device cannot determine a mapping of outputs to inputs.

SetLossCompensation(Int32)

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

```
void SetLossCompensation(int inputPower)
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	inputPower	The input power into the camera.

Exceptions

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

TYPE	NAME	DESCRIPTION
IEnumerable<CableInformation>	cables	The cable information of the cables.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified number of fibers for a cable is out of range for the device.
ArgumentException	Thrown when the specified cable offset for a cable is less then 0.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
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
Int32	numberOfChannels	The number of output channels to map.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

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

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified number of channels is out of range for the device.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
InvalidOperationException	Thrown when a polarity mapping operation is already in progress.

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
Boolean	`true` if a 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.

Declaration

```
bool CheckIfRLMMeetsHardwareRequirements(IRLM rlm)
```

Parameters

TYPE	NAME	DESCRIPTION
IRLM	rlm	The RLM to check.

Returns

TYPE	DESCRIPTION
Boolean	`true` if the RLM meets the hardware requirements; otherwise, `false`.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified RLM is uninitialized.

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.
ArgumentException	Thrown when the specified PTM is uninitialized.
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.
ArgumentException	Thrown when the specified RLM is uninitialized.
ArgumentException	Thrown when the specified RLM does not meet the RD-P hardware requirements.

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.GetLossCompensation()
IRDP.SetLossCompensation(Int32)
IRDP.GetMappingInProgress()
IRDP.StartMappingPolarityAsync(Int32)
IRDP.StartMappingPolarityAsync(IEnumerable<CableInformation>)
IRDP.EndMappingPolarityAsync()
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)
IPTM.NumberOfChannels
IPTM.MapPolarityAsync(Int32, CancellationToken)
IPTM.TurnOnVFLModeAsync(Int32, CancellationToken)
IPTM.TurnOnFlashingVFLModeAsync(Int32, CancellationToken)
IPTM.TurnOffLaserAsync(CancellationToken)
IPTM.MapOutputAsync(Int32, CancellationToken)
ICompositeDevice.Subdevices
IRLM.SupportsILResolution
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(CancellationToken)
IRLM.SetDUTILAsync(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)
IPowerMeter.ReadPowerAsync(Int32, Int32, CancellationToken)
IPowerMeter.GetILReferenceAsync(Int32, Int32, CancellationToken)
IPowerMeter.SetILReferenceAsync(Int32, Int32, Double, CancellationToken)
IPowerMeter.ReferenceILAsync(Int32, Int32, CancellationToken)
IPowerMeter.ReferenceILAsync(Int32, Int32, Boolean, CancellationToken)
IPowerMeter.ReadILAsync(Int32, Int32, CancellationToken)
IDetectorDevice.NumberOfDetectors
IDetectorDevice.Detectors
ILaserSource.NumberOfOutputs
ILaserSource.GetActiveLaserAsync(CancellationToken)
ILaserSource.TurnOnLaserAsync(Int32, CancellationToken)
ILaserSource.GetOutputChannelAsync(CancellationToken)
ILaserSource.ChangeOutputChannelAsync(Int32, CancellationToken)
ILaserSource.ReadMonitorPowerAsync(Int32, CancellationToken)
ILaserSource.GetFactoryPowerAsync(Int32, Int32, CancellationToken)
IWavelengthDevice.Wavelengths
ISwitchController.Switches
ISwitchController.GetSwitchChannelAsync(Int32, CancellationToken)
ISwitchController.ChangeSwitchChannelAsync(Int32, Int32, CancellationToken)
ISwitchController.ChangeMultipleSwitchesChannelsAsync(IEnumerable<DeviceSwitchAndChannelPair>, CancellationToken)
IDevice.Name
IDevice.SerialNumber
IDevice.FirmwareVersion
IDevice.Address
IDevice.IsInitialized
IDevice.InitializeAsync()
IDevice.Uninitialize()
IDevice.ResetAsync()
IDevice.RefreshAsync()
IDevice.PingAsync()
IDevice.QueryAsync(String, CancellationToken)
IDevice.WriteAsync(String, CancellationToken)
IDevice.ReadAsync(CancellationToken)
IDisposable.Dispose()
IFiberDevice.Fiber

Namespace: Santec.Hardware.Devices.Polarity

Assembly: Santec.Hardware.dll

Syntax

```
public interface IRDSP : IRDP, IPTM, ICompositeDevice, IRLM, ISantecDevice, IDiscretePowerMeter,
IPowerMeter, IDetectorDevice, ILaserSource, IWavelengthDevice, ISwitchController, IDevice, IDisposable,
IFiberDevice
```

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

Object
PhysicalDevice
SantecDevice
PTM

Implements

IPTM
ISantecDevice
IDevice
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.Uninitialize()
PhysicalDevice.Dispose()
PhysicalDevice.PingAsync()
PhysicalDevice.ResetAsync()
PhysicalDevice.QueryAsync(String, CancellationToken)
PhysicalDevice.WriteAsync(String, CancellationToken)
PhysicalDevice.ReadAsync(CancellationToken)
Object.ToString()
Object.Equals(Object)
Object.Equals(Object, Object)
Object.ReferenceEquals(Object, Object)
Object.GetHashCode()

Object.GetType()
Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.Polarity

Assembly: Santec.Hardware.dll

Syntax

```
public class PTM : SantecDevice, IPTM, ISantecDevice, IDevice, IDisposable
```

Properties

Name

Get the name of the device.

Declaration

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

Property Value

TYPE	DESCRIPTION
String	The name of the device.

Overrides

PhysicalDevice.Name

Remarks

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

NumberOfChannels

Get the number of channels on the device.

Declaration

```
public int NumberOfChannels { get; }
```

Property Value

TYPE	DESCRIPTION
Int32	The number of channels on the device.

Exceptions

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

Methods

InitializeAsync()

Asynchronously open the device connection and initialize the device settings.

Declaration

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

Returns

TYPE	DESCRIPTION
Task	The task representing the operation.

Overrides

SantecDevice.InitializeAsync()

Remarks

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

Exceptions

TYPE	CONDITION
UnexpectedDeviceResponseException	Thrown when the PTM channels response is not a valid number.
TimeoutException	Thrown when a communication failure causes no response to be received from the PTM.

MapOutputAsync(Int32, CancellationToken)

Asynchronously map the detector of a specified output channel.

Declaration

```
public async Task<int?> MapOutputAsync(int channel, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	channel	The output channel to map the detector for.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Nullable<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
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device map detector response is not a valid response.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.
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

```
public async Task<IReadOnlyList<int?>> MapPolarityAsync(int numberOfChannels, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	numberOfChannels	The number of output channels to map.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<IReadOnlyList<Nullable<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
ArgumentException	Thrown when the specified number of channels is out of range for the device.
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.
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.

RefreshAsync()

Asynchronously refresh the device settings.

Declaration

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

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Overrides

PhysicalDevice.RefreshAsync()

Remarks

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

Exceptions

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

TurnOffLaserAsync(CancellationToken)

Asynchronously turn off the active laser.

Declaration

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

Parameters

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

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

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

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

Parameters

TYPE	NAME	DESCRIPTION
Int32	channel	The output channel to turn on the laser for.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

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

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

Parameters

TYPE	NAME	DESCRIPTION
Int32	channel	The output channel to turn on the laser for.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified number of channels is out of range for the device.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device VFL response does not match the expected response.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.
NotSupportedException	Thrown when the operation is not supported by the device.

Implements

IPTM
ISantecDevice
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 NotSupportedException.

Inheritance

Object
CompositeSantecDevice
RDP

Implements

IRDP
IPTM
ISantecDevice
ICompositeDevice
IDevice
IDisposable

Inherited Members

CompositeSantecDevice.SerialNumber
CompositeSantecDevice.FirmwareVersion
CompositeSantecDevice.IsInitialized
CompositeSantecDevice.IsInitializing
CompositeSantecDevice.Subdevices
CompositeSantecDevice.Dispose()
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)
Object.ToString()
Object.Equals(Object)

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

Namespace: Santec.Hardware.Devices.Polarity

Assembly: Santec.Hardware.dll

Syntax

```
public class RDP : CompositeSantecDevice, IRDP, IPTM, ISantecDevice, ICompositeDevice, IDevice, IDisposable
```

Properties

Address

Get the connection address of the device.

Declaration

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

Property Value

TYPE	DESCRIPTION
String	The connection address of the device.

Overrides

CompositeSantecDevice.Address

Name

Get the name of the device.

Declaration

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

Property Value

TYPE	DESCRIPTION
String	The name of the device.

Overrides

CompositeSantecDevice.Name

Remarks

> **NOTE**
>
> This property will always return "RD-P".

NumberOfChannels

Get the number of channels on the device.

Declaration

```
public int NumberOfChannels { get; }
```

Property Value

TYPE	DESCRIPTION
Int32	The number of channels on the device.

Exceptions

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

Wavelength

Gets the wavelength of the RD-P.

Declaration

```
public int Wavelength { get; }
```

Property Value

TYPE	DESCRIPTION
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

```
public Task CollectOutputMappingDataAsync(int cable, int fiber, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	cable	The cable to collect the mapping data for.
Int32	fiber	The fiber of the cable to collect the mapping data for.

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

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

The fiber 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
ArgumentException	Thrown when the specified cable is out of range for the polarity mapping operation.
ArgumentException	Thrown when the specified fiber is out of range for the specified cable.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
InvalidOperationException	Thrown when a polarity mapping operation is not in progress.
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

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

Parameters

TYPE	NAME	DESCRIPTION
Int32	channel	The output channel to collect the mapping data for.

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

Returns

TYPE	DESCRIPTION
Task	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
ArgumentException	Thrown when the specified channel is out of range for the polarity mapping operation.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
InvalidOperationException	Thrown when a polarity mapping operation is not in progress.
InvalidOperationException	Thrown when a segmented polarity mapping operation is not in progress.

EndMappingPolarityAsync()

Asynchronously end a segmented polarity mapping operation.

Declaration

```
public Task EndMappingPolarityAsync()
```

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

> **NOTE**

This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized device.
InvalidOperationException	Thrown when a polarity mapping operation is not in progress.
InvalidOperationException	Thrown when a segmented polarity mapping operation is not in progress.

Finalize()

Destroys the RD-P.

Declaration

```
protected void Finalize()
```

GetLossCompensation()

Get the loss compensation of the RD-P.

Declaration

```
public int GetLossCompensation()
```

Returns

TYPE	DESCRIPTION
Int32	The loss compensation input power in dBm.

Exceptions

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

GetMappingInProgress()

Get whether a polarity mapping operation is in progress.

Declaration

```
public bool GetMappingInProgress()
```

Returns

TYPE	DESCRIPTION
Boolean	`true` if a polarity mapping operation is in progress; otherwise, `false`.

GetMappingMode()

Get the mapping mode of the RD-P.

Declaration

```
public EMappingMode GetMappingMode()
```

Returns

TYPE	DESCRIPTION
EMappingMode	The mapping mode of the RD-P.

Exceptions

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

GetMappingSensitivity()

Get the mapping sensitivity of the RD-P.

Declaration

```
public EMappingSensitivity GetMappingSensitivity()
```

Returns

TYPE	DESCRIPTION
EMappingSensitivity	The mapping sensitivity of the RD-P.

Exceptions

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

InitializeAsync()

Asynchronously open the device connection and initialize the device settings.

Declaration

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

Returns

TYPE	DESCRIPTION
Task	The task representing the operation.

Overrides

CompositeSantecDevice.InitializeAsync()

Remarks

NOTE

This operation will not block.

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 NotSupportedException.

Declaration

```
public Task<int?> MapOutputAsync(int channel, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	channel	The output channel to map the detector for.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Nullable<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
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device map detector response is not a valid response.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.
NotSupportedException	Thrown when the operation is not supported by the device.

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

```
public Task<IReadOnlyList<int?>> MapPolarityAsync(int numberOfChannels, IEnumerable<int?> expectedMapping,
CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	numberOfChannels	The number of output channels to map.
IEnumerable<Nullable<Int32>>	expectedMapping	The expected mapping result.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<IReadOnlyList<Nullable<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
ArgumentException	Thrown when the specified number of channels is out of range for the device.
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.
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.

MapPolarityAsync(Int32, CancellationToken)

Asynchronously map the polarity for a set of output channels.

Declaration

```
public Task<IReadOnlyList<int?>> MapPolarityAsync(int numberOfChannels, CancellationToken cancellationToken
= default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	numberOfChannels	The number of output channels to map.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<IReadOnlyList<Nullable<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
ArgumentException	Thrown when the specified number of channels is out of range for the device.
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.
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

```
public IReadOnlyList<int?> ProcessMappingData(IEnumerable<int?> expectedMapping = null)
```

Parameters

TYPE	NAME	DESCRIPTION
IEnumerable<Nullable<Int32>>	expectedMapping	Optional: The expected mapping results.

Returns

TYPE	DESCRIPTION
IReadOnlyList<Nullable<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
InvalidOperationException	Thrown when the method is called on an uninitialized device.
InvalidOperationException	Thrown when a polarity mapping operation is not in progress.
InvalidOperationException	Thrown when a segmented polarity mapping operation is not in progress.
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

```
public IReadOnlyList<int?> ProcessMappingData(int cableIndex, IEnumerable<int?> expectedMapping = null)
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	cableIndex	
IEnumerable<Nullable<Int32>>	expectedMapping	Optional: The expected mapping results.

Returns

TYPE	DESCRIPTION
IReadOnlyList<Nullable<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
ArgumentException	Thrown when the specified cable index is out of range for the polarity mapping operation.

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized device.
InvalidOperationException	Thrown when a polarity mapping operation is not in progress.
InvalidOperationException	Thrown when a segmented polarity mapping operation is not in progress.
InvalidOperationException	Thrown when a multi-cable polarity mapping operation is not in progress.
InvalidOperationException	Thrown when data is missing for one or more outputs.
PolarityMappingFailedException	Thrown when the device cannot determine a mapping of outputs to inputs.

RefreshAsync()

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

Declaration

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

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Overrides

CompositeSantecDevice.RefreshAsync()

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
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
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the RLM subdevice during its refresh operation.

SetLossCompensation(Int32)

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

```
public void SetLossCompensation(int inputPower)
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	inputPower	The input power into the camera.

Exceptions

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

```
public void SetMappingMode(EMappingMode mappingMode)
```

Parameters

TYPE	NAME	DESCRIPTION
EMappingMode	mappingMode	The mapping mode to change the RD-P to.

Exceptions

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

SetMappingSensitivity(EMappingSensitivity)

Set the mapping sensitivity of the RD-P.

Declaration

```
public void SetMappingSensitivity(EMappingSensitivity mappingSensitivity)
```

Parameters

TYPE	NAME	DESCRIPTION
EMappingSensitivity	mappingSensitivity	The mapping sensitivity to change the RD-P to.

Exceptions

TYPE	CONDITION
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

```
public Task StartMappingPolarityAsync(IEnumerable<CableInformation> cables)
```

Parameters

TYPE	NAME	DESCRIPTION
IEnumerable<CableInformation>	cables	The cable information of the cables.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

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

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified number of fibers for a cable is out of range for the device.
ArgumentException	Thrown when the specified cable offset for a cable is less then 0.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
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

```
public Task StartMappingPolarityAsync(int numberOfChannels)
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	numberOfChannels	The number of output channels to map.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

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

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified number of channels is out of range for the device.

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized device.
InvalidOperationException	Thrown when a polarity mapping operation is already in progress.

TurnOffLaserAsync(CancellationToken)

> **⚠ CAUTION**
>
> This operation is not supported for RD-SP devices. This operation will always throw a NotSupportedException.

Declaration

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

Parameters

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

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

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

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device laser response does not match the expected response.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.
NotSupportedException	Thrown when the operation is not supported by the device.

TurnOnFlashingVFLModeAsync(Int32, CancellationToken)

> **⚠ CAUTION**
>
> This operation is not supported for RD-P devices. This operation will always throw a NotSupportedException.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
Int32	channel	The output channel to turn on the laser for.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

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

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified number of channels is out of range for the device.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device flashing VFL response does not match the expected response.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.
NotSupportedException	Thrown when the operation is not supported by the device.

TurnOnVFLModeAsync(Int32, CancellationToken)

> **⚠ CAUTION**
>
> This operation is not supported for RD-P devices. This operation will always throw a NotSupportedException.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
Int32	channel	The output channel to turn on the laser for.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

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

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified number of channels is out of range for the device.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device VFL response does not match the expected response.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.
NotSupportedException	Thrown when the operation is not supported by the device.

Uninitialize()

Synchronously uninitialize device settings and close the connection.

Declaration

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

Overrides

CompositeSantecDevice.Uninitialize()

Implements

IRDP
IPTM
ISantecDevice
ICompositeDevice
IDevice
System.IDisposable

Class RDPFactory

Provides a factory for creating RD-P devices.

Inheritance

Object
RDPFactory

Implements

IRDPFactory

Inherited Members

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

Check for an RD-P camera on the system.

Declaration

```
public bool CheckForCamera()
```

Returns

TYPE	DESCRIPTION
Boolean	`true` if a 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.

Declaration

```
public bool CheckIfRLMMeetsHardwareRequirements(IRLM rlm)
```

Parameters

TYPE	NAME	DESCRIPTION
IRLM	rlm	The RLM to check.

Returns

TYPE	DESCRIPTION
Boolean	`true` if the RLM meets the hardware requirements; otherwise, `false`.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified RLM is uninitialized.

CreateRDP(IPTM)

Creates a new RD-P with the specified PTM.

> **NOTE**
>
> PTM must be initialized and have a camera connected.

Declaration

```
public 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.
ArgumentException	Thrown when the specified PTM is uninitialized.
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

```
public 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.
ArgumentException	Thrown when the specified RLM is uninitialized.
ArgumentException	Thrown when the specified RLM does not meet the RD-P hardware requirements.

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 NotSupportedException.

Inheritance

Object
CompositeSantecDevice
RDSP

Implements

IRDSP
IRDP
IPTM
ICompositeDevice
IRLM
ISantecDevice
IDiscretePowerMeter
IPowerMeter
IDetectorDevice
ILaserSource
IWavelengthDevice
ISwitchController
IDevice
IDisposable
IFiberDevice

Inherited Members

CompositeSantecDevice.SerialNumber
CompositeSantecDevice.FirmwareVersion
CompositeSantecDevice.IsInitialized
CompositeSantecDevice.IsInitializing
CompositeSantecDevice.Subdevices
CompositeSantecDevice.Dispose()
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.QueryAsync(String, CancellationToken)
CompositeSantecDevice.WriteAsync(String, CancellationToken)
CompositeSantecDevice.ReadAsync(CancellationToken)
Object.ToString()
Object.Equals(Object)
Object.Equals(Object, Object)
Object.ReferenceEquals(Object, Object)
Object.GetHashCode()
Object.GetType()
Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.Polarity

Assembly: Santec.Hardware.dll

Syntax

```
public class RDSP : CompositeSantecDevice, IRDSP, IRDP, IPTM, ICompositeDevice, IRLM, ISantecDevice,
IDiscretePowerMeter, IPowerMeter, IDetectorDevice, ILaserSource, IWavelengthDevice, ISwitchController,
IDevice, IDisposable, IFiberDevice
```

Properties

Address

Get the connection address of the device.

Declaration

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

Property Value

TYPE	DESCRIPTION
String	The connection address of the device.

Overrides

CompositeSantecDevice.Address

Detectors

Get the detectors connected to the device.

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

Declaration

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

Property Value

TYPE	DESCRIPTION
IReadOnlyList<DeviceDetector>	The list of the detectors connected to the device.

Exceptions

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

Fiber

Get the fiber information of the device.

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

Declaration

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

Property Value

TYPE	DESCRIPTION
FiberInformation	The fiber information of the device.

Exceptions

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

Name

Get the name of the device.

Declaration

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

Property Value

TYPE	DESCRIPTION
String	The name of the device.

Overrides

CompositeSantecDevice.Name

Remarks

> **NOTE**
>
> This property will always return "RD-P".

NumberOfChannels

Get the number of channels on the device.

Declaration

```
public int NumberOfChannels { get; }
```

Property Value

TYPE	DESCRIPTION
Int32	The number of channels on the device.

Exceptions

TYPE	CONDITION
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

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

Property Value

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

Exceptions

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

NumberOfOutputs

Get the number of laser outputs.

NOTE

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

Declaration

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

Property Value

TYPE	DESCRIPTION
Int32	The number of device outputs.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the property is accessed before the device 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

```
public bool SupportsILResolution { get; }
```

Property Value

TYPE	DESCRIPTION
Boolean	`true` if the RLM supports changing the IL resolution; otherwise, `false`.

Exceptions

TYPE	CONDITION

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

Switches

Get the switches connected to the device.

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

Declaration

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

Property Value

TYPE	DESCRIPTION
List<DeviceSwitch>	The list of the switches connected to the device.

Exceptions

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

Wavelength

Gets the wavelength of the RD-P.

Declaration

```
public int Wavelength { get; }
```

Property Value

TYPE	DESCRIPTION
Int32	The RD-P wavelength.

Wavelengths

Get the wavelengths supported by the device.

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

Declaration

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

Property Value

TYPE	DESCRIPTION
List<Int32>	A list of the device's wavelengths.

Exceptions

TYPE	CONDITION
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

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

Parameters

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

Returns

TYPE	DESCRIPTION
Task	The task representing the operation.

Remarks

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

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device has no switch at any of the specified switch indexes.
ArgumentException	Thrown when any the specified channels is out of range for the corresponding device switch.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

ChangeOutputChannelAsync(Int32, CancellationToken)

Asynchronously change the output channel of the laser source.

Declaration

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

Parameters

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

Returns

TYPE	DESCRIPTION
Task	The task representing the operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION

TYPE	CONDITION
ArgumentException	Thrown when the specified output channel is out of range for the laser source.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the RLM.

ChangeSwitchChannelAsync(Int32, Int32, CancellationToken)

Asynchronously change the channel of a connected switch.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
Int32	switchIndex	The switch number of the switch. `0` for the internal switch; `1` or `2` for external switches attached to the device.
Int32	channel	The channel to switch to.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device has no switch at the specified switch index.
ArgumentException	Thrown when the specified channel is out of range for the device switch.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

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

```
public Task CollectOutputMappingDataAsync(int cable, int fiber, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	cable	The cable to collect the mapping data for.
Int32	fiber	The fiber of the cable to collect the mapping data for.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

The fiber 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
ArgumentException	Thrown when the specified cable is out of range for the polarity mapping operation.
ArgumentException	Thrown when the specified fiber is out of range for the specified cable.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
InvalidOperationException	Thrown when a polarity mapping operation is not in progress.
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

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

Parameters

TYPE	NAME	DESCRIPTION
Int32	channel	The output channel to collect the mapping data for.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	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
ArgumentException	Thrown when the specified channel is out of range for the polarity mapping operation.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
InvalidOperationException	Thrown when a polarity mapping operation is not in progress.
InvalidOperationException	Thrown when a segmented polarity mapping operation is not in progress.

EndMappingPolarityAsync()

Asynchronously end a segmented polarity mapping operation.

Declaration

```
public Task EndMappingPolarityAsync()
```

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized device.
InvalidOperationException	Thrown when a polarity mapping operation is not in progress.
InvalidOperationException	Thrown when a segmented polarity mapping operation is not in progress.

GetActiveLaserAsync(CancellationToken)

Asynchronously get the active laser.

Declaration

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

Parameters

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

Returns

TYPE	DESCRIPTION
Task<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
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

GetDUTILAsync(CancellationToken)

Asynchronously get the DUT IL.

Declaration

```
public Task<double> GetDUTILAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

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

Returns

TYPE	DESCRIPTION
Task<Double>	The task representing the operation.

Remarks

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

Exceptions

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

GetFactoryPowerAsync(Int32, Int32, CancellationToken)

Asynchronously get the factory power for the specified output channel and wavelength.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
Int32	output	The output channel to get the factory power for.
Int32	wavelength	The wavelength to get the factory power for.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

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

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified output channel is out of range.
ArgumentException	Thrown when the device does not support the specified wavelength.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device factory power response is not a valid number.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

GetGainModeAsync(CancellationToken)

Asynchronously get the gain mode.

Declaration

```
public Task<EGainMode> GetGainModeAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

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

Returns

TYPE	DESCRIPTION
Task<EGainMode>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized RLM.
UnexpectedDeviceResponseException	Thrown when the device response does not correspond to a gain mode.
TimeoutException	Thrown when a communication failure causes no response to be received from the RLM.

GetILReferenceAsync(Int32, Int32, CancellationToken)

Asynchronously get the stored IL reference for a detector at a given wavelength.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	The detector to get the reference for.
Int32	wavelength	The wavelength to get the reference for.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

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

Remarks

NOTE

This operation will not block.

Exceptions

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

TYPE	CONDITION
ArgumentException	Thrown when the specified detector is out of range for the device.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

GetILResolutionAsync(CancellationToken)

Asynchronously get the IL reading resolution.

Declaration

```
public Task<EResolution> GetILResolutionAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

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

Returns

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

TYPE	CONDITION
InvalidOperationException	Thrown when the RLM does not support IL resolution operations.
UnexpectedDeviceResponseException	Thrown when the device response does not correspond to a resolution.
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

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

Parameters

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

Returns

TYPE	DESCRIPTION
Task<EInteractionMode>	The task representing the asynchronous operation.

Overrides

CompositeSantecDevice.GetInteractionModeAsync(CancellationToken)

Remarks

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

Exceptions

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

TYPE	CONDITION
UnexpectedDeviceResponseException	Thrown when the device response does not correspond to an interaction mode.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

GetLengthModeAsync(CancellationToken)

Asynchronously get the length mode.

Declaration

```
public Task<ELengthMode> GetLengthModeAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

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

Returns

TYPE	DESCRIPTION
Task<ELengthMode>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized RLM.
UnexpectedDeviceResponseException	Thrown when the device response does not correspond to a length mode.
TimeoutException	Thrown when a communication failure causes no response to be received from the RLM.

GetLossCompensation()

Get the loss compensation of the RD-P.

Declaration

```
public int GetLossCompensation()
```

Returns

TYPE	DESCRIPTION
Int32	The loss compensation input power in dBm.

Exceptions

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

GetMappingInProgress()

Get whether a polarity mapping operation is in progress.

Declaration

```
public bool GetMappingInProgress()
```

Returns

TYPE	DESCRIPTION
Boolean	`true` if a polarity mapping operation is in progress; otherwise, `false`.

GetMappingMode()

Get the mapping mode of the RD-P.

Declaration

```
public EMappingMode GetMappingMode()
```

Returns

TYPE	DESCRIPTION
EMappingMode	The mapping mode of the RD-P.

Exceptions

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

GetMappingSensitivity()

Get the mapping sensitivity of the RD-P.

Declaration

```
public EMappingSensitivity GetMappingSensitivity()
```

Returns

TYPE	DESCRIPTION
EMappingSensitivity	The mapping sensitivity of the RD-P.

Exceptions

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

GetOutputChannelAsync(CancellationToken)

Asynchronously get the current output channel of the laser source.

Declaration

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

Parameters

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

Returns

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

Remarks

NOTE

This operation will not block.

Exceptions

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

GetPositionBModeAsync(CancellationToken)

Asynchronously get the RL position B mode.

Declaration

```
public Task<EPositionBMode> GetPositionBModeAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

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

Returns

TYPE	DESCRIPTION
Task<EPositionBMode>	The task representing the asynchronous operation.

Remarks

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

Exceptions

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

GetRLModeAsync(CancellationToken)

Asynchronously get the RL sensitivity mode.

Declaration

```
public Task<ERLMode> GetRLModeAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

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

Returns

TYPE	DESCRIPTION
Task<ERLMode>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized RLM.
UnexpectedDeviceResponseException	Thrown when the device response does not correspond to an RL mode.
TimeoutException	Thrown when a communication failure causes no response to be received from the RLM.

GetRLReferenceAsync(CancellationToken)

Asynchronously get the stored RL reference.

Declaration

```
public Task<double> GetRLReferenceAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

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

Returns

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

Remarks

NOTE

This operation will not block.

Exceptions

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

GetSwitchChannelAsync(Int32, CancellationToken)

Asynchronously get the channel of a connected switch.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
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.
CancellationToken	cancellationToken	

Returns

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

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device has no switch at the specified switch index.

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

GetTraceDiagnosticsAsync(CancellationToken)

Asynchronously get the trace diagnostics for the last RL measurement.

Declaration

```
public Task<TraceDiagnostics> GetTraceDiagnosticsAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

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

Returns

TYPE	DESCRIPTION
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
InvalidOperationException	Thrown when the method is called on an uninitialized RLM.

TYPE	CONDITION
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the RLM.
NotSupportedException	Thrown when the method is called on an RLM through a USB connection.

InitializeAsync()

Asynchronously open the device connection and initialize the device settings.

Declaration

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

Returns

TYPE	DESCRIPTION
Task	The task representing the operation.

Overrides

CompositeSantecDevice.InitializeAsync()

Remarks

NOTE

This operation will not block.

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 NotSupportedException.

Declaration

```
public Task<int?> MapOutputAsync(int channel, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	channel	The output channel to map the detector for.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Nullable<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
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device map detector response is not a valid response.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.
NotSupportedException	Thrown when the operation is not supported by the device.

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

```
public Task<IReadOnlyList<int?>> MapPolarityAsync(int numberOfChannels, IEnumerable<int?> expectedMapping,
CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	numberOfChannels	The number of output channels to map.
IEnumerable<Nullable<Int32>>	expectedMapping	The expected mapping result.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<IReadOnlyList<Nullable<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
ArgumentException	Thrown when the specified number of channels is out of range for the device.
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.
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.

MapPolarityAsync(Int32, CancellationToken)

Asynchronously map the polarity for a set of output channels.

Declaration

```
public Task<IReadOnlyList<int?>> MapPolarityAsync(int numberOfChannels, CancellationToken cancellationToken
= default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	numberOfChannels	The number of output channels to map.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<IReadOnlyList<Nullable<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
ArgumentException	Thrown when the specified number of channels is out of range for the device.
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.
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

```
public IReadOnlyList<int?> ProcessMappingData(IEnumerable<int?> expectedMapping = null)
```

Parameters

TYPE	NAME	DESCRIPTION
IEnumerable<Nullable<Int32>>	expectedMapping	Optional: The expected mapping results.

Returns

TYPE	DESCRIPTION
IReadOnlyList<Nullable<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
InvalidOperationException	Thrown when the method is called on an uninitialized device.
InvalidOperationException	Thrown when a polarity mapping operation is not in progress.
InvalidOperationException	Thrown when a segmented polarity mapping operation is not in progress.
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

```
public IReadOnlyList<int?> ProcessMappingData(int cable, IEnumerable<int?> expectedMapping = null)
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	cable	The 1-based index of the cable to process the mapping data for.
IEnumerable<Nullable<Int32>>	expectedMapping	Optional: The expected mapping results.

Returns

TYPE	DESCRIPTION
IReadOnlyList<Nullable<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
ArgumentException	Thrown when the specified cable index is out of range for the polarity mapping operation.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
InvalidOperationException	Thrown when a polarity mapping operation is not in progress.
InvalidOperationException	Thrown when a segmented polarity mapping operation is not in progress.
InvalidOperationException	Thrown when a multi-cable polarity mapping operation is not in progress.
InvalidOperationException	Thrown when data is missing for one or more outputs.
PolarityMappingFailedException	Thrown when the device cannot determine a mapping of outputs to inputs.

ReadILAsync(Int32, Int32, CancellationToken)

Asynchronously read the insertion loss from a detector at a given wavelength.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	The detector number of the detector to read the insertion loss from.
Int32	wavelength	The wavelength to read the insertion loss at.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

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

Remarks

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

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
ArgumentException	Thrown when the specified detector is out of range for the device.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device read IL response is not a valid number.
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

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

Parameters

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

Returns

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

Remarks

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

Exceptions

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

ReadMultipleILsAsync(IEnumerable<Int32>, Int32, CancellationToken)

Asynchronously read the insertion loss from multiple detectors at a given wavelength.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
IEnumerable<Int32>	detectors	The detectors to read the insertion loss from.

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

Returns

TYPE	DESCRIPTION
Task<List<Double>>	The task representing the asynchronous operation.

Remarks

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

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
ArgumentException	Thrown when any of the specified detectors are out of range for the device.
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.
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

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

Parameters

TYPE	NAME	DESCRIPTION
IEnumerable<Int32>	detectors	The detectors to read from.
Int32	wavelength	The wavelength to read the power at.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<List<Double>>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
ArgumentException	Thrown when any of the specified detectors are out of range for the device.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

ReadPowerAsync(Int32, Int32, CancellationToken)

Asynchronously read the power from an device detector at a given wavelength.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	The detector number of the detector to read from.
Int32	wavelength	The wavelength to read the power at.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

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

Remarks

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

Exceptions

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

ReadRLAsync(Int32, Double, Double, CancellationToken)

Asynchronously read the return loss at a given wavelength using the specified reference lengths.

Declaration

```
public Task<RLReading> ReadRLAsync(int wavelength, double lengthReferenceA, double lengthReferenceB,
CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	wavelength	The wavelength to read the return loss at.
Double	lengthReferenceA	The reference length to use for position A.
Double	lengthReferenceB	The reference length to use for position B.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
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
ArgumentException	Thrown when the RLM does not support the specified wavelength.
ArgumentException	Thrown when either RL reference length is less than `0`.
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.

TYPE	CONDITION
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

```
public Task<RLReading> ReadRLAsync(int wavelength, double lengthReference, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	wavelength	The wavelength to read the return loss at.
Double	lengthReference	The reference length to use for the reading.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
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
ArgumentException	Thrown when the RLM does not support the specified wavelength.
ArgumentException	Thrown when the RLM reference length is less than `0`.

TYPE	CONDITION
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.
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

```
public Task<RLReading> ReadRLAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

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

Returns

TYPE	DESCRIPTION
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

TYPE	CONDITION
ArgumentException	Thrown when the RLM does not support the specified wavelength.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the RLM.

ReferenceILAsync(Int32, Int32, Boolean, CancellationToken)

Asynchronously perform an insertion loss reference for a detector at a given wavelength.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	The detector number of the detector to perform the reference for.
Int32	wavelength	The wavelength to perform the reference at.
Boolean	applyReferenceToAllDetectors	Indicate if the reference read should be applied to all detectors for the current channel.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

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

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
ArgumentException	Thrown when the specified detector is out of range for the device.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device reference IL response is not a valid number.
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

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

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	The detector number of the detector to perform the reference for.
Int32	wavelength	The wavelength to perform the reference at.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

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

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
ArgumentException	Thrown when the specified detector is out of range for the device.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device reference IL response is not a valid number.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

ReferenceRLAsync(CancellationToken)

Asynchronously perform a return loss length reference.

Declaration

```
public Task<double> ReferenceRLAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

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

Returns

TYPE	DESCRIPTION
Task<Double>	The task representing the operation.

Remarks

NOTE

This operation will not block.

Exceptions

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

TYPE	CONDITION
UnexpectedDeviceResponseException	Thrown when the device reference RL response is not a valid number.
TimeoutException	Thrown when a communication failure causes no response to be received from the RLM.

RefreshAsync()

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

Declaration

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

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Overrides

CompositeSantecDevice.RefreshAsync()

Remarks

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

Exceptions

TYPE	CONDITION
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.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the RLM subdevice during its refresh operation.

SetDUTILAsync(Double, CancellationToken)

Asynchronously set the DUT IL.

Declaration

```
public Task SetDUTILAsync(double il, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Double	il	The DUT IL value.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
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
ArgumentException	Thrown when the specified DUT IL is less than `0`.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the RLM.

SetGainModeAsync(EGainMode, CancellationToken)

Asynchronously set the gain mode.

Declaration

```
public Task SetGainModeAsync(EGainMode gainMode, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
EGainMode	gainMode	The gain mode to change to.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

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

Exceptions

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

SetILReferenceAsync(Int32, Int32, Double, CancellationToken)

Asynchronously set the IL reference for a detector at a given wavelength.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	The detector to set the reference for.
Int32	wavelength	The wavelength to set the reference for.

TYPE	NAME	DESCRIPTION
Double	reference	The IL reference value.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
ArgumentException	Thrown when the specified detector is out of range for the device.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

SetILResolutionAsync(EResolution, CancellationToken)

Asynchronously set the IL reading resolution.

Declaration

```
public Task SetILResolutionAsync(EResolution resolution, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
EResolution	resolution	The resolution to set the detector to.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
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
ArgumentException	Thrown when the IL resolution is not supported by the RLM.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
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.
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

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

Parameters

TYPE	NAME	DESCRIPTION
EInteractionMode	interactionMode	The interaction mode to set the device to.

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

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Overrides

CompositeSantecDevice.SetInteractionModeAsync(EInteractionMode, CancellationToken)

Remarks

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

Exceptions

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

SetLengthModeAsync(ELengthMode, CancellationToken)

Asynchronously set the length mode.

Declaration

```
public Task SetLengthModeAsync(ELengthMode lengthMode, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
ELengthMode	lengthMode	The length mode to change to.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

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

Exceptions

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

SetLossCompensation(Int32)

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

```
public void SetLossCompensation(int inputPower)
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	inputPower	The input power into the camera.

Exceptions

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

TYPE	CONDITION
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

```
public void SetMappingMode(EMappingMode mappingMode)
```

Parameters

TYPE	NAME	DESCRIPTION
EMappingMode	mappingMode	The mapping mode to change the RD-P to.

Exceptions

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

SetMappingSensitivity(EMappingSensitivity)

Set the mapping sensitivity of the RD-P.

Declaration

```
public void SetMappingSensitivity(EMappingSensitivity mappingSensitivity)
```

Parameters

TYPE	NAME	DESCRIPTION
EMappingSensitivity	mappingSensitivity	The mapping sensitivity to change the RD-P to.

Exceptions

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

SetPositionBModeAsync(EPositionBMode, CancellationToken)

Asynchronously set the RL position B mode.

Declaration

```
public Task SetPositionBModeAsync(EPositionBMode positionBMode, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
EPositionBMode	positionBMode	The position B mode to change to.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

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

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized RLM.
UnexpectedDeviceResponseException	Thrown when the device positionB mode response does not match the expected response.
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

```
public Task SetRLModeAsync(ERLMode rlMode, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
ERLMode	rlMode	The RL mode to change to.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized RLM.
UnexpectedDeviceResponseException	Thrown when the device RL mode response does not match the expected response.
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

```
public Task SetRLReferenceAsync(double reference, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Double	reference	The RL length reference value.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
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
ArgumentException	Thrown when the RL length reference value is less than `0`.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the RLM.

StartMappingPolarityAsync(IEnumerable<CableInformation>)

Asynchronously start a multi-cable segmented polarity mapping operation for a set of cables.

Declaration

```
public Task StartMappingPolarityAsync(IEnumerable<CableInformation> cables)
```

Parameters

TYPE	NAME	DESCRIPTION
IEnumerable<CableInformation>	cables	The cable information of the cables.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified number of fibers for a cable is out of range for the device.
ArgumentException	Thrown when the specified cable offset for a cable is less then 0.

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

```
public Task StartMappingPolarityAsync(int numberOfChannels)
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	numberOfChannels	The number of output channels to map.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified number of channels is out of range for the device.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
InvalidOperationException	Thrown when a polarity mapping operation is already in progress.

TurnOffLaserAsync(CancellationToken)

Asynchronously turn off the active laser.

Declaration

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

Parameters

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

Returns

TYPE	DESCRIPTION
Task	The task representing the operation.

Remarks

NOTE

This operation will not block.

Exceptions

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

TurnOnFlashingVFLModeAsync(Int32, CancellationToken)

CAUTION

This operation is not supported for RD-SP devices. This operation will always throw a NotSupportedException.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
Int32	channel	The output channel to turn on the laser for.

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

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified number of channels is out of range for the device.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device flashing VFL response does not match the expected response.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.
NotSupportedException	Thrown when the operation is not supported by the device.

TurnOnLaserAsync(Int32, CancellationToken)

Asynchronously turn on the specified laser.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
Int32	wavelength	The wavelength of the laser to turn on.

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

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

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

TurnOnVFLModeAsync(Int32, CancellationToken)

CAUTION

This operation is not supported for RD-P devices. This operation will always throw a NotSupportedException.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
Int32	channel	The output channel to turn on the laser for.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified number of channels is out of range for the device.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device VFL response does not match the expected response.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.
NotSupportedException	Thrown when the operation is not supported by the device.

Uninitialize()

Synchronously uninitialize device settings and close the connection.

Declaration

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

Overrides

CompositeSantecDevice.Uninitialize()

Implements

IRDSP
IRDP
IPTM
ICompositeDevice
IRLM
ISantecDevice
IDiscretePowerMeter
IPowerMeter
IDetectorDevice
ILaserSource
IWavelengthDevice

ISwitchController
IDevice
System.IDisposable
IFiberDevice

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

Object
SimulatedSantecDevice
SimulatedPTM

Implements

IPTM
ISantecDevice
IDevice
IDisposable

Inherited Members

SimulatedSantecDevice.creatingDevice
SimulatedSantecDevice.Dispose()
SimulatedSantecDevice.SerialNumber
SimulatedSantecDevice.FirmwareVersion
SimulatedSantecDevice.Address
SimulatedSantecDevice.IsInitialized
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)
Object.ToString()
Object.Equals(Object)
Object.Equals(Object, Object)

Object.ReferenceEquals(Object, Object)
Object.GetHashCode()
Object.GetType()
Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.Polarity

Assembly: Santec.Hardware.dll

Syntax

```
public class SimulatedPTM : SimulatedSantecDevice, IPTM, ISantecDevice, IDevice, IDisposable
```

Properties

FailureChance

Declaration

```
public int FailureChance { get; }
```

Property Value

TYPE	DESCRIPTION
Int32	

Name

Get the name of the device.

Declaration

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

Property Value

TYPE	DESCRIPTION
String	The name of the device.

Overrides

SimulatedSantecDevice.Name

Remarks

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

NumberOfChannels

Get the number of channels on the device.

Declaration

```
public int NumberOfChannels { get; }
```

Property Value

TYPE	DESCRIPTION
Int32	The number of channels on the device.

Exceptions

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

PolarityMapping

Declaration

```
public IReadOnlyList<int?> PolarityMapping { get; }
```

Property Value

TYPE	DESCRIPTION
IReadOnlyList<Nullable<Int32>>	

Methods

ConfigureFailureChance(Int32)

Declaration

```
public void ConfigureFailureChance(int failureChance)
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	failureChance	

ConfigureMapping(IEnumerable<Nullable<Int32>>)

Declaration

```
public void ConfigureMapping(IEnumerable<int?> mapping)
```

Parameters

TYPE	NAME	DESCRIPTION
IEnumerable<Nullable<Int32>>	mapping	

MapOutputAsync(Int32, CancellationToken)

Asynchronously map the detector of a specified output channel.

Declaration

```
public async Task<int?> MapOutputAsync(int channel, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	channel	The output channel to map the detector for.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Nullable<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
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device map detector response is not a valid response.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.
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

```
public async Task<IReadOnlyList<int?>> MapPolarityAsync(int numberOfChannels, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	numberOfChannels	The number of output channels to map.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<IReadOnlyList<Nullable<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
ArgumentException	Thrown when the specified number of channels is out of range for the device.
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.
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

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

Parameters

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

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

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

Exceptions

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

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

Parameters

TYPE	NAME	DESCRIPTION
Int32	channel	The output channel to turn on the laser for.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

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

Exceptions

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

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

Parameters

TYPE	NAME	DESCRIPTION
Int32	channel	The output channel to turn on the laser for.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified number of channels is out of range for the device.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device VFL response does not match the expected response.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.
NotSupportedException	Thrown when the operation is not supported by the device.

Implements

IPTM
ISantecDevice
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

Object
CompositeSantecDevice
SimulatedRDP

Implements

IRDP
ICompositeDevice
IPTM
ISantecDevice
IDevice
IDisposable

Inherited Members

CompositeSantecDevice.SerialNumber
CompositeSantecDevice.FirmwareVersion
CompositeSantecDevice.IsInitialized
CompositeSantecDevice.IsInitializing
CompositeSantecDevice.Subdevices
CompositeSantecDevice.Dispose()
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()
Object.ToString()
Object.Equals(Object)
Object.Equals(Object, Object)

Object.ReferenceEquals(Object, Object)
Object.GetHashCode()
Object.GetType()
Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.Polarity

Assembly: Santec.Hardware.dll

Syntax

```
public class SimulatedRDP : CompositeSantecDevice, IRDP, ICompositeDevice, IPTM, ISantecDevice, IDevice,
IDisposable
```

Properties

Address

Get the connection address of the device.

Declaration

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

Property Value

TYPE	DESCRIPTION
String	The connection address of the device.

Overrides

CompositeSantecDevice.Address

FailureChance

Declaration

```
public int FailureChance { get; }
```

Property Value

TYPE	DESCRIPTION
Int32	

Name

Get the name of the device.

Declaration

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

Property Value

TYPE	DESCRIPTION
String	The name of the device.

Overrides

CompositeSantecDevice.Name

Remarks

> **NOTE**
>
> This property will always return "RD-P".

NumberOfChannels

Get the number of channels on the device.

Declaration

```
public int NumberOfChannels { get; }
```

Property Value

TYPE	DESCRIPTION
Int32	The number of channels on the device.

Exceptions

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

PolarityMapping

Declaration

```
public IReadOnlyList<int?> PolarityMapping { get; }
```

Property Value

TYPE	DESCRIPTION
IReadOnlyList<Nullable<Int32>>	

Wavelength

Gets the wavelength of the RD-P.

Declaration

```
public int Wavelength { get; }
```

Property Value

TYPE	DESCRIPTION
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

```
public Task CollectOutputMappingDataAsync(int cable, int fiber, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	cable	The cable to collect the mapping data for.
Int32	fiber	The fiber of the cable to collect the mapping data for.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

The fiber 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
ArgumentException	Thrown when the specified cable is out of range for the polarity mapping operation.
ArgumentException	Thrown when the specified fiber is out of range for the specified cable.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
InvalidOperationException	Thrown when a polarity mapping operation is not in progress.

TYPE	CONDITION
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

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

Parameters

TYPE	NAME	DESCRIPTION
Int32	channel	The output channel to collect the mapping data for.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	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
ArgumentException	Thrown when the specified channel is out of range for the polarity mapping operation.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
InvalidOperationException	Thrown when a polarity mapping operation is not in progress.
InvalidOperationException	Thrown when a segmented polarity mapping operation is not in progress.

ConfigureFailureChance(Int32)

Declaration

```
public void ConfigureFailureChance(int failureChance)
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	failureChance	

ConfigureMapping(IEnumerable<Nullable<Int32>>)

Declaration

```
public void ConfigureMapping(IEnumerable<int?> mapping)
```

Parameters

TYPE	NAME	DESCRIPTION
IEnumerable<Nullable<Int32>>	mapping	

EndMappingPolarityAsync()

Asynchronously end a segmented polarity mapping operation.

Declaration

```
public Task EndMappingPolarityAsync()
```

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized device.
InvalidOperationException	Thrown when a polarity mapping operation is not in progress.
InvalidOperationException	Thrown when a segmented polarity mapping operation is not in progress.

GetLossCompensation()

Get the loss compensation of the RD-P.

Declaration

```
public int GetLossCompensation()
```

Returns

TYPE	DESCRIPTION
Int32	The loss compensation input power in dBm.

Exceptions

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

GetMappingInProgress()

Get whether a polarity mapping operation is in progress.

Declaration

```
public bool GetMappingInProgress()
```

Returns

TYPE	DESCRIPTION
Boolean	`true` if a polarity mapping operation is in progress; otherwise, `false`.

GetMappingMode()

Get the mapping mode of the RD-P.

Declaration

```
public EMappingMode GetMappingMode()
```

Returns

TYPE	DESCRIPTION
EMappingMode	The mapping mode of the RD-P.

Exceptions

TYPE	CONDITION

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

GetMappingSensitivity()

Get the mapping sensitivity of the RD-P.

Declaration

```
public EMappingSensitivity GetMappingSensitivity()
```

Returns

TYPE	DESCRIPTION
EMappingSensitivity	The mapping sensitivity of the RD-P.

Exceptions

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

InitializeAsync()

Asynchronously open the device connection and initialize the device settings.

Declaration

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

Returns

TYPE	DESCRIPTION
Task	The task representing the operation.

Overrides

CompositeSantecDevice.InitializeAsync()

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION

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 NotSupportedException.

Declaration

```
public Task<int?> MapOutputAsync(int channel, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	channel	The output channel to map the detector for.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Nullable<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.

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

Exceptions

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

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

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

```
public Task<IReadOnlyList<int?>> MapPolarityAsync(int numberOfChannels, IEnumerable<int?> expectedMapping,
CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	numberOfChannels	The number of output channels to map.
IEnumerable<Nullable<Int32>>	expectedMapping	The expected mapping result.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<IReadOnlyList<Nullable<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
ArgumentException	Thrown when the specified number of channels is out of range for the device.
InvalidOperationException	Thrown when the method is called on an uninitialized device.

TYPE	CONDITION
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.
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.

MapPolarityAsync(Int32, CancellationToken)

Asynchronously map the polarity for a set of output channels.

Declaration

```
public Task<IReadOnlyList<int?>> MapPolarityAsync(int numberOfChannels, CancellationToken cancellationToken
= default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	numberOfChannels	The number of output channels to map.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<IReadOnlyList<Nullable<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

TYPE	CONDITION
ArgumentException	Thrown when the specified number of channels is out of range for the device.
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.
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

```
public IReadOnlyList<int?> ProcessMappingData(IEnumerable<int?> expectedMapping)
```

Parameters

TYPE	NAME	DESCRIPTION
IEnumerable<Nullable<Int32>>	expectedMapping	Optional: The expected mapping results.

Returns

TYPE	DESCRIPTION
IReadOnlyList<Nullable<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
InvalidOperationException	Thrown when the method is called on an uninitialized device.

TYPE	CONDITION
InvalidOperationException	Thrown when a polarity mapping operation is not in progress.
InvalidOperationException	Thrown when a segmented polarity mapping operation is not in progress.
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

```
public IReadOnlyList<int?> ProcessMappingData(int cable, IEnumerable<int?> expectedMapping)
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	cable	The 1-based index of the cable to process the mapping data for.
IEnumerable<Nullable<Int32>>	expectedMapping	Optional: The expected mapping results.

Returns

TYPE	DESCRIPTION
IReadOnlyList<Nullable<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
ArgumentException	Thrown when the specified cable index is out of range for the polarity mapping operation.

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized device.
InvalidOperationException	Thrown when a polarity mapping operation is not in progress.
InvalidOperationException	Thrown when a segmented polarity mapping operation is not in progress.
InvalidOperationException	Thrown when a multi-cable polarity mapping operation is not in progress.
InvalidOperationException	Thrown when data is missing for one or more outputs.
PolarityMappingFailedException	Thrown when the device cannot determine a mapping of outputs to inputs.

RefreshAsync()

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

Declaration

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

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Overrides

CompositeSantecDevice.RefreshAsync()

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
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.

TYPE	CONDITION
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the subdevice during its refresh operation.

SetLossCompensation(Int32)

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

```
public void SetLossCompensation(int inputPower)
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	inputPower	The input power into the camera.

Exceptions

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

```
public void SetMappingMode(EMappingMode mappingMode)
```

Parameters

TYPE	NAME	DESCRIPTION
EMappingMode	mappingMode	The mapping mode to change the RD-P to.

Exceptions

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

SetMappingSensitivity(EMappingSensitivity)

Set the mapping sensitivity of the RD-P.

Declaration

```
public void SetMappingSensitivity(EMappingSensitivity mappingSensitivity)
```

Parameters

TYPE	NAME	DESCRIPTION
EMappingSensitivity	mappingSensitivity	The mapping sensitivity to change the RD-P to.

Exceptions

TYPE	CONDITION
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

```
public Task StartMappingPolarityAsync(IEnumerable<CableInformation> cables)
```

Parameters

TYPE	NAME	DESCRIPTION
IEnumerable<CableInformation>	cables	The cable information of the cables.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

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

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified number of fibers for a cable is out of range for the device.
ArgumentException	Thrown when the specified cable offset for a cable is less then 0.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
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

```
public Task StartMappingPolarityAsync(int numberOfChannels)
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	numberOfChannels	The number of output channels to map.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

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

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified number of channels is out of range for the device.

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized device.
InvalidOperationException	Thrown when a polarity mapping operation is already in progress.

TurnOffLaserAsync(CancellationToken)

> **⚠ CAUTION**
>
> This operation is not supported for RDP devices. This operation will always throw a NotSupportedException.

Declaration

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

Parameters

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

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

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

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device laser response does not match the expected response.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.
NotSupportedException	Thrown when the operation is not supported by the device.

TurnOnFlashingVFLModeAsync(Int32, CancellationToken)

> **⚠ CAUTION**
>
> This operation is not supported for RDP devices. This operation will always throw a NotSupportedException.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
Int32	channel	The output channel to turn on the laser for.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

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

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified number of channels is out of range for the device.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device flashing VFL response does not match the expected response.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.
NotSupportedException	Thrown when the operation is not supported by the device.

TurnOnVFLModeAsync(Int32, CancellationToken)

> **⚠ CAUTION**
>
> This operation is not supported for RDP devices. This operation will always throw a NotSupportedException.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
Int32	channel	The output channel to turn on the laser for.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

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

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified number of channels is out of range for the device.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device VFL response does not match the expected response.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.
NotSupportedException	Thrown when the operation is not supported by the device.

Implements

IRDP
ICompositeDevice

IPTM
ISantecDevice
IDevice
System.IDisposable

Class SimulatedRDPFactory

Provides a factory for creating simulated RD-P devices.

> **NOTE**
>
> This class is intended for testing/development purposes.

Inheritance

Object

SimulatedRDPFactory

Implements

IRDPFactory

Inherited Members

Object.ToString()
Object.Equals(Object)
Object.Equals(Object, Object)
Object.ReferenceEquals(Object, Object)
Object.GetHashCode()
Object.GetType()
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
Boolean	`true` if the RD-P factory will always create RD-SPs; otherwise, `false`.

Remarks

Use ConfigureRDSPCreation(Boolean) to set this value.

Methods

CheckForCamera()

Check for an RD-P camera on the system.

Declaration

```
public bool CheckForCamera()
```

Returns

TYPE	DESCRIPTION
Boolean	`true` if a RD-P camera is found; otherwise, `false`. The default value is `true`.

Remarks

Use ConfigureCanDetectCamera(Boolean) to set this value.

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.

Declaration

```
public bool CheckIfRLMMeetsHardwareRequirements(IRLM RLM)
```

Parameters

TYPE	NAME	DESCRIPTION
IRLM	RLM	

Returns

TYPE	DESCRIPTION
Boolean	`true` if the RLM meets the hardware requirements; otherwise, `false`.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified RLM is uninitialized.

ConfigureCanDetectCamera(Boolean)

Configure value returned by CheckForCamera().

Declaration

```
public void ConfigureCanDetectCamera(bool canDetectCamera)
```

Parameters

TYPE	NAME	DESCRIPTION
Boolean	canDetectCamera	Whether the factory can detect the RD-P camera.

ConfigureRDSPCreation(Boolean)

Configure value returned by AlwaysCreateRDSP.

Declaration

```
public void ConfigureRDSPCreation(bool alwaysCreateRDSP)
```

Parameters

TYPE	NAME	DESCRIPTION
Boolean	alwaysCreateRDSP	Whether the factory will always create RD-SPs from RLMs.

CreateRDP(IPTM)

Creates a new RD-P with the specified PTM.

> **NOTE**
>
> PTM must be initialized and have a camera connected.

Declaration

```
public 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.
ArgumentException	Thrown when the specified PTM is uninitialized.
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

```
public 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.
ArgumentException	Thrown when the specified RLM is uninitialized.
ArgumentException	Thrown when the specified RLM does not meet the RD-P hardware requirements.

Implements

IRDPFactory

Class SimulatedRDSP

Inheritance

Object
CompositeSantecDevice
SimulatedRDSP

Implements

IRDSP
IRDP
ICompositeDevice
IRLM
IDiscretePowerMeter
IPowerMeter
IDetectorDevice
ILaserSource
IWavelengthDevice
ISwitchController
IFiberDevice
IPTM
ISantecDevice
IDevice
IDisposable

Inherited Members

CompositeSantecDevice.SerialNumber
CompositeSantecDevice.FirmwareVersion
CompositeSantecDevice.IsInitialized
CompositeSantecDevice.IsInitializing
CompositeSantecDevice.Subdevices
CompositeSantecDevice.Dispose()
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.QueryAsync(String, CancellationToken)
CompositeSantecDevice.WriteAsync(String, CancellationToken)
CompositeSantecDevice.ReadAsync(CancellationToken)
CompositeSantecDevice.Uninitialize()
Object.ToString()
Object.Equals(Object)
Object.Equals(Object, Object)
Object.ReferenceEquals(Object, Object)
Object.GetHashCode()
Object.GetType()
Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.Polarity

Assembly: Santec.Hardware.dll

Syntax

```
public class SimulatedRDSP : CompositeSantecDevice, IRDSP, IRDP, ICompositeDevice, IRLM,
IDiscretePowerMeter, IPowerMeter, IDetectorDevice, ILaserSource, IWavelengthDevice, ISwitchController,
IFiberDevice, IPTM, ISantecDevice, IDevice, IDisposable
```

Properties

Address

Get the connection address of the device.

Declaration

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

Property Value

TYPE	DESCRIPTION
String	The connection address of the device.

Overrides

CompositeSantecDevice.Address

Detectors

Get the detectors connected to the device.

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

Declaration

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

Property Value

TYPE	DESCRIPTION
IReadOnlyList<DeviceDetector>	The list of the detectors connected to the device.

Exceptions

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

FailureChance

Declaration

```
public int FailureChance { get; }
```

Property Value

TYPE	DESCRIPTION
Int32	

Fiber

Get the fiber information of the device.

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

Declaration

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

Property Value

TYPE	DESCRIPTION
FiberInformation	The fiber information of the device.

Exceptions

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

MultiCableMappingInProgress

Declaration

```
public bool MultiCableMappingInProgress { get; }
```

Property Value

TYPE	DESCRIPTION
Boolean	

Name

Get the name of the device.

Declaration

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

Property Value

TYPE	DESCRIPTION
String	The name of the device.

Overrides

CompositeSantecDevice.Name

Remarks

> **NOTE**
>
> This property will always return "RD-P".

NumberOfChannels

Get the number of channels on the device.

Declaration

```
public int NumberOfChannels { get; }
```

Property Value

TYPE	DESCRIPTION
Int32	The number of channels on the device.

Exceptions

TYPE	CONDITION
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

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

Property Value

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

Exceptions

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

NumberOfOutputs

Get the number of laser outputs.

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

Declaration

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

Property Value

TYPE	DESCRIPTION
Int32	The number of device outputs.

Exceptions

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

PolarityMapping

Declaration

```
public IReadOnlyList<int?> PolarityMapping { get; }
```

Property Value

TYPE	DESCRIPTION
IReadOnlyList<Nullable<Int32>>	

SegmentedMappingInProgress

Declaration

```
public bool SegmentedMappingInProgress { get; }
```

Property Value

TYPE	DESCRIPTION
Boolean	

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

```
public bool SupportsILResolution { get; }
```

Property Value

TYPE	DESCRIPTION
Boolean	`true` if the RLM supports changing the IL resolution; otherwise, `false`.

Exceptions

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

Switches

Get the switches connected to the device.

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

Declaration

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

Property Value

TYPE	DESCRIPTION
List<DeviceSwitch>	The list of the switches connected to the device.

Exceptions

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

Wavelength

Gets the wavelength of the RD-P.

Declaration

```
public int Wavelength { get; }
```

Property Value

TYPE	DESCRIPTION
Int32	The RD-P wavelength.

Wavelengths

Get the wavelengths supported by the device.

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

Declaration

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

Property Value

TYPE	DESCRIPTION
List<Int32>	A list of the device's wavelengths.

Exceptions

TYPE	CONDITION
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

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

Parameters

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

Returns

TYPE	DESCRIPTION
Task	The task representing the operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device has no switch at any of the specified switch indexes.
ArgumentException	Thrown when any the specified channels is out of range for the corresponding device switch.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

ChangeOutputChannelAsync(Int32, CancellationToken)

Asynchronously change the output channel of the laser source.

Declaration

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

Parameters

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

Returns

TYPE	DESCRIPTION
Task	The task representing the operation.

Remarks

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

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified output channel is out of range for the laser source.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the RLM.

ChangeSwitchChannelAsync(Int32, Int32, CancellationToken)

Asynchronously change the channel of a connected switch.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
Int32	switchIndex	The switch number of the switch. `0` for the internal switch; `1` or `2` for external switches attached to the device.

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

Returns

TYPE	DESCRIPTION
Task	The task representing the operation.

Remarks

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

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device has no switch at the specified switch index.
ArgumentException	Thrown when the specified channel is out of range for the device switch.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

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

```
public Task CollectOutputMappingDataAsync(int cable, int fiber, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	cable	The cable to collect the mapping data for.
Int32	fiber	The fiber of the cable to collect the mapping data for.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

The fiber 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
ArgumentException	Thrown when the specified cable is out of range for the polarity mapping operation.
ArgumentException	Thrown when the specified fiber is out of range for the specified cable.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
InvalidOperationException	Thrown when a polarity mapping operation is not in progress.
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

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

Parameters

TYPE	NAME	DESCRIPTION
Int32	channel	The output channel to collect the mapping data for.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	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
ArgumentException	Thrown when the specified channel is out of range for the polarity mapping operation.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
InvalidOperationException	Thrown when a polarity mapping operation is not in progress.
InvalidOperationException	Thrown when a segmented polarity mapping operation is not in progress.

ConfigureFailureChance(Int32)

Declaration

```
public void ConfigureFailureChance(int failureChance)
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	failureChance	

ConfigureMapping(IEnumerable<Nullable<Int32>>)

Declaration

```
public void ConfigureMapping(IEnumerable<int?> mapping)
```

Parameters

TYPE	NAME	DESCRIPTION
IEnumerable<Nullable<Int32>>	mapping	

EndMappingPolarityAsync()

Asynchronously end a segmented polarity mapping operation.

Declaration

```
public Task EndMappingPolarityAsync()
```

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

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

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized device.
InvalidOperationException	Thrown when a polarity mapping operation is not in progress.
InvalidOperationException	Thrown when a segmented polarity mapping operation is not in progress.

GetActiveLaserAsync(CancellationToken)

Asynchronously get the active laser.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION

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

Returns

TYPE	DESCRIPTION
Task<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
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

GetDUTILAsync(CancellationToken)

Asynchronously get the DUT IL.

Declaration

```
public Task<double> GetDUTILAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

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

Returns

TYPE	DESCRIPTION
Task<Double>	The task representing the operation.

Remarks

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

Exceptions

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

GetFactoryPowerAsync(Int32, Int32, CancellationToken)

Asynchronously get the factory power for the specified output channel and wavelength.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
Int32	output	The output channel to get the factory power for.
Int32	wavelength	The wavelength to get the factory power for.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

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

Remarks

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

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified output channel is out of range.
ArgumentException	Thrown when the device does not support the specified wavelength.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device factory power response is not a valid number.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

GetGainModeAsync(CancellationToken)

Asynchronously get the gain mode.

Declaration

```
public Task<EGainMode> GetGainModeAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

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

Returns

TYPE	DESCRIPTION
Task<EGainMode>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized RLM.
UnexpectedDeviceResponseException	Thrown when the device response does not correspond to a gain mode.

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

GetILReferenceAsync(Int32, Int32, CancellationToken)

Asynchronously get the stored IL reference for a detector at a given wavelength.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	The detector to get the reference for.
Int32	wavelength	The wavelength to get the reference for.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

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

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
ArgumentException	Thrown when the specified detector is out of range for the device.
InvalidOperationException	Thrown when the method is called on an uninitialized device.

TYPE	CONDITION
UnexpectedDeviceResponseException	Thrown when the device IL reference query response is not a valid number.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

GetILResolutionAsync(CancellationToken)

Asynchronously get the IL reading resolution.

Declaration

```
public Task<EResolution> GetILResolutionAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

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

Returns

TYPE	DESCRIPTION
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
InvalidOperationException	Thrown when the method is called on an uninitialized device.
InvalidOperationException	Thrown when the RLM does not support IL resolution operations.
UnexpectedDeviceResponseException	Thrown when the device response does not correspond to a resolution.

TYPE	CONDITION
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

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

Parameters

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

Returns

TYPE	DESCRIPTION
Task<EInteractionMode>	The task representing the asynchronous operation.

Overrides

CompositeSantecDevice.GetInteractionModeAsync(CancellationToken)

Remarks

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

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device response does not correspond to an interaction mode.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

GetLengthModeAsync(CancellationToken)

Asynchronously get the length mode.

Declaration

```
public Task<ELengthMode> GetLengthModeAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

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

Returns

TYPE	DESCRIPTION
Task<ELengthMode>	The task representing the asynchronous operation.

Remarks

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

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized RLM.
UnexpectedDeviceResponseException	Thrown when the device response does not correspond to a length mode.
TimeoutException	Thrown when a communication failure causes no response to be received from the RLM.

GetLossCompensation()

Get the loss compensation of the RD-P.

Declaration

```
public int GetLossCompensation()
```

Returns

TYPE	DESCRIPTION
Int32	The loss compensation input power in dBm.

Exceptions

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

GetMappingInProgress()

Get whether a polarity mapping operation is in progress.

Declaration

```
public bool GetMappingInProgress()
```

Returns

TYPE	DESCRIPTION
Boolean	`true` if a polarity mapping operation is in progress; otherwise, `false`.

GetMappingMode()

Get the mapping mode of the RD-P.

Declaration

```
public EMappingMode GetMappingMode()
```

Returns

TYPE	DESCRIPTION
EMappingMode	The mapping mode of the RD-P.

Exceptions

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

GetMappingSensitivity()

Get the mapping sensitivity of the RD-P.

Declaration

```
public EMappingSensitivity GetMappingSensitivity()
```

Returns

TYPE	DESCRIPTION
EMappingSensitivity	The mapping sensitivity of the RD-P.

Exceptions

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

GetOutputChannelAsync(CancellationToken)

Asynchronously get the current output channel of the laser source.

Declaration

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

Parameters

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

Returns

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

Remarks

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

Exceptions

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

GetPositionBModeAsync(CancellationToken)

Asynchronously get the RL position B mode.

Declaration

```
public Task<EPositionBMode> GetPositionBModeAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

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

Returns

TYPE	DESCRIPTION
Task<EPositionBMode>	The task representing the asynchronous operation.

Remarks

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

Exceptions

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

GetRLModeAsync(CancellationToken)

Asynchronously get the RL sensitivity mode.

Declaration

```
public Task<ERLMode> GetRLModeAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

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

Returns

TYPE	DESCRIPTION
Task<ERLMode>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized RLM.
UnexpectedDeviceResponseException	Thrown when the device response does not correspond to an RL mode.
TimeoutException	Thrown when a communication failure causes no response to be received from the RLM.

GetRLReferenceAsync(CancellationToken)

Asynchronously get the stored RL reference.

Declaration

```
public Task<double> GetRLReferenceAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

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

Returns

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

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
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.

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

GetSwitchChannelAsync(Int32, CancellationToken)

Asynchronously get the channel of a connected switch.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
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.
CancellationToken	cancellationToken	

Returns

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

Remarks

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

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device has no switch at the specified switch index.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

GetTraceDiagnosticsAsync(CancellationToken)

Asynchronously get the trace diagnostics for the last RL measurement.

Declaration

```
public Task<TraceDiagnostics> GetTraceDiagnosticsAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

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

Returns

TYPE	DESCRIPTION
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
InvalidOperationException	Thrown when the method is called on an uninitialized RLM.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the RLM.

TYPE	CONDITION
NotSupportedException	Thrown when the method is called on an RLM through a USB connection.

InitializeAsync()

Asynchronously open the device connection and initialize the device settings.

Declaration

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

Returns

TYPE	DESCRIPTION
Task	The task representing the operation.

Overrides

CompositeSantecDevice.InitializeAsync()

Remarks

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

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 NotSupportedException.

Declaration

```
public Task<int?> MapOutputAsync(int channel, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	channel	The output channel to map the detector for.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Nullable<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
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device map detector response is not a valid response.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.
NotSupportedException	Thrown when the operation is not supported by the device.

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

```
public Task<IReadOnlyList<int?>> MapPolarityAsync(int numberOfChannels, IEnumerable<int?> expectedMapping,
CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	numberOfChannels	The number of output channels to map.
IEnumerable<Nullable<Int32>>	expectedMapping	The expected mapping result.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<IReadOnlyList<Nullable<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
ArgumentException	Thrown when the specified number of channels is out of range for the device.
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.
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.

MapPolarityAsync(Int32, CancellationToken)

Asynchronously map the polarity for a set of output channels.

Declaration

```
public Task<IReadOnlyList<int?>> MapPolarityAsync(int numberOfChannels, CancellationToken cancellationToken
= default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	numberOfChannels	The number of output channels to map.

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

Returns

TYPE	DESCRIPTION
Task<IReadOnlyList<Nullable<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
ArgumentException	Thrown when the specified number of channels is out of range for the device.
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.
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

```
public IReadOnlyList<int?> ProcessMappingData(IEnumerable<int?> expectedMapping = null)
```

Parameters

TYPE	NAME	DESCRIPTION
IEnumerable<Nullable<Int32>>	expectedMapping	Optional: The expected mapping results.

Returns

TYPE	DESCRIPTION
IReadOnlyList<Nullable<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
InvalidOperationException	Thrown when the method is called on an uninitialized device.
InvalidOperationException	Thrown when a polarity mapping operation is not in progress.
InvalidOperationException	Thrown when a segmented polarity mapping operation is not in progress.
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

```
public IReadOnlyList<int?> ProcessMappingData(int cable, IEnumerable<int?> expectedMapping = null)
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	cable	The 1-based index of the cable to process the mapping data for.

TYPE	NAME	DESCRIPTION
IEnumerable<Nullable<Int32>>	expectedMapping	Optional: The expected mapping results.

Returns

TYPE	DESCRIPTION
IReadOnlyList<Nullable<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
ArgumentException	Thrown when the specified cable index is out of range for the polarity mapping operation.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
InvalidOperationException	Thrown when a polarity mapping operation is not in progress.
InvalidOperationException	Thrown when a segmented polarity mapping operation is not in progress.
InvalidOperationException	Thrown when a multi-cable polarity mapping operation is not in progress.
InvalidOperationException	Thrown when data is missing for one or more outputs.
PolarityMappingFailedException	Thrown when the device cannot determine a mapping of outputs to inputs.

ReadILAsync(Int32, Int32, CancellationToken)

Asynchronously read the insertion loss from a detector at a given wavelength.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	The detector number of the detector to read the insertion loss from.
Int32	wavelength	The wavelength to read the insertion loss at.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

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

Remarks

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

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
ArgumentException	Thrown when the specified detector is out of range for the device.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device read IL response is not a valid number.
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

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

Parameters

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

Returns

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

Remarks

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

Exceptions

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

ReadMultipleILsAsync(IEnumerable<Int32>, Int32, CancellationToken)

Asynchronously read the insertion loss from multiple detectors at a given wavelength.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
IEnumerable<Int32>	detectors	The detectors to read the insertion loss from.

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

Returns

TYPE	DESCRIPTION
Task<List<Double>>	The task representing the asynchronous operation.

Remarks

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

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
ArgumentException	Thrown when any of the specified detectors are out of range for the device.
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.
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

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

Parameters

TYPE	NAME	DESCRIPTION
IEnumerable<Int32>	detectors	The detectors to read from.
Int32	wavelength	The wavelength to read the power at.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<List<Double>>	The task representing the asynchronous operation.

Remarks

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

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
ArgumentException	Thrown when any of the specified detectors are out of range for the device.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

ReadPowerAsync(Int32, Int32, CancellationToken)

Asynchronously read the power from an device detector at a given wavelength.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	The detector number of the detector to read from.
Int32	wavelength	The wavelength to read the power at.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

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

Remarks

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

Exceptions

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

ReadRLAsync(Int32, Double, Double, CancellationToken)

Asynchronously read the return loss at a given wavelength using the specified reference lengths.

Declaration

```
public Task<RLReading> ReadRLAsync(int wavelength, double lengthReferenceA, double lengthReferenceB,
CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	wavelength	The wavelength to read the return loss at.
Double	lengthReferenceA	The reference length to use for position A.
Double	lengthReferenceB	The reference length to use for position B.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
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
ArgumentException	Thrown when the RLM does not support the specified wavelength.
ArgumentException	Thrown when either RL reference length is less than `0`.
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.

TYPE	CONDITION
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

```
public Task<RLReading> ReadRLAsync(int wavelength, double lengthReference, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	wavelength	The wavelength to read the return loss at.
Double	lengthReference	The reference length to use for the reading.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
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
ArgumentException	Thrown when the RLM does not support the specified wavelength.
ArgumentException	Thrown when the RLM reference length is less than `0`.

TYPE	CONDITION
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.
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

```
public Task<RLReading> ReadRLAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

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

Returns

TYPE	DESCRIPTION
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

TYPE	CONDITION
ArgumentException	Thrown when the RLM does not support the specified wavelength.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the RLM.

ReferenceILAsync(Int32, Int32, Boolean, CancellationToken)

Asynchronously perform an insertion loss reference for a detector at a given wavelength.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	The detector number of the detector to perform the reference for.
Int32	wavelength	The wavelength to perform the reference at.
Boolean	applyReferenceToAllDetectors	Indicate if the reference read should be applied to all detectors for the current channel.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

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

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
ArgumentException	Thrown when the specified detector is out of range for the device.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device reference IL response is not a valid number.
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

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

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	The detector number of the detector to perform the reference for.
Int32	wavelength	The wavelength to perform the reference at.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

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

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
ArgumentException	Thrown when the specified detector is out of range for the device.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device reference IL response is not a valid number.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

ReferenceRLAsync(CancellationToken)

Asynchronously perform a return loss length reference.

Declaration

```
public Task<double> ReferenceRLAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

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

Returns

TYPE	DESCRIPTION
Task<Double>	The task representing the operation.

Remarks

NOTE

This operation will not block.

Exceptions

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

TYPE	CONDITION
UnexpectedDeviceResponseException	Thrown when the device reference RL response is not a valid number.
TimeoutException	Thrown when a communication failure causes no response to be received from the RLM.

RefreshAsync()

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

Declaration

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

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Overrides

CompositeSantecDevice.RefreshAsync()

Remarks

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

Exceptions

TYPE	CONDITION
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.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the RLM subdevice during its refresh operation.

SetDUTILAsync(Double, CancellationToken)

Asynchronously set the DUT IL.

Declaration

```
public Task SetDUTILAsync(double il, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Double	il	The DUT IL value.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
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
ArgumentException	Thrown when the specified DUT IL is less than `0`.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the RLM.

SetGainModeAsync(EGainMode, CancellationToken)

Asynchronously set the gain mode.

Declaration

```
public Task SetGainModeAsync(EGainMode gainMode, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
EGainMode	gainMode	The gain mode to change to.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

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

Exceptions

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

SetILReferenceAsync(Int32, Int32, Double, CancellationToken)

Asynchronously set the IL reference for a detector at a given wavelength.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	The detector to set the reference for.
Int32	wavelength	The wavelength to set the reference for.

TYPE	NAME	DESCRIPTION
Double	reference	The IL reference value.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
ArgumentException	Thrown when the specified detector is out of range for the device.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

SetILResolutionAsync(EResolution, CancellationToken)

Asynchronously set the IL reading resolution.

Declaration

```
public Task SetILResolutionAsync(EResolution resolution, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
EResolution	resolution	The resolution to set the detector to.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
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
ArgumentException	Thrown when the IL resolution is not supported by the RLM.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
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.
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

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

Parameters

TYPE	NAME	DESCRIPTION
EInteractionMode	interactionMode	The interaction mode to set the device to.

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

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Overrides

CompositeSantecDevice.SetInteractionModeAsync(EInteractionMode, CancellationToken)

Remarks

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

Exceptions

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

SetLengthModeAsync(ELengthMode, CancellationToken)

Asynchronously set the length mode.

Declaration

```
public Task SetLengthModeAsync(ELengthMode lengthMode, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
ELengthMode	lengthMode	The length mode to change to.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

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

Exceptions

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

SetLossCompensation(Int32)

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

```
public void SetLossCompensation(int inputPower)
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	inputPower	The input power into the camera.

Exceptions

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

TYPE	CONDITION
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

```
public void SetMappingMode(EMappingMode mappingMode)
```

Parameters

TYPE	NAME	DESCRIPTION
EMappingMode	mappingMode	The mapping mode to change the RD-P to.

Exceptions

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

SetMappingSensitivity(EMappingSensitivity)

Set the mapping sensitivity of the RD-P.

Declaration

```
public void SetMappingSensitivity(EMappingSensitivity mappingSensitivity)
```

Parameters

TYPE	NAME	DESCRIPTION
EMappingSensitivity	mappingSensitivity	The mapping sensitivity to change the RD-P to.

Exceptions

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

SetPositionBModeAsync(EPositionBMode, CancellationToken)

Asynchronously set the RL position B mode.

Declaration

```
public Task SetPositionBModeAsync(EPositionBMode positionBMode, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
EPositionBMode	positionBMode	The position B mode to change to.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

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

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized RLM.
UnexpectedDeviceResponseException	Thrown when the device positionB mode response does not match the expected response.
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

```
public Task SetRLModeAsync(ERLMode rlMode, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
ERLMode	rlMode	The RL mode to change to.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized RLM.
UnexpectedDeviceResponseException	Thrown when the device RL mode response does not match the expected response.
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

```
public Task SetRLReferenceAsync(double reference, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Double	reference	The RL length reference value.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
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
ArgumentException	Thrown when the RL length reference value is less than `0`.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the RLM.

StartMappingPolarityAsync(IEnumerable<CableInformation>)

Asynchronously start a multi-cable segmented polarity mapping operation for a set of cables.

Declaration

```
public Task StartMappingPolarityAsync(IEnumerable<CableInformation> cables)
```

Parameters

TYPE	NAME	DESCRIPTION
IEnumerable<CableInformation>	cables	The cable information of the cables.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified number of fibers for a cable is out of range for the device.
ArgumentException	Thrown when the specified cable offset for a cable is less then 0.

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

```
public Task StartMappingPolarityAsync(int numberOfChannels)
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	numberOfChannels	The number of output channels to map.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified number of channels is out of range for the device.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
InvalidOperationException	Thrown when a polarity mapping operation is already in progress.

TurnOffLaserAsync(CancellationToken)

CAUTION

This operation is not supported for RDP devices. This operation will always throw a NotSupportedException.

Declaration

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

Parameters

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

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device laser response does not match the expected response.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.
NotSupportedException	Thrown when the operation is not supported by the device.

TurnOnFlashingVFLModeAsync(Int32, CancellationToken)

CAUTION

This operation is not supported for RDP devices. This operation will always throw a NotSupportedException.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
Int32	channel	The output channel to turn on the laser for.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

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

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified number of channels is out of range for the device.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device flashing VFL response does not match the expected response.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.
NotSupportedException	Thrown when the operation is not supported by the device.

TurnOnLaserAsync(Int32, CancellationToken)

Asynchronously turn on the specified laser.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION

TYPE	NAME	DESCRIPTION
Int32	wavelength	The wavelength of the laser to turn on.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

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

TurnOnVFLModeAsync(Int32, CancellationToken)

CAUTION

This operation is not supported for RDP devices. This operation will always throw a NotSupportedException.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
Int32	channel	The output channel to turn on the laser for.

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

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified number of channels is out of range for the device.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device VFL response does not match the expected response.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.
NotSupportedException	Thrown when the operation is not supported by the device.

Implements

IRDSP
IRDP
ICompositeDevice
IRLM
IDiscretePowerMeter
IPowerMeter
IDetectorDevice
ILaserSource
IWavelengthDevice
ISwitchController
IFiberDevice
IPTM
ISantecDevice
IDevice
System.IDisposable

Namespace Santec.Hardware.Devices.Polarization

Classes

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.

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.

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

IPLM

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

Enums

EPolarizationMode

Specifies the polarization mode of a PLM.

EPolarizationState

Specifies the polarization state of a PLM.

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 PLM.

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.
Vertical	Vertical polarization state.

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)
IPowerMeter.GetILReferenceAsync(Int32, Int32, CancellationToken)
IPowerMeter.SetILReferenceAsync(Int32, Int32, Double, CancellationToken)
IPowerMeter.ReferenceILAsync(Int32, Int32, CancellationToken)
IPowerMeter.ReferenceILAsync(Int32, Int32, Boolean, CancellationToken)
IPowerMeter.ReadILAsync(Int32, Int32, CancellationToken)
ILaserSource.NumberOfOutputs
ILaserSource.GetActiveLaserAsync(CancellationToken)
ILaserSource.TurnOnLaserAsync(Int32, CancellationToken)
ILaserSource.TurnOffLaserAsync(CancellationToken)
ILaserSource.GetOutputChannelAsync(CancellationToken)
ILaserSource.ChangeOutputChannelAsync(Int32, CancellationToken)
ILaserSource.ReadMonitorPowerAsync(Int32, CancellationToken)
ILaserSource.GetFactoryPowerAsync(Int32, 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.GetDetectorDarkAsync(Int32, CancellationToken)
IDetectorDevice.NumberOfDetectors
IDetectorDevice.Detectors
IDevice.Name
IDevice.SerialNumber
IDevice.FirmwareVersion
IDevice.Address
IDevice.IsInitialized

IDevice.InitializeAsync()
IDevice.Uninitialize()
IDevice.ResetAsync()
IDevice.RefreshAsync()
IDevice.PingAsync()
IDevice.QueryAsync(String, CancellationToken)
IDevice.WriteAsync(String, CancellationToken)
IDevice.ReadAsync(CancellationToken)
IDisposable.Dispose()
IFiberDevice.Fiber

Namespace: Santec.Hardware.Devices.Polarization

Assembly: Santec.Hardware.dll

Syntax

```
public interface IPLM : ISantecDevice, IDiscretePowerMeter, IPowerMeter, ILaserSource, ISwitchController,
IBackreflectionMeter, IWavelengthDevice, IDetectorDeviceWithDarking, IDetectorDevice, IDevice, IDisposable,
IFiberDevice
```

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
Int32	detector	The detector to get the reference values for.
Int32	wavelength	The wavelength to get the reference values for.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<List<Double>>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the PLM does not support the specified wavelength.
ArgumentException	Thrown when the specified detector is out of range for the PLM.
InvalidOperationException	Thrown when the method is called on an uninitialized PLM.
UnexpectedDeviceResponseException	Thrown when the PLM PDL references query response is not a valid list of values.
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
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<EPolarizationMode>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

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

TYPE	CONDITION
UnexpectedDeviceResponseException	Thrown when the PLM response does not correspond to a polarization mode.
TimeoutException	Thrown when a communication failure causes no response to be received from the PLM.

GetPolarizationStateAsync(CancellationToken)

Asynchronously get the polarization state.

Declaration

```
Task<EPolarizationState> GetPolarizationStateAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

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

Returns

TYPE	DESCRIPTION
Task<EPolarizationState>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized PLM.
UnexpectedDeviceResponseException	Thrown when the PLM response does not correspond to a polarization state.
TimeoutException	Thrown when a communication failure causes no response to be received from the PLM.

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
Int32	waveplateIndex	The waveplate to get the position of. `0` or `1` to specify which waveplate.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

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

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified waveplate is out of range for the device.
InvalidOperationException	Thrown when the method is called on an uninitialized PLM.
UnexpectedDeviceResponseException	Thrown when the PLM waveplate position response is not a valid waveplate position.
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
Int32	detector	The detector to read the polarization dependent loss for.

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

Returns

TYPE	DESCRIPTION
Task<PDLAndBRReading>	The task representing the asynchronous operation.

Remarks

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

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the PLM does not support the specified wavelength.
ArgumentException	Thrown when the specified detector is out of range for the PLM.
InvalidOperationException	Thrown when the method is called on an uninitialized PLM.
UnexpectedDeviceResponseException	Thrown when the PLM read PDL and BR response is not a valid set of values.
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

TYPE	NAME	DESCRIPTION
Int32	detector	The detector to read the polarization dependent loss for.
Int32	wavelength	The wavelength to read the polarization dependent loss at.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<PDLReading>	The task representing the asynchronous operation.

Remarks

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

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the PLM does not support the specified wavelength.
ArgumentException	Thrown when the specified detector is out of range for the PLM.
InvalidOperationException	Thrown when the method is called on an uninitialized PLM.
UnexpectedDeviceResponseException	Thrown when the PLM read PDL response is not a valid set of values.
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
Int32	detector	The detector to perform the reference for.
Int32	wavelength	The wavelength to perform the reference at.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<List<Double>>	The task representing the operation.

Remarks

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

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the PLM does not support the specified wavelength.
ArgumentException	Thrown when the specified detector is out of range for the PLM.
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.
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.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

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

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.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

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

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized PLM.
UnexpectedDeviceResponseException	Thrown when the PLM polarization state response does not match the expected response.
TimeoutException	Thrown when a communication failure causes no response to be received from the PLM.

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
Int32	waveplateIndex	The waveplate to set the position of. `0` or `1` to specify which waveplate.
Double	position	The position to set the waveplate to. Between `0` and `360` degrees.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Exceptions

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

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

Object
PDLReading
PDLAndBRReading

Inherited Members

PDLReading.ILAverage
PDLReading.PDL
Object.ToString()
Object.Equals(Object)
Object.Equals(Object, Object)
Object.ReferenceEquals(Object, Object)
Object.GetHashCode()
Object.GetType()
Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.Polarization

Assembly: Santec.Hardware.dll

Syntax

```
public class PDLAndBRReading : PDLReading
```

Constructors

PDLAndBRReading(Double, Double, Double)

Initialize a new PDL and BR reading result.

Declaration

```
public PDLAndBRReading(double ilAverage, double pdl, double br)
```

Parameters

TYPE	NAME	DESCRIPTION
Double	ilAverage	The insertion loss average of the fiber.
Double	pdl	The polarization dependent loss of the fiber.
Double	br	The backreflection of the fiber.

Properties

BR

Get the backreflection of the fiber.

Declaration

```
public double BR { get; }
```

Property Value

TYPE	DESCRIPTION
Double	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

Object
PDLReading
PDLAndBRReading

Inherited Members

Object.ToString()
Object.Equals(Object)
Object.Equals(Object, Object)
Object.ReferenceEquals(Object, Object)
Object.GetHashCode()
Object.GetType()
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
Double	ilAverage	The insertion loss average of the fiber.
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
Double	The insertion loss average.

PDL

Get the polarization dependent loss of the fiber.

Declaration

```
public double PDL { get; }
```

Property Value

TYPE	DESCRIPTION
Double	The polarization dependent loss.

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

Object
PhysicalDevice
SantecDevice
PLM

Implements

IPLM
ISantecDevice
IDiscretePowerMeter
IPowerMeter
ILaserSource
ISwitchController
IBackreflectionMeter
IWavelengthDevice
IDetectorDeviceWithDarking
IDetectorDevice
IDevice
IDisposable
IFiberDevice

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.Dispose()
PhysicalDevice.PingAsync()
PhysicalDevice.ResetAsync()

PhysicalDevice.QueryAsync(String, CancellationToken)
PhysicalDevice.WriteAsync(String, CancellationToken)
PhysicalDevice.ReadAsync(CancellationToken)
Object.ToString()
Object.Equals(Object)
Object.Equals(Object, Object)
Object.ReferenceEquals(Object, Object)
Object.GetHashCode()
Object.GetType()
Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.Polarization

Assembly: Santec.Hardware.dll

Syntax

```
public class PLM : SantecDevice, IPLM, ISantecDevice, IDiscretePowerMeter, IPowerMeter, ILaserSource,
ISwitchController, IBackreflectionMeter, IWavelengthDevice, IDetectorDeviceWithDarking, IDetectorDevice,
IDevice, IDisposable, IFiberDevice
```

Properties

Detectors

Get the detectors connected to the device.

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

Declaration

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

Property Value

TYPE	DESCRIPTION
IReadOnlyList<DeviceDetector>	The list of the detectors connected to the device.

Exceptions

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

Fiber

Get the fiber information of the device.

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

Declaration

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

Property Value

TYPE	DESCRIPTION
FiberInformation	The fiber information of the device.

Exceptions

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

Name

Get the name of the device.

Declaration

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

Property Value

TYPE	DESCRIPTION
String	The name of the device.

Overrides

PhysicalDevice.Name

Remarks

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

NumberOfDetectors

Get the number of detectors connected to the device.

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

Declaration

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

Property Value

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

Exceptions

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

NumberOfOutputs

Get the number of laser outputs.

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

Declaration

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

Property Value

TYPE	DESCRIPTION
Int32	The number of device outputs.

Exceptions

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

Switches

Get the switches connected to the device.

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

Declaration

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

Property Value

TYPE	DESCRIPTION

TYPE	DESCRIPTION
List<DeviceSwitch>	The list of the switches connected to the device.

Exceptions

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

Wavelengths

Get the wavelengths supported by the device.

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

Declaration

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

Property Value

TYPE	DESCRIPTION
List<Int32>	A list of the device's wavelengths.

Exceptions

TYPE	CONDITION
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

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

Parameters

TYPE	NAME	DESCRIPTION

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

Returns

TYPE	DESCRIPTION
Task	The task representing the operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device has no switch at any of the specified switch indexes.
ArgumentException	Thrown when any the specified channels is out of range for the corresponding device switch.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

ChangeOutputChannelAsync(Int32, CancellationToken)

Asynchronously change the output channel of the laser source.

Declaration

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

Parameters

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

Returns

TYPE	DESCRIPTION
Task	The task representing the operation.

Remarks

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

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified output channel is out of range for the laser source.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the RLM.

ChangeSwitchChannelAsync(Int32, Int32, CancellationToken)

Asynchronously change the channel of a connected switch.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
Int32	switchIndex	The switch number of the switch. `0` for the internal switch; `1` or `2` for external switches attached to the device.

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

Returns

TYPE	DESCRIPTION
Task	The task representing the operation.

Remarks

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

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device has no switch at the specified switch index.
ArgumentException	Thrown when the specified channel is out of range for the device switch.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

DarkAllDetectorsAsync(CancellationToken)

Asynchronously dark all of the detectors that support darking.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION

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

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

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

Exceptions

TYPE	CONDITION
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.
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.

DarkBRDetectorAsync(CancellationToken)

Asynchronously dark the BR detector.

Declaration

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

Parameters

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

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

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

Exceptions

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

DarkDetectorAsync(Int32, CancellationToken)

Asynchronously dark the specified detector.

Declaration

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

Parameters

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

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

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

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified detector is out of range for the device.
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.
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.

GetActiveLaserAsync(CancellationToken)

Asynchronously get the active laser.

Declaration

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

Parameters

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

Returns

TYPE	DESCRIPTION
Task<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

TYPE	CONDITION
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.
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

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

Parameters

TYPE	NAME	DESCRIPTION
Int32	wavelength	The wavelength to get the BR zero for.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Double>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
InvalidOperationException	Thrown when the method is called on an uninitialized device.

TYPE	CONDITION
UnexpectedDeviceResponseException	Thrown when the BR reference query response is not a valid number.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

GetDetectorDarkAsync(Int32, CancellationToken)

Asynchronously get whether the specified detector has a dark value.

Declaration

```
public Task<bool> GetDetectorDarkAsync(int detector, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	The detector to determine if it has a dark value.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Boolean>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified detector is out of range for the detector.
InvalidOperationException	Thrown when the method is called on an uninitialized detector.
UnexpectedDeviceResponseException	Thrown when the device detector dark query response is not a valid boolean value.

TYPE	CONDITION
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

GetFactoryPowerAsync(Int32, Int32, CancellationToken)

Asynchronously get the factory power for the specified output channel and wavelength.

Declaration

```
public Task<double> GetFactoryPowerAsync(int output, int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	output	The output channel to get the factory power for.
Int32	wavelength	The wavelength to get the factory power for.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Double>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified output channel is out of range.
ArgumentException	Thrown when the device does not support the specified wavelength.
InvalidOperationException	Thrown when the method is called on an uninitialized device.

TYPE	CONDITION
UnexpectedDeviceResponseException	Thrown when the device factory power response is not a valid number.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

GetILReferenceAsync(Int32, Int32, CancellationToken)

Asynchronously get the stored IL reference for a detector at a given wavelength.

Declaration

```
public Task<double> GetILReferenceAsync(int detector, int wavelength, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	The detector to get the reference for.
Int32	wavelength	The wavelength to get the reference for.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Double>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
ArgumentException	Thrown when the specified detector is out of range for the device.

TYPE	CONDITION
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.
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

```
public Task<int> GetOutputChannelAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Int32>	The task representing the operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

GetPDLReferencesAsync(Int32, Int32, CancellationToken)

Asynchronously get the stored PDL reference values for a detector at a given wavelength.

Declaration

```
public async Task<List<double>> GetPDLReferencesAsync(int detector, int wavelength, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	The detector to get the reference values for.
Int32	wavelength	The wavelength to get the reference values for.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<List<Double>>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the PLM does not support the specified wavelength.
ArgumentException	Thrown when the specified detector is out of range for the PLM.
InvalidOperationException	Thrown when the method is called on an uninitialized PLM.
UnexpectedDeviceResponseException	Thrown when the PLM PDL references query response is not a valid list of values.
TimeoutException	Thrown when a communication failure causes no response to be received from the PLM.

GetPolarizationModeAsync(CancellationToken)

Asynchronously get the polarization mode.

Declaration

```
public async Task<EPolarizationMode> GetPolarizationModeAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<EPolarizationMode>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized PLM.
UnexpectedDeviceResponseException	Thrown when the PLM response does not correspond to a polarization mode.
TimeoutException	Thrown when a communication failure causes no response to be received from the PLM.

GetPolarizationStateAsync(CancellationToken)

Asynchronously get the polarization state.

Declaration

```
public async Task<EPolarizationState> GetPolarizationStateAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<EPolarizationState>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized PLM.
UnexpectedDeviceResponseException	Thrown when the PLM response does not correspond to a polarization state.
TimeoutException	Thrown when a communication failure causes no response to be received from the PLM.

GetSwitchChannelAsync(Int32, CancellationToken)

Asynchronously get the channel of a connected switch.

Declaration

```
public Task<int> GetSwitchChannelAsync(int switchIndex, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
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.
CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
Task<Int32>	The task representing the operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device has no switch at the specified switch index.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

GetWaveplatePositionAsync(Int32, CancellationToken)

Asynchronously get the position of a given waveplate.

Declaration

```
public async Task<double> GetWaveplatePositionAsync(int waveplateIndex, CancellationToken cancellationToken
= default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	waveplateIndex	The waveplate to get the position of. `0` or `1` to specify which waveplate.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Double>	The task representing the asynchronous operation.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified waveplate is out of range for the device.
InvalidOperationException	Thrown when the method is called on an uninitialized PLM.
UnexpectedDeviceResponseException	Thrown when the PLM waveplate position response is not a valid waveplate position.

TYPE	CONDITION
TimeoutException	Thrown when a communication failure causes no response to be received from the PLM.

InitializeAsync()

Asynchronously open the device connection and initialize the device settings.

Declaration

```
public override async Task InitializeAsync()
```

Returns

TYPE	DESCRIPTION
Task	The task representing the operation.

Overrides

SantecDevice.InitializeAsync()

Remarks

> **NOTE**
>
> This operation will not block.

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.
TimeoutException	Thrown when a communication failure causes no response to be received from the PLM.

ReadBRAsync(Int32, CancellationToken)

Asynchronously read the backreflection at a given wavelength.

Declaration

```
public Task<double> ReadBRAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	wavelength	The wavelength to read the backreflection at.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Double>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device read BR response is not a valid number.
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

```
public Task<double> ReadILAsync(int detector, int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	The detector number of the detector to read the insertion loss from.

TYPE	NAME	DESCRIPTION
Int32	wavelength	The wavelength to read the insertion loss at.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Double>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
ArgumentException	Thrown when the specified detector is out of range for the device.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device read IL response is not a valid number.
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

```
public Task<double> ReadMonitorPowerAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION

TYPE	NAME	DESCRIPTION
Int32	wavelength	The wavelength to read the monitor power at.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Double>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device monitor power response is not a valid number.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

ReadMultipleILsAsync(IEnumerable<Int32>, Int32, CancellationToken)

Asynchronously read the insertion loss from multiple detectors at a given wavelength.

Declaration

```
public Task<List<double>> ReadMultipleILsAsync(IEnumerable<int> detectors, int wavelength, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
IEnumerable<Int32>	detectors	The detectors to read the insertion loss from.

TYPE	NAME	DESCRIPTION
Int32	wavelength	The wavelength to read the insertion loss at.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<List<Double>>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
ArgumentException	Thrown when any of the specified detectors are out of range for the device.
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.
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

```
public Task<List<double>> ReadMultiplePowersAsync(IEnumerable<int> detectors, int wavelength,
CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
IEnumerable<Int32>	detectors	The detectors to read from.
Int32	wavelength	The wavelength to read the power at.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<List<Double>>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
ArgumentException	Thrown when any of the specified detectors are out of range for the device.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

ReadPDLAndBRAsync(Int32, Int32, CancellationToken)

Asynchronously read the polarization dependent loss from a detector at a given wavelength and the backreflection at the wavelength.

Declaration

```
public async Task<PDLAndBRReading> ReadPDLAndBRAsync(int detector, int wavelength, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	The detector to read the polarization dependent loss for.
Int32	wavelength	The wavelength to read the polarization dependent loss and backreflection at.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<PDLAndBRReading>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the PLM does not support the specified wavelength.
ArgumentException	Thrown when the specified detector is out of range for the PLM.
InvalidOperationException	Thrown when the method is called on an uninitialized PLM.
UnexpectedDeviceResponseException	Thrown when the PLM read PDL and BR response is not a valid set of values.
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

```
public async Task<PDLReading> ReadPDLAsync(int detector, int wavelength, CancellationToken cancellationToken
= default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	The detector to read the polarization dependent loss for.
Int32	wavelength	The wavelength to read the polarization dependent loss at.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<PDLReading>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the PLM does not support the specified wavelength.
ArgumentException	Thrown when the specified detector is out of range for the PLM.
InvalidOperationException	Thrown when the method is called on an uninitialized PLM.
UnexpectedDeviceResponseException	Thrown when the PLM read PDL response is not a valid set of values.
TimeoutException	Thrown when a communication failure causes no response to be received from the PLM.

ReadPowerAsync(Int32, Int32, CancellationToken)

Asynchronously read the power from an device detector at a given wavelength.

Declaration

```
public Task<double> ReadPowerAsync(int detector, int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	The detector number of the detector to read from.
Int32	wavelength	The wavelength to read the power at.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Double>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
ArgumentException	Thrown when the specified detector is out of range for the device.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device read power response is not a valid number.
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

```
public Task<double> ReferenceBRAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	wavelength	The wavelength to perform the BR zero reference at.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Double>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device reference BR response is not a valid number.
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

```
public Task<double> ReferenceILAsync(int detector, int wavelength, bool applyReferenceToAllDetectors,
CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	The detector number of the detector to perform the reference for.
Int32	wavelength	The wavelength to perform the reference at.
Boolean	applyReferenceToAllDetectors	Indicate if the reference read should be applied to all detectors for the current channel.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Double>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
ArgumentException	Thrown when the specified detector is out of range for the device.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device reference IL response is not a valid number.
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

```
public Task<double> ReferenceILAsync(int detector, int wavelength, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	The detector number of the detector to perform the reference for.
Int32	wavelength	The wavelength to perform the reference at.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Double>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
ArgumentException	Thrown when the specified detector is out of range for the device.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device reference IL response is not a valid number.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

ReferencePDLAsync(Int32, Int32, CancellationToken)

Asynchronously perform a polarization dependent loss reference for a detector at a given wavelength.

Declaration

```
public async Task<List<double>> ReferencePDLAsync(int detector, int wavelength, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	The detector to perform the reference for.
Int32	wavelength	The wavelength to perform the reference at.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<List<Double>>	The task representing the operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the PLM does not support the specified wavelength.
ArgumentException	Thrown when the specified detector is out of range for the PLM.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the PLM.

RefreshAsync()

Asynchronously refresh the device settings.

Declaration

```
public override async Task RefreshAsync()
```

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Overrides

PhysicalDevice.RefreshAsync()

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the PLM.

SetBRReferenceAsync(Int32, Double, CancellationToken)

Asynchronously set the BR zero reference at a given wavelength.

Declaration

```
public Task SetBRReferenceAsync(int wavelength, double reference, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	wavelength	The wavelength to set the BR zero for.

TYPE	NAME	DESCRIPTION
Double	reference	The BR zero value.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
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.
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

```
public Task SetILReferenceAsync(int detector, int wavelength, double reference, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	The detector to set the reference for.

TYPE	NAME	DESCRIPTION
Int32	wavelength	The wavelength to set the reference for.
Double	reference	The IL reference value.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
ArgumentException	Thrown when the specified detector is out of range for the device.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

SetPolarizationModeAsync(EPolarizationMode, CancellationToken)

Asynchronously set the polarization mode.

Declaration

```
public async Task SetPolarizationModeAsync(EPolarizationMode polarizationMode, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
EPolarizationMode	polarizationMode	The polarization mode to change to.

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the PLM.

SetPolarizationStateAsync(EPolarizationState, CancellationToken)

Asynchronously set the polarization mode.

Declaration

```
public async Task SetPolarizationStateAsync(EPolarizationState polarizationState, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
EPolarizationState	polarizationState	The polarization state to change to.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized PLM.
UnexpectedDeviceResponseException	Thrown when the PLM polarization state response does not match the expected response.
TimeoutException	Thrown when a communication failure causes no response to be received from the PLM.

SetWaveplatePositionAsync(Int32, Double, CancellationToken)

Asynchronously set the position of a given waveplate.

Declaration

```
public async Task SetWaveplatePositionAsync(int waveplateIndex, double position, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	waveplateIndex	The waveplate to set the position of. `0` or `1` to specify which waveplate.
Double	position	The position to set the waveplate to. Between `0` and `360` degrees.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified waveplate is out of range for the device.
ArgumentException	Thrown when the specified position is not a valid waveplate position.
InvalidOperationException	Thrown when the method is called on an uninitialized PLM.
UnexpectedDeviceResponseException	Thrown when the PLM waveplate position response does not match the expected response.
TimeoutException	Thrown when a communication failure causes no response to be received from the PLM.

TurnOffLaserAsync(CancellationToken)

Asynchronously turn off the active laser.

Declaration

```
public Task TurnOffLaserAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device laser response does not match the expected response.

TYPE	CONDITION
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

```
public Task TurnOnLaserAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	wavelength	The wavelength of the laser to turn on.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device laser response does not match the expected response.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

Uninitialize()

Synchronously uninitialize device settings and close the connection.

Declaration

```
public override void Uninitialize()
```

Overrides

PhysicalDevice.Uninitialize()

Implements

IPLM
ISantecDevice
IDiscretePowerMeter
IPowerMeter
ILaserSource
ISwitchController
IBackreflectionMeter
IWavelengthDevice
IDetectorDeviceWithDarking
IDetectorDevice
IDevice
System.IDisposable
IFiberDevice

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

Object
SimulatedSantecDevice
SimulatedPLM

Implements

IPLM
ISantecDevice
IDiscretePowerMeter
ISwitchController
IFiberDevice
IPowerMeter
IBackreflectionMeter
IDetectorDeviceWithDarking
IDetectorDevice
ILaserSource
IWavelengthDevice
IDevice
IDisposable

Inherited Members

SimulatedSantecDevice.creatingDevice
SimulatedSantecDevice.Dispose()
SimulatedSantecDevice.SerialNumber
SimulatedSantecDevice.FirmwareVersion
SimulatedSantecDevice.Address
SimulatedSantecDevice.IsInitialized
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)
Object.ToString()
Object.Equals(Object)
Object.Equals(Object, Object)
Object.ReferenceEquals(Object, Object)
Object.GetHashCode()
Object.GetType()
Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.Polarization

Assembly: Santec.Hardware.dll

Syntax

```
public class SimulatedPLM : SimulatedSantecDevice, IPLM, ISantecDevice, IDiscretePowerMeter,
ISwitchController, IFiberDevice, IPowerMeter, IBackreflectionMeter, IDetectorDeviceWithDarking,
IDetectorDevice, ILaserSource, IWavelengthDevice, IDevice, IDisposable
```

Properties

BRMax

Declaration

```
public double BRMax { get; }
```

Property Value

TYPE	DESCRIPTION
Double	The maximum BR reading value. The default value is `-65`.

BRMin

Declaration

```
public double BRMin { get; }
```

Property Value

TYPE	DESCRIPTION
Double	The minimum BR reading value. The default value is `-75`.

BRReferenceMax

Declaration

```
public double BRReferenceMax { get; }
```

Property Value

TYPE	DESCRIPTION
Double	The maximum BR reference value. The default value is `-35`.

BRReferenceMin

Declaration

```
public double BRReferenceMin { get; }
```

Property Value

TYPE	DESCRIPTION
Double	The minimum BR reference value. The default value is `-36`.

Detectors

Get the detectors connected to the device.

> **NOTE**
>
> The device must be initialized before this property contains the proper value.

Declaration

```
public IReadOnlyList<DeviceDetector> Detectors { get; }
```

Property Value

TYPE	DESCRIPTION
IReadOnlyList<DeviceDetector>	The list of the detectors connected to the device.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the property is accessed before the device has been initialized.

Fiber

Get the fiber information of the device.

> **NOTE**
>
> The device must be initialized before this property contains the proper value.

Declaration

```
public FiberInformation Fiber { get; }
```

Property Value

TYPE	DESCRIPTION
FiberInformation	The fiber information of the device.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the property is accessed before the device has been initialized.

ILMax

Declaration

```
public double ILMax { get; }
```

Property Value

TYPE	DESCRIPTION
Double	The maximum IL reading value. The default value is `0.15`.

ILMin

Declaration

```
public double ILMin { get; }
```

Property Value

TYPE	DESCRIPTION
Double	The minimum IL reading value. The default value is `0`.

ILReferenceMax

Declaration

```
public double ILReferenceMax { get; }
```

Property Value

TYPE	DESCRIPTION
Double	The maximum IL reference value. The default value is `5`.

ILReferenceMin

Declaration

```
public double ILReferenceMin { get; }
```

Property Value

TYPE	DESCRIPTION
Double	The minimum IL reference value. The default value is `2`.

MonitorPowerMax

Declaration

```
public double MonitorPowerMax { get; }
```

Property Value

TYPE	DESCRIPTION
Double	The maximum monitor power value. The default value is `-30`.

MonitorPowerMin

Declaration

```
public double MonitorPowerMin { get; }
```

Property Value

TYPE	DESCRIPTION
Double	The minimum monitor power value. The default value is `-30`.

Name

Get the name of the device.

Declaration

```
public override string Name { get; }
```

Property Value

TYPE	DESCRIPTION
String	The name of the device.

Overrides

SimulatedSantecDevice.Name

Remarks

NOTE

This property will always return "PLM".

NumberOfDetectors

Get the number of detectors connected to the device.

> **NOTE**
>
> The device must be initialized before this property contains the proper value.

Declaration

```
public int NumberOfDetectors { get; }
```

Property Value

TYPE	DESCRIPTION
Int32	The number of detectors connected to the device.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the property is accessed before the device has been initialized.

NumberOfOutputs

Get the number of laser outputs.

> **NOTE**
>
> The device must be initialized before this property contains the proper value.

Declaration

```
public int NumberOfOutputs { get; }
```

Property Value

TYPE	DESCRIPTION
Int32	The number of device outputs.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the property is accessed before the device has been initialized.

Get the maximum PDL value returned by a PDL reading.

Declaration

```
public double PDLMax { get; }
```

Property Value

TYPE	DESCRIPTION
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
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
Double	The maximum power reading value. The default value is `-3`.

PowerMin

Declaration

```
public double PowerMin { get; }
```

Property Value

TYPE	DESCRIPTION
Double	The minimum power reading value. The default value is `-3.15`.

Switches

Get the switches connected to the device.

> **NOTE**
>
> The device must be initialized before this property contains the proper value.

Declaration

```
public List<DeviceSwitch> Switches { get; }
```

Property Value

TYPE	DESCRIPTION
List<DeviceSwitch>	The list of the switches connected to the device.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the property is accessed before the device has been initialized.

Wavelengths

Get the wavelengths supported by the device.

> **NOTE**
>
> The device must be initialized before this property contains the proper value.

Declaration

```
public List<int> Wavelengths { get; }
```

Property Value

TYPE	DESCRIPTION
List<Int32>	A list of the device's wavelengths.

Exceptions

TYPE	CONDITION

TYPE	CONDITION
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

```
public async Task ChangeMultipleSwitchesChannelsAsync(IEnumerable<DeviceSwitchAndChannelPair>
deviceSwitchesAndChannels, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
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.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device has no switch at any of the specified switch indexes.
ArgumentException	Thrown when any the specified channels is out of range for the corresponding device switch.
InvalidOperationException	Thrown when the method is called on an uninitialized device.

TYPE	CONDITION
UnexpectedDeviceResponseException	Thrown when the device switch channel response does not match the expected response.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

ChangeOutputChannelAsync(Int32, CancellationToken)

Asynchronously change the output channel of the laser source.

Declaration

```
public async Task ChangeOutputChannelAsync(int output, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	output	The output channel to switch to.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified output channel is out of range for the laser source.
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.

TYPE	CONDITION
TimeoutException	Thrown when a communication failure causes no response to be received from the RLM.

ChangeSwitchChannelAsync(Int32, Int32, CancellationToken)

Asynchronously change the channel of a connected switch.

Declaration

```
public async Task ChangeSwitchChannelAsync(int switchIndex, int channel, CancellationToken cancellationToken
= default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	switchIndex	The switch number of the switch. `0` for the internal switch; `1` or `2` for external switches attached to the device.
Int32	channel	The channel to switch to.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device has no switch at the specified switch index.
ArgumentException	Thrown when the specified channel is out of range for the device switch.
InvalidOperationException	Thrown when the method is called on an uninitialized device.

TYPE	CONDITION
UnexpectedDeviceResponseException	Thrown when the device switch channel response does not match the expected response.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

ConfigureBR(Double, Double)

Declaration

```
public void ConfigureBR(double brMin, double brMax)
```

Parameters

TYPE	NAME	DESCRIPTION
Double	brMin	
Double	brMax	

ConfigureBRReference(Double, Double)

Declaration

```
public void ConfigureBRReference(double brReferenceMin, double brReferenceMax)
```

Parameters

TYPE	NAME	DESCRIPTION
Double	brReferenceMin	
Double	brReferenceMax	

ConfigureIL(Double, Double)

Declaration

```
public void ConfigureIL(double ilMin, double ilMax)
```

Parameters

TYPE	NAME	DESCRIPTION
Double	ilMin	
Double	ilMax	

ConfigureILReference(Double, Double)

Declaration

```
public void ConfigureILReference(double ilReferenceMin, double ilReferenceMax)
```

Parameters

TYPE	NAME	DESCRIPTION
Double	ilReferenceMin	
Double	ilReferenceMax	

ConfigureMonitorPower(Double, Double)

Declaration

```
public void ConfigureMonitorPower(double monitorPowerMin, double monitorPowerMax)
```

Parameters

TYPE	NAME	DESCRIPTION
Double	monitorPowerMin	
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
Double	pdlMin	The minimum PDL value.
Double	pdlMax	The maximum PDL value.

ConfigurePower(Double, Double)

Declaration

```
public void ConfigurePower(double powerMin, double powerMax)
```

Parameters

TYPE	NAME	DESCRIPTION
Double	powerMin	
Double	powerMax	

DarkAllDetectorsAsync(CancellationToken)

Asynchronously dark all of the detectors that support darking

Asynchronously dark all of the detectors that support darking.

Declaration

```
public async Task DarkAllDetectorsAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
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.
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.

DarkBRDetectorAsync(CancellationToken)

Asynchronously dark the BR detector.

Declaration

```
public Task DarkBRDetectorAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

DarkDetectorAsync(Int32, CancellationToken)

Asynchronously dark the specified detector.

Declaration

```
public Task DarkDetectorAsync(int detector, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	The detector to dark.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified detector is out of range for the device.
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.
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.

GetActiveLaserAsync(CancellationToken)

Asynchronously get the active laser.

Declaration

```
public async Task<int> GetActiveLaserAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<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
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.
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

```
public async Task<double> GetBRReferenceAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	wavelength	The wavelength to get the BR zero for.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Double>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the BR reference query response is not a valid number.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

GetDetectorDarkAsync(Int32, CancellationToken)

Asynchronously get whether the specified detector has a dark value.

Declaration

```
public Task<bool> GetDetectorDarkAsync(int detector, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	The detector to determine if it has a dark value.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Boolean>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified detector is out of range for the detector.

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized detector.
UnexpectedDeviceResponseException	Thrown when the device detector dark query response is not a valid boolean value.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

GetFactoryPowerAsync(Int32, Int32, CancellationToken)

Asynchronously get the factory power for the specified output channel and wavelength.

Declaration

```
public async Task<double> GetFactoryPowerAsync(int output, int wavelength, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	output	The output channel to get the factory power for.
Int32	wavelength	The wavelength to get the factory power for.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Double>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified output channel is out of range.

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device factory power response is not a valid number.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

GetILReferenceAsync(Int32, Int32, CancellationToken)

Asynchronously get the stored IL reference for a detector at a given wavelength.

Declaration

```
public async Task<double> GetILReferenceAsync(int detector, int wavelength, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	The detector to get the reference for.
Int32	wavelength	The wavelength to get the reference for.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Double>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
ArgumentException	Thrown when the specified detector is out of range for the device.
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.
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

```
public async Task<int> GetOutputChannelAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Int32>	The task representing the operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized device.

TYPE	CONDITION
UnexpectedDeviceResponseException	Thrown when the device switch channel response is not a valid switch channel.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

GetPDLReferencesAsync(Int32, Int32, CancellationToken)

Asynchronously get the stored PDL reference values for a detector at a given wavelength.

Declaration

```
public async Task<List<double>> GetPDLReferencesAsync(int detector, int wavelength, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	The detector to get the reference values for.
Int32	wavelength	The wavelength to get the reference values for.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<List<Double>>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the PLM does not support the specified wavelength.
ArgumentException	Thrown when the specified detector is out of range for the PLM.

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized PLM.
UnexpectedDeviceResponseException	Thrown when the PLM PDL references query response is not a valid list of values.
TimeoutException	Thrown when a communication failure causes no response to be received from the PLM.

GetPolarizationModeAsync(CancellationToken)

Asynchronously get the polarization mode.

Declaration

```
public async Task<EPolarizationMode> GetPolarizationModeAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<EPolarizationMode>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized PLM.
UnexpectedDeviceResponseException	Thrown when the PLM response does not correspond to a polarization mode.
TimeoutException	Thrown when a communication failure causes no response to be received from the PLM.

GetPolarizationStateAsync(CancellationToken)

Asynchronously get the polarization mode.

Declaration

```
public async Task<EPolarizationState> GetPolarizationStateAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<EPolarizationState>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized PLM.
UnexpectedDeviceResponseException	Thrown when the PLM response does not correspond to a polarization mode.
TimeoutException	Thrown when a communication failure causes no response to be received from the PLM.

GetSwitchChannelAsync(Int32, CancellationToken)

Asynchronously get the channel of a connected switch.

Declaration

```
public async Task<int> GetSwitchChannelAsync(int switchIndex, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION

TYPE	NAME	DESCRIPTION
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.
CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
Task<Int32>	The task representing the operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device has no switch at the specified switch index.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

GetWaveplatePositionAsync(Int32, CancellationToken)

Asynchronously get the position of a given waveplate.

Declaration

```
public async Task<double> GetWaveplatePositionAsync(int waveplateIndex, CancellationToken cancellationToken
= default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	waveplateIndex	The waveplate to get the position of. `0` or `1` to specify which waveplate.

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Double>	The task representing the asynchronous operation.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified waveplate is out of range for the device.
InvalidOperationException	Thrown when the method is called on an uninitialized PLM.
UnexpectedDeviceResponseException	Thrown when the PLM waveplate position response is not a valid waveplate position.
TimeoutException	Thrown when a communication failure causes no response to be received from the PLM.

ReadBRAsync(Int32, CancellationToken)

Asynchronously read the backreflection at a given wavelength.

Declaration

```
public async Task<double> ReadBRAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	wavelength	The wavelength to read the backreflection at.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Double>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device read BR response is not a valid number.
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

```
public async Task<double> ReadILAsync(int detector, int wavelength, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	The detector number of the detector to read the insertion loss from.
Int32	wavelength	The wavelength to read the insertion loss at.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Double>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
ArgumentException	Thrown when the specified detector is out of range for the device.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device read IL response is not a valid number.
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

```
public async Task<double> ReadMonitorPowerAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	wavelength	The wavelength to read the monitor power at.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Double>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device monitor power response is not a valid number.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

ReadMultipleILsAsync(IEnumerable<Int32>, Int32, CancellationToken)

Asynchronously read the insertion loss from multiple detectors at a given wavelength.

Declaration

```
public async Task<List<double>> ReadMultipleILsAsync(IEnumerable<int> detectors, int wavelength,
CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
IEnumerable<Int32>	detectors	The detectors to read the insertion loss from.
Int32	wavelength	The wavelength to read the insertion loss at.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<List<Double>>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
ArgumentException	Thrown when any of the specified detectors are out of range for the device.
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.
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

```
public async Task<List<double>> ReadMultiplePowersAsync(IEnumerable<int> detectors, int wavelength,
CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
IEnumerable<Int32>	detectors	The detectors to read from.
Int32	wavelength	The wavelength to read the power at.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<List<Double>>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
ArgumentException	Thrown when any of the specified detectors are out of range for the device.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

ReadPDLAndBRAsync(Int32, Int32, CancellationToken)

Asynchronously read the polarization dependent loss from a detector at a given wavelength and the backreflection at the wavelength.

Declaration

```
public async Task<PDLAndBRReading> ReadPDLAndBRAsync(int detector, int wavelength, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	The detector to read the polarization dependent loss for.
Int32	wavelength	The wavelength to read the polarization dependent loss and backreflection at.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<PDLAndBRReading>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the PLM does not support the specified wavelength.
ArgumentException	Thrown when the specified detector is out of range for the PLM.
InvalidOperationException	Thrown when the method is called on an uninitialized PLM.
UnexpectedDeviceResponseException	Thrown when the PLM read PDL and BR response is not a valid set of values.
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

```
public async Task<PDLReading> ReadPDLAsync(int detector, int wavelength, CancellationToken cancellationToken
= default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	The detector to read the polarization dependent loss for.
Int32	wavelength	The wavelength to read the polarization dependent loss at.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<PDLReading>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the PLM does not support the specified wavelength.
ArgumentException	Thrown when the specified detector is out of range for the PLM.
InvalidOperationException	Thrown when the method is called on an uninitialized PLM.
UnexpectedDeviceResponseException	Thrown when the PLM read PDL response is not a valid set of values.
TimeoutException	Thrown when a communication failure causes no response to be received from the PLM.

ReadPowerAsync(Int32, Int32, CancellationToken)

Asynchronously read the power from an device detector at a given wavelength.

Declaration

```
public async Task<double> ReadPowerAsync(int detector, int wavelength, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	The detector number of the detector to read from.
Int32	wavelength	The wavelength to read the power at.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Double>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
ArgumentException	Thrown when the specified detector is out of range for the device.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device read power response is not a valid number.
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

```
public async Task<double> ReferenceBRAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	wavelength	The wavelength to perform the BR zero reference at.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Double>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device reference BR response is not a valid number.
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 a detector at a given wavelength.

Declaration

```
public async Task<double> ReferenceILAsync(int detector, int wavelength, bool applyReferenceToAllDetectors,
CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	The detector number of the detector to perform the reference for.
Int32	wavelength	The wavelength to perform the reference at.
Boolean	applyReferenceToAllDetectors	Indicate if the reference read should be applied to all detectors for the current channel.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Double>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
ArgumentException	Thrown when the specified detector is out of range for the device.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device reference IL response is not a valid number.
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

```
public async Task<double> ReferenceILAsync(int detector, int wavelength, CancellationToken cancellationToken
= default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	The detector number of the detector to perform the reference for.
Int32	wavelength	The wavelength to perform the reference at.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Double>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
ArgumentException	Thrown when the specified detector is out of range for the device.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device reference IL response is not a valid number.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

ReferencePDLAsync(Int32, Int32, CancellationToken)

Asynchronously perform a polarization dependent loss reference for a detector at a given wavelength.

Declaration

```
public async Task<List<double>> ReferencePDLAsync(int detector, int wavelength, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	The detector to perform the reference for.
Int32	wavelength	The wavelength to perform the reference at.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<List<Double>>	The task representing the operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the PLM does not support the specified wavelength.
ArgumentException	Thrown when the specified detector is out of range for the PLM.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the PLM.

SetBRReferenceAsync(Int32, Double, CancellationToken)

Asynchronously set the BR zero reference at a given wavelength.

Declaration

```
public async Task SetBRReferenceAsync(int wavelength, double reference, CancellationToken cancellationToken
= default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	wavelength	The wavelength to set the BR zero for.
Double	reference	The BR zero value.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
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.
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

```
public async Task SetILReferenceAsync(int detector, int wavelength, double reference, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	The detector to set the reference for.
Int32	wavelength	The wavelength to set the reference for.
Double	reference	The IL reference value.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.

TYPE	CONDITION
ArgumentException	Thrown when the specified detector is out of range for the device.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

SetPolarizationModeAsync(EPolarizationMode, CancellationToken)

Asynchronously set the polarization mode.

Declaration

```
public async Task SetPolarizationModeAsync(EPolarizationMode polarizationMode, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
EPolarizationMode	polarizationMode	The polarization mode to change to.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized PLM.

TYPE	CONDITION
UnexpectedDeviceResponseException	Thrown when the PLM polarization mode response does not match the expected response.
TimeoutException	Thrown when a communication failure causes no response to be received from the PLM.

SetPolarizationStateAsync(EPolarizationState, CancellationToken)

Asynchronously set the polarization mode.

Declaration

```
public async Task SetPolarizationStateAsync(EPolarizationState polarizationState, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
EPolarizationState	polarizationState	The polarization state to change to.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized PLM.
UnexpectedDeviceResponseException	Thrown when the PLM polarization state response does not match the expected response.
TimeoutException	Thrown when a communication failure causes no response to be received from the PLM.

SetWaveplatePositionAsync(Int32, Double, CancellationToken)

Asynchronously set the position of a given waveplate.

Declaration

```
public async Task SetWaveplatePositionAsync(int waveplateIndex, double position, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	waveplateIndex	The waveplate to set the position of. `0` or `1` to specify which waveplate.
Double	position	The position to set the waveplate to. Between `0` and `360` degrees.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified waveplate is out of range for the device.
ArgumentException	Thrown when the specified position is not a valid waveplate position.
InvalidOperationException	Thrown when the method is called on an uninitialized PLM.
UnexpectedDeviceResponseException	Thrown when the PLM waveplate position response does not match the expected response.
TimeoutException	Thrown when a communication failure causes no response to be received from the PLM.

TurnOffLaserAsync(CancellationToken)

Asynchronously turn off the active laser.

Declaration

```
public async Task TurnOffLaserAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device laser response does not match the expected response.
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

```
public async Task TurnOnLaserAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	wavelength	The wavelength of the laser to turn on.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device laser response does not match the expected response.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

Implements

IPLM
ISantecDevice
IDiscretePowerMeter
ISwitchController
IFiberDevice
IPowerMeter
IBackreflectionMeter
IDetectorDeviceWithDarking
IDetectorDevice
ILaserSource
IWavelengthDevice
IDevice
System.IDisposable

Namespace Santec.Hardware.Devices.Power

Classes

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.

OPMDetector

Represents an OPM detector

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.

Enums

EGainRange

Specifies the gain range of an OPM.

ELoggingStatus

Specifies the logging status of an OPM.

EReadingMode

Specifies the reading mode of an OPM.

EResolution

Specifies the reading resolution of an OPM.

ETriggerBehaviourMode

Specifies how the OPM responds to a trigger.

ETriggerEdge

Specifies the trigger edge of an OPM.

Enum EGainRange

Specifies the gain range of an OPM.

Namespace: Santec.Hardware.Devices.Power

Assembly: Santec.Hardware.dll

Syntax

```
public enum EGainRange
```

Fields

NAME	DESCRIPTION
Auto	Automatic gain range.
Minus10_Minus30	-10dBm to -30dBm.
Minus30_Minus50	-30dBm to -50dBm
Minus50_Minus70	-50dBm to -70dBm
Minus70_Minus80	-70dBm to -80dBm
Plus8_Minus10	+8dBm to -10dBm.

Enum ELoggingStatus

Specifies the logging status of an OPM.

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 EReadingMode

Specifies the reading mode of an OPM.

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 an OPM.

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 the OPM 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.

Enum ETriggerEdge

Specifies the trigger edge of an OPM.

Namespace: Santec.Hardware.Devices.Power

Assembly: Santec.Hardware.dll

Syntax

```
public enum ETriggerEdge
```

Fields

NAME	DESCRIPTION
Falling	Trigger on the falling edge.
Rising	Trigger on the rising edge.

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)
IPowerMeter.GetILReferenceAsync(Int32, Int32, CancellationToken)
IPowerMeter.SetILReferenceAsync(Int32, Int32, Double, CancellationToken)
IPowerMeter.ReferenceILAsync(Int32, Int32, CancellationToken)
IPowerMeter.ReferenceILAsync(Int32, Int32, Boolean, CancellationToken)
IPowerMeter.ReadILAsync(Int32, Int32, CancellationToken)
IDetectorDevice.NumberOfDetectors
IDetectorDevice.Detectors
IDevice.Name
IDevice.SerialNumber
IDevice.FirmwareVersion
IDevice.Address
IDevice.IsInitialized
IDevice.InitializeAsync()
IDevice.Uninitialize()
IDevice.ResetAsync()
IDevice.RefreshAsync()
IDevice.PingAsync()
IDevice.QueryAsync(String, CancellationToken)
IDevice.WriteAsync(String, CancellationToken)
IDevice.ReadAsync(CancellationToken)
IDisposable.Dispose()

Namespace: Santec.Hardware.Devices.Power

Assembly: Santec.Hardware.dll

Syntax

```
public interface IContinuousPowerMeter : IPowerMeter, 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)
IPowerMeter.GetILReferenceAsync(Int32, Int32, CancellationToken)
IPowerMeter.SetILReferenceAsync(Int32, Int32, Double, CancellationToken)
IPowerMeter.ReferenceILAsync(Int32, Int32, CancellationToken)
IPowerMeter.ReferenceILAsync(Int32, Int32, Boolean, CancellationToken)
IPowerMeter.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.ResetAsync()
IDevice.RefreshAsync()
IDevice.PingAsync()
IDevice.QueryAsync(String, CancellationToken)
IDevice.WriteAsync(String, CancellationToken)
IDevice.ReadAsync(CancellationToken)
IDisposable.Dispose()

Namespace: Santec.Hardware.Devices.Power

Assembly: Santec.Hardware.dll

Syntax

```
public interface IDiscretePowerMeter : IPowerMeter, 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
IEnumerable<Int32>	detectors	The detectors to read the insertion loss from.

TYPE	NAME	DESCRIPTION
Int32	wavelength	The wavelength to read the insertion loss at.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<List<Double>>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
ArgumentException	Thrown when any of the specified detectors are out of range for the device.
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.
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
IEnumerable<Int32>	detectors	The detectors to read from.
Int32	wavelength	The wavelength to read the power at.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<List<Double>>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
ArgumentException	Thrown when any of the specified detectors are out of range for the device.
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.
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.GetDetectorDarkAsync(Int32, CancellationToken)
IModuleController.NumberOfModules
IModuleController.GetSelectedModuleAsync(CancellationToken)
IModuleController.SetSelectedModuleAsync(Int32, CancellationToken)
IPowerMeter.ReadPowerAsync(Int32, Int32, CancellationToken)
IPowerMeter.GetILReferenceAsync(Int32, Int32, CancellationToken)
IPowerMeter.SetILReferenceAsync(Int32, Int32, Double, CancellationToken)
IPowerMeter.ReferenceILAsync(Int32, Int32, CancellationToken)
IPowerMeter.ReferenceILAsync(Int32, Int32, Boolean, CancellationToken)
IPowerMeter.ReadILAsync(Int32, Int32, CancellationToken)
IDetectorDevice.NumberOfDetectors
IDetectorDevice.Detectors
IDevice.Name
IDevice.SerialNumber
IDevice.FirmwareVersion
IDevice.Address
IDevice.IsInitialized
IDevice.InitializeAsync()
IDevice.Uninitialize()
IDevice.ResetAsync()
IDevice.RefreshAsync()
IDevice.PingAsync()
IDevice.QueryAsync(String, CancellationToken)
IDevice.WriteAsync(String, CancellationToken)
IDevice.ReadAsync(CancellationToken)
IDisposable.Dispose()

Namespace: Santec.Hardware.Devices.Power

Assembly: Santec.Hardware.dll

Syntax

```
public interface IOPM : ISantecDevice, IDetectorDeviceWithDarking, IModuleController, IContinuousPowerMeter,
IPowerMeter, IDetectorDevice, IDevice, IDisposable
```

Properties

MaximumAveragingTime

Get the maximum averaging time, in milliseconds, supported by the device.

> **NOTE**
>
> The device must be initialized before this property contains the proper value.

Declaration

```
double MaximumAveragingTime { get; }
```

Property Value

TYPE	DESCRIPTION
Double	The maximum averaging time in milliseconds.

MaximumNumberOfLoggingPoints

Get the maximum number of logging points supported by the device.

> **NOTE**
>
> The device must be initialized before this property contains the proper value.

Declaration

```
int MaximumNumberOfLoggingPoints { get; }
```

Property Value

TYPE	DESCRIPTION
Int32	The maximum number of logging points.

MinimumAveragingTime

Get the minimum averaging time, in milliseconds, supported by the device.

> **NOTE**
>
> The device must be initialized before this property contains the proper value.

Declaration

```
double MinimumAveragingTime { get; }
```

Property Value

TYPE	DESCRIPTION

TYPE	DESCRIPTION
Double	The minimum averaging time in milliseconds.

MinimumNumberOfLoggingPoints

Get the minimum number of logging points supported by the device.

> **NOTE**
>
> The device must be initialized before this property contains the proper value.

Declaration

```
int MinimumNumberOfLoggingPoints { get; }
```

Property Value

TYPE	DESCRIPTION
Int32	The minimum number of logging points.

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
Int32	detector	The detector number of the detector to get the averaging time for.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Double>	The task representing the asynchronous operation.

Remarks

> **NOTE**

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified detector is out of range for the device.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

GetDetectorAveragingTimeAsync(CancellationToken)

Asynchronously get the averaging time, in milliseconds, for the currently selected detector.

Declaration

```
Task<double> GetDetectorAveragingTimeAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Double>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
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.

TYPE	CONDITION
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<(bool isAutoGainRange, EGainRange gainRange)> GetDetectorGainRangeAsync(int detector, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	The detector number of the detector to get the gain range for.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<(T1, T2)<Boolean, EGainRange>>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified detector is out of range for the device.
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 reading mode response does not correspond to a reading mode.

TYPE	CONDITION
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

GetDetectorGainRangeAsync(CancellationToken)

Asynchronously get the gain range for the currently selected detector.

Declaration

```
Task<(bool isAutoGainRange, EGainRange gainRange)> GetDetectorGainRangeAsync(CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<(T1, T2)<Boolean, EGainRange>>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
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 reading mode response does not correspond to a reading mode.
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
Int32	detector	The detector number of the detector to get the reading mode for.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<EReadingMode>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified detector is out of range for the device.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

GetDetectorReadingsModeAsync(CancellationToken)

Asynchronously get the reading mode for the currently selected detector.

Declaration

```
Task<EReadingMode> GetDetectorReadingsModeAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<EReadingMode>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
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.
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
Int32	detector	The detector number of the detector to get the resolution for.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION

TYPE	DESCRIPTION
Task<EResolution>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified detector is out of range for the device.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device response does not correspond to a resolution.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

GetDetectorResolutionAsync(CancellationToken)

Asynchronously get the reading resolution for the currently selected detector.

Declaration

```
Task<EResolution> GetDetectorResolutionAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<EResolution>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device response does not correspond to a resolution.
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
Int32	detector	The detector number of the detector to get the wavelength for.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Int32>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified detector is out of range for the device.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

GetDetectorWavelengthAsync(CancellationToken)

Asynchronously get the wavelength for the currently selected detector.

Declaration

```
Task<int> GetDetectorWavelengthAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Int32>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device wavelength response is not a valid wavelength.
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 currently selected device detector at the currently selected wavelength.

Declaration

```
Task<double> GetILReferenceAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Double>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

GetSelectedDetectorAsync(CancellationToken)

Asynchronously get the currently selected detector.

> **NOTE**
>
> This method is an alias for GetSelectedModuleAsync(CancellationToken).

Declaration

```
Task<int> GetSelectedDetectorAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Int32>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device selected detector response is not a valid detector.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

LogDetectorAsync(Int32, Double, ETriggerEdge, ETriggerBehaviourMode, CancellationToken)

Asynchronously perform a logging operation for the currently selected detector.

Declaration

```
Task<IReadOnlyList<double>> LogDetectorAsync(int numberOfPoints, double averagingTime, ETriggerEdge
triggerEdge, ETriggerBehaviourMode triggerBehaviourMode, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	numberOfPoints	The number of points for the logging operation.
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.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<IReadOnlyList<Double>>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified number of logging points is out of range for the device.
ArgumentException	Thrown when the specified averaging time is out of range for the device.
ArgumentException	Thrown when the specified trigger behaviour mode is ignore triggers.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

LogDetectorAsync(Int32, Int32, Double, ETriggerEdge, ETriggerBehaviourMode, CancellationToken)

Asynchronously perform a logging operation for a device detector.

Declaration

```
Task<IReadOnlyList<double>> LogDetectorAsync(int detector, int numberOfPoints, double averagingTime,
ETriggerEdge triggerEdge, ETriggerBehaviourMode triggerBehaviourMode, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	The detector number of the detector to perform the logging operation for.

TYPE	NAME	DESCRIPTION
Int32	numberOfPoints	The number of points for the logging operation.
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.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<IReadOnlyList<Double>>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified detector is out of range for the device.
ArgumentException	Thrown when the specified number of logging points is out of range for the device.
ArgumentException	Thrown when the specified averaging time is out of range for the device.
ArgumentException	Thrown when the specified trigger behaviour mode is ignore triggers.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when a logging response does not match the expected response.

TYPE	CONDITION
UnexpectedDeviceResponseException	Thrown when a logging response is not a valid number.
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
Int32	wavelength	The wavelength to read the powers/powers in watts/ILs at.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<List<Double>>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified wavelength is out of range for the device.
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.

TYPE	CONDITION
UnexpectedDeviceResponseException	Thrown when any of the detector responses are not valid numbers.
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
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<List<Double>>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
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.
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
Int32	wavelength	The wavelength to read the ILs at.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<List<Double>>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified wavelength is out of range for the device.
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.
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
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<List<Double>>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
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.
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
Int32	wavelength	The wavelength to read the powers at.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<List<Double>>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified wavelength is out of range for the device.
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.
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
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<List<Double>>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
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.
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
Int32	wavelength	The wavelength to read the powers at.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<List<Double>>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified wavelength is out of range for the device.
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.
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
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<List<Double>>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized device.

TYPE	CONDITION
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.
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
Int32	detector	The detector number of the detector to read from.
Int32	wavelength	The wavelength to read the power/power in watts/IL at.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Double>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified wavelength is out of range for the device.

TYPE	CONDITION
ArgumentException	Thrown when the specified detector is out of range for the device.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

ReadDetectorAsync(CancellationToken)

Asynchronously read the power/power in watts/IL from the currently selected device detector at the currently selected wavelength.

Declaration

```
Task<double> ReadDetectorAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Double>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device read response is not a valid number.

TYPE	CONDITION
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

ReadILAsync(CancellationToken)

Asynchronously read the insertion loss from the currently selected device detector at the currently selected wavelength.

Declaration

```
Task<double> ReadILAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Double>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device read IL response is not a valid number.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

ReadPowerAsync(CancellationToken)

Asynchronously read the power from the currently selected device detector at the currently selected wavelength.

Declaration

```
Task<double> ReadPowerAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Double>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device read power response is not a valid number.
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
Int32	detector	The detector number of the detector to read from.
Int32	wavelength	The wavelength to read the power at.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Double>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified wavelength is out of range for the device.
ArgumentException	Thrown when the specified detector is out of range for the device.
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.
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 currently selected device detector at the currently selected wavelength.

Declaration

```
Task<double> ReadPowerInWattsAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Double>	The task representing the asynchronous operation.

Remarks

> **NOTE**

This operation will not block.

Exceptions

TYPE	CONDITION
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.
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 currently selected device detector at the currently selected wavelength.

Declaration

```
Task<double> ReferenceILAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Double>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device reference IL response is not a valid number.

TYPE	CONDITION
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
Double	averagingTime	The averaging time, in milliseconds, to set all detectors to.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified averaging time is out of range for the device.
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.
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.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
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.
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.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
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.
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
Int32	wavelength	The wavelength to set all the detectors to.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified wavelength is out of range for the device.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

SetDetectorAveragingTimeAsync(Double, CancellationToken)

Asynchronously set the averaging time, in milliseconds, for the currently selected detector.

Declaration

```
Task SetDetectorAveragingTimeAsync(double averagingTime, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Double	averagingTime	The averaging time, in milliseconds, to set the detector to.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified averaging time is out of range for the device.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device set averaging time response does not match the expected response.
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
Int32	detector	The detector number of the detector to set the resolution for.
Double	averagingTime	The averaging time, in milliseconds, to set the detector to.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified averaging time is out of range for the device.
ArgumentException	Thrown when the specified detector is out of range for the device.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

SetDetectorGainRangeAsync(EGainRange, CancellationToken)

Asynchronously set the gain range for the currently selected detector.

Declaration

```
Task SetDetectorGainRangeAsync(EGainRange gainRange, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
EGainRange	gainRange	The gain range to set the detector to.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

SetDetectorGainRangeAsync(Int32, EGainRange, CancellationToken)

Asynchronously set the gain range for a device detector.

Declaration

```
Task SetDetectorGainRangeAsync(int detector, EGainRange gainRange, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	The detector number of the detector to set the reading mode for.
EGainRange	gainRange	The gain range to set the detector to.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified detector is out of range for the device.

TYPE	CONDITION
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

SetDetectorReadingModeAsync(EReadingMode, CancellationToken)

Asynchronously set the reading mode for the currently selected detector.

Declaration

```
Task SetDetectorReadingModeAsync(EReadingMode readingMode, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
EReadingMode	readingMode	The reading mode to set the detector to.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
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.

TYPE	CONDITION
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
Int32	detector	The detector number of the detector to set the reading mode for.
EReadingMode	readingMode	The reading mode to set the detector to.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified detector is out of range for the device.
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.

TYPE	CONDITION
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

SetDetectorResolutionAsync(EResolution, CancellationToken)

Asynchronously set the reading resolution for the currently selected detector.

Declaration

```
Task SetDetectorResolutionAsync(EResolution resolution, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
EResolution	resolution	The resolution to set the detector to.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device set detector resolution response does not match the expected response.
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
Int32	detector	The detector number of the detector to set the resolution for.
EResolution	resolution	The resolution to set the detector to.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified detector is out of range for the device.
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.
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
Int32	detector	The detector number of the detector to set the wavelength for.
Int32	wavelength	The wavelength to set the detector to.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified wavelength is out of range for the device.
ArgumentException	Thrown when the specified detector is out of range for the device.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

SetDetectorWavelengthAsync(Int32, CancellationToken)

Asynchronously set the wavelength for the currently selected detector.

Declaration

```
Task SetDetectorWavelengthAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	wavelength	The wavelength to set the detector to.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified wavelength is out of range for the device.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

SetILReferenceAsync(Double, CancellationToken)

Asynchronously set the IL reference the currently selected device detector at the currently selected wavelength.

Declaration

```
Task SetILReferenceAsync(double reference, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Double	reference	The IL reference value.

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Exceptions

TYPE	CONDITION
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

SetSelectedDetectorAsync(Int32, CancellationToken)

Asynchronously set the currently selected detector.

> **NOTE**
>
> This method is an alias for SetSelectedModuleAsync(Int32, CancellationToken).

Declaration

```
Task SetSelectedDetectorAsync(int detector, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	The detector number of the detector to select.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified detector is out of range for the device.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device select detector response does not match the expected response.
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

IDetectorDevice.NumberOfDetectors
IDetectorDevice.Detectors
IDevice.Name
IDevice.SerialNumber
IDevice.FirmwareVersion
IDevice.Address
IDevice.IsInitialized
IDevice.InitializeAsync()
IDevice.Uninitialize()
IDevice.ResetAsync()
IDevice.RefreshAsync()
IDevice.PingAsync()
IDevice.QueryAsync(String, CancellationToken)
IDevice.WriteAsync(String, CancellationToken)
IDevice.ReadAsync(CancellationToken)
IDisposable.Dispose()

Namespace: Santec.Hardware.Devices.Power

Assembly: Santec.Hardware.dll

Syntax

```
public interface IPowerMeter : 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
Int32	detector	The detector to get the reference for.
Int32	wavelength	The wavelength to get the reference for.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Double>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
ArgumentException	Thrown when the specified detector is out of range for the device.
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.
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
Int32	detector	The detector number of the detector to read the insertion loss from.
Int32	wavelength	The wavelength to read the insertion loss at.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Double>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
ArgumentException	Thrown when the specified detector is out of range for the device.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device read IL response is not a valid number.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

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
Int32	detector	The detector number of the detector to read from.
Int32	wavelength	The wavelength to read the power at.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Double>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
ArgumentException	Thrown when the specified detector is out of range for the device.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device read power response is not a valid number.
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
Int32	detector	The detector number of the detector to perform the reference for.
Int32	wavelength	The wavelength to perform the reference at.
Boolean	applyReferenceToAllDetectors	Indicate if the reference read should be applied to all detectors for the current channel.

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Double>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
ArgumentException	Thrown when the specified detector is out of range for the device.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device reference IL response is not a valid number.
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
Int32	detector	The detector number of the detector to perform the reference for.

TYPE	NAME	DESCRIPTION
Int32	wavelength	The wavelength to perform the reference at.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Double>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
ArgumentException	Thrown when the specified detector is out of range for the device.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device reference IL response is not a valid number.
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

TYPE	NAME	DESCRIPTION
Int32	detector	The detector to set the reference for.
Int32	wavelength	The wavelength to set the reference for.
Double	reference	The IL reference value.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
ArgumentException	Thrown when the specified detector is out of range for the device.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

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

Object
PhysicalDevice
SantecDevice
OPM

Implements

IOPM
ISantecDevice
IDetectorDeviceWithDarking
IModuleController
IContinuousPowerMeter
IPowerMeter
IDetectorDevice
IDevice
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.Dispose()
PhysicalDevice.PingAsync()
PhysicalDevice.ResetAsync()
PhysicalDevice.QueryAsync(String, CancellationToken)
PhysicalDevice.WriteAsync(String, CancellationToken)
PhysicalDevice.ReadAsync(CancellationToken)
Object.ToString()

Object.Equals(Object)
Object.Equals(Object, Object)
Object.ReferenceEquals(Object, Object)
Object.GetHashCode()
Object.GetType()
Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.Power

Assembly: Santec.Hardware.dll

Syntax

```
public class OPM : SantecDevice, IOPM, ISantecDevice, IDetectorDeviceWithDarking, IModuleController,
IContinuousPowerMeter, IPowerMeter, 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

```
public IReadOnlyList<DeviceDetector> Detectors { get; }
```

Property Value

TYPE	DESCRIPTION
IReadOnlyList<DeviceDetector>	The list of the detectors connected to the device.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the property is accessed before the device has been initialized.

MaximumAveragingTime

Get the maximum averaging time, in milliseconds, supported by the device.

> **NOTE**
>
> The device must be initialized before this property contains the proper value.

Declaration

```
public double MaximumAveragingTime { get; }
```

Property Value

TYPE	DESCRIPTION
Double	The maximum averaging time in milliseconds.

Remarks

NOTE

This property will always return 1000ms.

MaximumNumberOfLoggingPoints

Get the maximum number of logging points supported by the device.

NOTE

The device must be initialized before this property contains the proper value.

Declaration

```
public int MaximumNumberOfLoggingPoints { get; }
```

Property Value

TYPE	DESCRIPTION
Int32	The maximum number of logging points.

Remarks

NOTE

This property will always return 128000.

MinimumAveragingTime

Get the minimum averaging time, in milliseconds, supported by the device.

NOTE

The device must be initialized before this property contains the proper value.

Declaration

```
public double MinimumAveragingTime { get; }
```

Property Value

TYPE	DESCRIPTION
Double	The minimum averaging time in milliseconds.

Remarks

> **NOTE**
>
> This property will always return 0.05ms.

MinimumNumberOfLoggingPoints

Get the minimum number of logging points supported by the device.

> **NOTE**
>
> The device must be initialized before this property contains the proper value.

Declaration

```
public int MinimumNumberOfLoggingPoints { get; }
```

Property Value

TYPE	DESCRIPTION
Int32	The minimum number of logging points.

Remarks

> **NOTE**
>
> This property will always return 1.

Name

Get the name of the device.

Declaration

```
public override string Name { get; }
```

Property Value

TYPE	DESCRIPTION
String	The name of the device.

Overrides

PhysicalDevice.Name

Remarks

> **NOTE**
>
> This property will always return "OPM".

NumberOfDetectors

Get the number of detectors connected to the device.

NOTE

The device must be initialized before this property contains the proper value.

Declaration

```
public int NumberOfDetectors { get; }
```

Property Value

TYPE	DESCRIPTION
Int32	The number of detectors connected to the device.

Remarks

NOTE

This property is an alias for NumberOfModules.

Exceptions

TYPE	CONDITION
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

```
public int NumberOfModules { get; }
```

Property Value

TYPE	DESCRIPTION
Int32	The number of modules in the device.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the property is accessed before the device has been initialized.

Methods

DarkAllDetectorsAsync(CancellationToken)

Asynchronously dark all of the detectors that support darking.

Declaration

```
public Task DarkAllDetectorsAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
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.
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

```
public Task DarkDetectorAsync(int detector, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	The detector to dark.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified detector is out of range for the device.
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.
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.

GetDetectorAveragingTimeAsync(Int32, CancellationToken)

Asynchronously get the averaging time, in milliseconds, for a device detector.

Declaration

```
public async Task<double> GetDetectorAveragingTimeAsync(int detector, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION

TYPE	NAME	DESCRIPTION
Int32	detector	The detector number of the detector to get the averaging time for.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Double>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified detector is out of range for the device.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

GetDetectorAveragingTimeAsync(CancellationToken)

Asynchronously get the averaging time, in milliseconds, for the currently selected detector.

Declaration

```
public async Task<double> GetDetectorAveragingTimeAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Double>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

GetDetectorDarkAsync(Int32, CancellationToken)

Asynchronously get whether the specified detector has a dark value.

Declaration

```
public Task<bool> GetDetectorDarkAsync(int detector, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	The detector to determine if it has a dark value.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Boolean>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified detector is out of range for the detector.
InvalidOperationException	Thrown when the method is called on an uninitialized detector.
UnexpectedDeviceResponseException	Thrown when the device detector dark query response is not a valid boolean value.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

GetDetectorGainRangeAsync(Int32, CancellationToken)

Declaration

```
public Task<(bool isAutoGainRange, EGainRange gainRange)> GetDetectorGainRangeAsync(int detector,
CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	
CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
Task<(T1, T2)<Boolean, EGainRange>>	

GetDetectorGainRangeAsync(CancellationToken)

Asynchronously get the gain range for the currently selected detector.

Declaration

```
public Task<(bool isAutoGainRange, EGainRange gainRange)> GetDetectorGainRangeAsync(CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<(T1, T2)<Boolean, EGainRange>>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
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 reading mode response does not correspond to a reading mode.
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

```
public async Task<EReadingMode> GetDetectorReadingModeAsync(int detector, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	The detector number of the detector to get the reading mode for.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<EReadingMode>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified detector is out of range for the device.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

GetDetectorReadingsModeAsync(CancellationToken)

Asynchronously get the reading mode for the currently selected detector.

Declaration

```
public async Task<EReadingMode> GetDetectorReadingsModeAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<EReadingMode>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized device.

TYPE	CONDITION
UnexpectedDeviceResponseException	Thrown when the device reading mode response does not correspond to a reading mode.
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

```
public async Task<EResolution> GetDetectorResolutionAsync(int detector, CancellationToken cancellationToken
= default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	The detector number of the detector to get the resolution for.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<EResolution>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified detector is out of range for the device.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device response does not correspond to a resolution.

TYPE	CONDITION
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

GetDetectorResolutionAsync(CancellationToken)

Asynchronously get the reading resolution for the currently selected detector.

Declaration

```
public async Task<EResolution> GetDetectorResolutionAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<EResolution>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device response does not correspond to a resolution.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

GetDetectorTriggerEdgeAsync(Int32, CancellationToken)

Declaration

```
public Task<ETriggerEdge> GetDetectorTriggerEdgeAsync(int detector, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	
CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
Task<ETriggerEdge>	

GetDetectorTriggerEdgeAsync(CancellationToken)

Declaration

```
public Task<ETriggerEdge> GetDetectorTriggerEdgeAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
Task<ETriggerEdge>	

GetDetectorWavelengthAsync(Int32, CancellationToken)

Asynchronously get the wavelength for a device detector.

Declaration

```
public async Task<int> GetDetectorWavelengthAsync(int detector, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	The detector number of the detector to get the wavelength for.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Int32>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified detector is out of range for the device.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device wavelength response is not a valid wavelength.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

GetDetectorWavelengthAsync(CancellationToken)

Asynchronously get the wavelength for the currently selected detector.

Declaration

```
public async Task<int> GetDetectorWavelengthAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Int32>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device wavelength response is not a valid wavelength.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

GetILReferenceAsync(Int32, Int32, CancellationToken)

Asynchronously get the stored IL reference for a detector at a given wavelength.

Declaration

```
public async Task<double> GetILReferenceAsync(int detector, int wavelength, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	The detector to get the reference for.
Int32	wavelength	The wavelength to get the reference for.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Double>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.

TYPE	CONDITION
ArgumentException	Thrown when the specified detector is out of range for the device.
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.
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 currently selected device detector at the currently selected wavelength.

Declaration

```
public async Task<double> GetILReferenceAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Double>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
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.

TYPE	CONDITION
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

GetSelectedDetectorAsync(CancellationToken)

Asynchronously get the currently selected detector.

> **NOTE**
>
> This method is an alias for GetSelectedModuleAsync(CancellationToken).

Declaration

```
public async Task<int> GetSelectedDetectorAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Int32>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device selected detector response is not a valid detector.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

GetSelectedModuleAsync(CancellationToken)

Asynchronously get the currently selected module.

Declaration

```
public Task<int> GetSelectedModuleAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Int32>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device selected module response is not a valid module.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

InitializeAsync()

Asynchronously open the device connection and initialize the device settings.

Declaration

```
public override async Task InitializeAsync()
```

Returns

TYPE	DESCRIPTION
Task	The task representing the operation.

Overrides

SantecDevice.InitializeAsync()

Remarks

NOTE

This operation will not block.

LogDetectorAsync(Int32, Double, ETriggerEdge, ETriggerBehaviourMode, CancellationToken)

Asynchronously perform a logging operation for a device detector.

Declaration

```
public Task<IReadOnlyList<double>> LogDetectorAsync(int numberOfPoints, double averagingTime, ETriggerEdge
triggerEdge, ETriggerBehaviourMode triggerBehaviourMode, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	numberOfPoints	The number of points for the logging operation.
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.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<IReadOnlyList<Double>>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified detector is out of range for the device.
ArgumentException	Thrown when the specified number of logging points is out of range for the device.

TYPE	CONDITION
ArgumentException	Thrown when the specified averaging time is out of range for the device.
ArgumentException	Thrown when the specified trigger behaviour mode is ignore triggers.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

LogDetectorAsync(Int32, Int32, Double, ETriggerEdge, ETriggerBehaviourMode, CancellationToken)

Asynchronously perform a logging operation for a device detector.

Declaration

```
public Task<IReadOnlyList<double>> LogDetectorAsync(int detector, int numberOfPoints, double averagingTime,
ETriggerEdge triggerEdge, ETriggerBehaviourMode triggerBehaviourMode, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	The detector number of the detector to perform the logging operation for.
Int32	numberOfPoints	The number of points for the logging operation.
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.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<IReadOnlyList<Double>>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified detector is out of range for the device.
ArgumentException	Thrown when the specified number of logging points is out of range for the device.
ArgumentException	Thrown when the specified averaging time is out of range for the device.
ArgumentException	Thrown when the specified trigger behaviour mode is ignore triggers.
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.
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

```
public Task<List<double>> ReadAllDetectorsAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION

TYPE	NAME	DESCRIPTION
Int32	wavelength	The wavelength to read the powers/powers in watts/ILs at.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<List<Double>>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified wavelength is out of range for the device.
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.
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

```
public Task<List<double>> ReadAllDetectorsAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<List<Double>>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
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.
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

```
public Task<List<double>> ReadAllILsAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	wavelength	The wavelength to read the ILs at.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<List<Double>>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified wavelength is out of range for the device.
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.
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

```
public Task<List<double>> ReadAllILsAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<List<Double>>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
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.
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

```
public Task<List<double>> ReadAllPowersAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	wavelength	The wavelength to read the powers at.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<List<Double>>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION

TYPE	CONDITION
ArgumentException	Thrown when the specified wavelength is out of range for the device.
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.
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

```
public Task<List<double>> ReadAllPowersAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<List<Double>>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized device.

TYPE	CONDITION
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.
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

```
public Task<List<double>> ReadAllPowersInWattsAsync(int wavelength, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	wavelength	The wavelength to read the powers at.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<List<Double>>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified wavelength is out of range for the device.
InvalidOperationException	Thrown when the method is called on an uninitialized device.

TYPE	CONDITION
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.
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

```
public Task<List<double>> ReadAllPowersInWattsAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<List<Double>>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
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.

TYPE	CONDITION
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

```
public Task<double> ReadDetectorAsync(int detector, int wavelength, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	The detector number of the detector to read from.
Int32	wavelength	The wavelength to read the power/power in watts/IL at.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Double>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified wavelength is out of range for the device.
ArgumentException	Thrown when the specified detector is out of range for the device.
InvalidOperationException	Thrown when the method is called on an uninitialized device.

TYPE	CONDITION
UnexpectedDeviceResponseException	Thrown when the device read power in watts response is not a valid number.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

ReadDetectorAsync(CancellationToken)

Asynchronously read the power/power in watts/IL from the currently selected device detector at the currently selected wavelength.

Declaration

```
public Task<double> ReadDetectorAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Double>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device read response is not a valid number.
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

```
public Task<double> ReadILAsync(int detector, int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	The detector number of the detector to read the insertion loss from.
Int32	wavelength	The wavelength to read the insertion loss at.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Double>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
ArgumentException	Thrown when the specified detector is out of range for the device.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device read IL response is not a valid number.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

ReadILAsync(CancellationToken)

Asynchronously read the insertion loss from the currently selected device detector at the currently selected wavelength.

Declaration

```
public Task<double> ReadILAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Double>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device read IL response is not a valid number.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

ReadPowerAsync(Int32, Int32, CancellationToken)

Asynchronously read the power from an device detector at a given wavelength.

Declaration

```
public Task<double> ReadPowerAsync(int detector, int wavelength, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	The detector number of the detector to read from.
Int32	wavelength	The wavelength to read the power at.

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Double>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
ArgumentException	Thrown when the specified detector is out of range for the device.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device read power response is not a valid number.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

ReadPowerAsync(CancellationToken)

Asynchronously read the power from the currently selected device detector at the currently selected wavelength.

Declaration

```
public Task<double> ReadPowerAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Double>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device read power response is not a valid number.
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

```
public Task<double> ReadPowerInWattsAsync(int detector, int wavelength, CancellationToken cancellationToken
= default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	The detector number of the detector to read from.
Int32	wavelength	The wavelength to read the power at.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Double>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified wavelength is out of range for the device.
ArgumentException	Thrown when the specified detector is out of range for the device.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

ReadPowerInWattsAsync(CancellationToken)

Asynchronously read the power in watts from all detectors at the currently selected wavelength.

Declaration

```
public Task<double> ReadPowerInWattsAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Double>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
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.
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

```
public async Task<double> ReferenceILAsync(int detector, int wavelength, bool applyReferenceToAllDetectors,
CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	The detector number of the detector to perform the reference for.
Int32	wavelength	The wavelength to perform the reference at.
Boolean	applyReferenceToAllDetectors	Indicate if the reference read should be applied to all detectors for the current channel.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Double>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
ArgumentException	Thrown when the specified detector is out of range for the device.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device reference IL response is not a valid number.
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

```
public Task<double> ReferenceILAsync(int detector, int wavelength, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	The detector number of the detector to perform the reference for.
Int32	wavelength	The wavelength to perform the reference at.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Double>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
ArgumentException	Thrown when the specified detector is out of range for the device.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device reference IL response is not a valid number.
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 currently selected device detector at the currently selected wavelength.

Declaration

```
public Task<double> ReferenceILAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Double>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized device.

TYPE	CONDITION
UnexpectedDeviceResponseException	Thrown when the device reference IL response is not a valid number.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

RefreshAsync()

Asynchronously refresh the device settings.

Declaration

```
public override async Task RefreshAsync()
```

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Overrides

PhysicalDevice.RefreshAsync()

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized RLM.
UnexpectedDeviceResponseException	Thrown when the OPM modules response is not a valid module count.

SetAllDetectorAveragingTimesAsync(Double, CancellationToken)

Asynchronously set the averaging time, in milliseconds, for all device detectors.

Declaration

```
public Task SetAllDetectorAveragingTimesAsync(double averagingTime, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION

TYPE	NAME	DESCRIPTION
Double	averagingTime	The averaging time, in milliseconds, to set all detectors to.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified averaging time is out of range for the device.
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.
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

```
public Task SetAllDetectorReadingModesAsync(EReadingMode readingMode, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
EReadingMode	readingMode	The reading mode to set the detectors to.

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
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.
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

```
public Task SetAllDetectorResolutionsAsync(EResolution resolution, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
EResolution	resolution	The resolution to set all the detectors to.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
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.
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

```
public Task SetAllDetectorWavelengthsAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	wavelength	The wavelength to set all the detectors to.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified wavelength is out of range for the device.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

SetDetectorAveragingTimeAsync(Double, CancellationToken)

Asynchronously set the averaging time, in milliseconds, for the currently selected detector.

Declaration

```
public Task SetDetectorAveragingTimeAsync(double averagingTime, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Double	averagingTime	The averaging time, in milliseconds, to set the detector to.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION

TYPE	CONDITION
ArgumentException	Thrown when the specified averaging time is out of range for the device.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device set averaging time response does not match the expected response.
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

```
public Task SetDetectorAveragingTimeAsync(int detector, double averagingTime, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	The detector number of the detector to set the resolution for.
Double	averagingTime	The averaging time, in milliseconds, to set the detector to.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION

TYPE	CONDITION
ArgumentException	Thrown when the specified averaging time is out of range for the device.
ArgumentException	Thrown when the specified detector is out of range for the device.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

SetDetectorGainRangeAsync(EGainRange, CancellationToken)

Asynchronously set the gain range for the currently selected detector.

Declaration

```
public Task SetDetectorGainRangeAsync(EGainRange gainRange, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
EGainRange	gainRange	The gain range to set the detector to.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

SetDetectorGainRangeAsync(Int32, EGainRange, CancellationToken)

Asynchronously set the gain range for a device detector.

Declaration

```
public Task SetDetectorGainRangeAsync(int detector, EGainRange gainRange, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	The detector number of the detector to set the reading mode for.
EGainRange	gainRange	The gain range to set the detector to.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified detector is out of range for the device.

TYPE	CONDITION
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

SetDetectorReadingModeAsync(EReadingMode, CancellationToken)

Asynchronously set the reading mode for the currently selected detector.

Declaration

```
public Task SetDetectorReadingModeAsync(EReadingMode readingMode, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
EReadingMode	readingMode	The reading mode to set the detector to.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
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.

TYPE	CONDITION
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

```
public Task SetDetectorReadingModeAsync(int detector, EReadingMode readingMode, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	The detector number of the detector to set the reading mode for.
EReadingMode	readingMode	The reading mode to set the detector to.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified detector is out of range for the device.
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.

TYPE	CONDITION
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

SetDetectorResolutionAsync(EResolution, CancellationToken)

Asynchronously set the reading resolution for the currently selected detector.

Declaration

```
public Task SetDetectorResolutionAsync(EResolution resolution, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
EResolution	resolution	The resolution to set the detector to.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device set detector resolution response does not match the expected response.
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

```
public Task SetDetectorResolutionAsync(int detector, EResolution resolution, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	The detector number of the detector to set the resolution for.
EResolution	resolution	The resolution to set the detector to.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified detector is out of range for the device.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

SetDetectorTriggerEdgeAsync(ETriggerEdge, CancellationToken)

Declaration

```
public Task SetDetectorTriggerEdgeAsync(ETriggerEdge triggerEdge, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
ETriggerEdge	triggerEdge	
CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
Task	

SetDetectorTriggerEdgeAsync(Int32, ETriggerEdge, CancellationToken)

Declaration

```
public Task SetDetectorTriggerEdgeAsync(int detector, ETriggerEdge triggerEdge, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	
ETriggerEdge	triggerEdge	
CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
Task	

SetDetectorWavelengthAsync(Int32, Int32, CancellationToken)

Asynchronously set the wavelength for a device detector.

Declaration

```
public Task SetDetectorWavelengthAsync(int detector, int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	The detector number of the detector to set the wavelength for.
Int32	wavelength	The wavelength to set the detector to.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

TYPE	NAME	DESCRIPTION

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified wavelength is out of range for the device.
ArgumentException	Thrown when the specified detector is out of range for the device.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

SetDetectorWavelengthAsync(Int32, CancellationToken)

Asynchronously set the wavelength for the currently selected detector.

Declaration

```
public Task SetDetectorWavelengthAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	wavelength	The wavelength to set the detector to.

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified wavelength is out of range for the device.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

SetILReferenceAsync(Double, CancellationToken)

Asynchronously set the IL reference the currently selected device detector at the currently selected wavelength.

Declaration

```
public Task SetILReferenceAsync(double reference, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Double	reference	The IL reference value.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Exceptions

TYPE	CONDITION
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.
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

```
public Task SetILReferenceAsync(int detector, int wavelength, double reference, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	The detector to set the reference for.
Int32	wavelength	The wavelength to set the reference for.
Double	reference	The IL reference value.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
ArgumentException	Thrown when the specified detector is out of range for the device.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

SetSelectedDetectorAsync(Int32, CancellationToken)

Asynchronously set the currently selected detector.

> **NOTE**
>
> This method is an alias for SetSelectedModuleAsync(Int32, CancellationToken).

Declaration

```
public async Task SetSelectedDetectorAsync(int detector, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	The detector number of the detector to select.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified detector is out of range for the device.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device select detector response does not match the expected response.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

SetSelectedModuleAsync(Int32, CancellationToken)

Asynchronously set the currently selected module.

Declaration

```
public Task SetSelectedModuleAsync(int moduleIndex, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	moduleIndex	The index of the module to select.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified module is out of range for the device.

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device select module response does not match the expected response.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

Uninitialize()

Synchronously uninitialize device settings and close the connection.

Declaration

```
public override void Uninitialize()
```

Overrides

PhysicalDevice.Uninitialize()

Implements

IOPM
ISantecDevice
IDetectorDeviceWithDarking
IModuleController
IContinuousPowerMeter
IPowerMeter
IDetectorDevice
IDevice
System.IDisposable

Class OPMDetector

Represents an OPM detector

Inheritance

Object
DeviceDetector
OPMDetector

Inherited Members

DeviceDetector.Index
DeviceDetector.IsRDSP
Object.ToString()
Object.Equals(Object)
Object.Equals(Object, Object)
Object.ReferenceEquals(Object, Object)
Object.GetHashCode()
Object.GetType()
Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.Power

Assembly: Santec.Hardware.dll

Syntax

```
public class OPMDetector : DeviceDetector
```

Constructors

OPMDetector(Int32, Int32, Int32)

Initialize new OPM detector.

Declaration

```
public OPMDetector(int index, int minimumWavelength, int maximumWavelength)
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	index	The index of the detector.
Int32	minimumWavelength	The minimum wavelength of the detector.
Int32	maximumWavelength	The maximum wavelength of the detector.

Properties

MaximumWavelength

Get the maximum wavelength of the detector.

Declaration

```
public int MaximumWavelength { get; }
```

Property Value

TYPE	DESCRIPTION
Int32	The maximum wavelength of the detector.

MinimumWavelength

Get the minimum wavelength of the detector.

Declaration

```
public int MinimumWavelength { get; }
```

Property Value

TYPE	DESCRIPTION
Int32	The minimum wavelength of the detector.

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.

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.

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 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.
Standard	Standard 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)
IPowerMeter.ReadPowerAsync(Int32, Int32, CancellationToken)
IPowerMeter.GetILReferenceAsync(Int32, Int32, CancellationToken)
IPowerMeter.SetILReferenceAsync(Int32, Int32, Double, CancellationToken)
IPowerMeter.ReferenceILAsync(Int32, Int32, CancellationToken)
IPowerMeter.ReferenceILAsync(Int32, Int32, Boolean, CancellationToken)
IPowerMeter.ReadILAsync(Int32, Int32, CancellationToken)
IDetectorDevice.NumberOfDetectors
IDetectorDevice.Detectors
ILaserSource.NumberOfOutputs
ILaserSource.GetActiveLaserAsync(CancellationToken)
ILaserSource.TurnOnLaserAsync(Int32, CancellationToken)
ILaserSource.TurnOffLaserAsync(CancellationToken)
ILaserSource.GetOutputChannelAsync(CancellationToken)
ILaserSource.ChangeOutputChannelAsync(Int32, CancellationToken)
ILaserSource.ReadMonitorPowerAsync(Int32, CancellationToken)
ILaserSource.GetFactoryPowerAsync(Int32, Int32, CancellationToken)
IWavelengthDevice.Wavelengths
ISwitchController.Switches
ISwitchController.GetSwitchChannelAsync(Int32, CancellationToken)
ISwitchController.ChangeSwitchChannelAsync(Int32, Int32, CancellationToken)
ISwitchController.ChangeMultipleSwitchesChannelsAsync(IEnumerable<DeviceSwitchAndChannelPair>, CancellationToken)
IDevice.Name
IDevice.SerialNumber
IDevice.FirmwareVersion
IDevice.Address
IDevice.IsInitialized
IDevice.InitializeAsync()
IDevice.Uninitialize()
IDevice.ResetAsync()
IDevice.RefreshAsync()
IDevice.PingAsync()
IDevice.QueryAsync(String, CancellationToken)
IDevice.WriteAsync(String, CancellationToken)
IDevice.ReadAsync(CancellationToken)

IDisposable.Dispose()
IFiberDevice.Fiber

Namespace: Santec.Hardware.Devices.ReturnLoss

Assembly: Santec.Hardware.dll

Syntax

```
public interface IRLM : ISantecDevice, IDiscretePowerMeter, IPowerMeter, IDetectorDevice, ILaserSource,
IWavelengthDevice, ISwitchController, IDevice, IDisposable, IFiberDevice
```

Properties

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
Boolean	`true` if the RLM supports changing the IL resolution; otherwise, `false`.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the property is accessed before the RLM has been initialized.

Methods

GetDUTILAsync(CancellationToken)

Asynchronously get the DUT IL.

Declaration

```
Task<double> GetDUTILAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Double>	The task representing the operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized RLM.
UnexpectedDeviceResponseException	Thrown when the device DUT IL response is not a valid number.
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
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<EGainMode>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized RLM.
UnexpectedDeviceResponseException	Thrown when the device response does not correspond to a gain mode.
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
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
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
InvalidOperationException	Thrown when the method is called on an uninitialized device.

TYPE	CONDITION
InvalidOperationException	Thrown when the RLM does not support IL resolution operations.
UnexpectedDeviceResponseException	Thrown when the device response does not correspond to a resolution.
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
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<ELengthMode>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized RLM.
UnexpectedDeviceResponseException	Thrown when the device response does not correspond to a length mode.
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
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<EPositionBMode>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
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.
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
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<ERLMode>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized RLM.
UnexpectedDeviceResponseException	Thrown when the device response does not correspond to an RL mode.
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
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Double>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
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.
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
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
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
InvalidOperationException	Thrown when the method is called on an uninitialized RLM.

TYPE	CONDITION
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the RLM.
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
Int32	wavelength	The wavelength to read the return loss at.
Double	lengthReferenceA	The reference length to use for position A.
Double	lengthReferenceB	The reference length to use for position B.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
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
ArgumentException	Thrown when the RLM does not support the specified wavelength.
ArgumentException	Thrown when either RL reference length is less than `0`.
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.
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
Int32	wavelength	The wavelength to read the return loss at.
Double	lengthReference	The reference length to use for the reading.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
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
ArgumentException	Thrown when the RLM does not support the specified wavelength.
ArgumentException	Thrown when the RLM reference length is less than `0`.
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.
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
Int32	wavelength	The wavelength to read the return loss at.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
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
ArgumentException	Thrown when the RLM does not support the specified wavelength.
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.
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
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION

TYPE	DESCRIPTION
Task<Double>	The task representing the operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized RLM.
UnexpectedDeviceResponseException	Thrown when the device reference RL response is not a valid number.
TimeoutException	Thrown when a communication failure causes no response to be received from the RLM.

SetDUTILAsync(Double, CancellationToken)

Asynchronously set the DUT IL.

Declaration

```
Task SetDUTILAsync(double il, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Double	il	The DUT IL value.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
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
ArgumentException	Thrown when the specified DUT IL is less than `0`.
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.
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.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized RLM.

TYPE	CONDITION
UnexpectedDeviceResponseException	Thrown when the device gain mode response does not match the expected response.
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.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
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
ArgumentException	Thrown when the IL resolution is not supported by the RLM.

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized device.
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.
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.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized RLM.

TYPE	CONDITION
UnexpectedDeviceResponseException	Thrown when the device length mode response does not match the expected response.
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.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized RLM.
UnexpectedDeviceResponseException	Thrown when the device positionB mode response does not match the expected response.
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.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized RLM.
UnexpectedDeviceResponseException	Thrown when the device RL mode response does not match the expected response.
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

TYPE	NAME	DESCRIPTION
Double	reference	The RL length reference value.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
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
ArgumentException	Thrown when the RL length reference value is less than `0`.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the RLM.

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

Object
PhysicalDevice
SantecDevice
RLM

Implements

IRLM
ISantecDevice
IDiscretePowerMeter
IPowerMeter
IDetectorDevice
ILaserSource
IWavelengthDevice
ISwitchController
IDevice
IDisposable
IFiberDevice

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.Dispose()
PhysicalDevice.PingAsync()
PhysicalDevice.ResetAsync()
PhysicalDevice.QueryAsync(String, CancellationToken)
PhysicalDevice.WriteAsync(String, CancellationToken)

PhysicalDevice.ReadAsync(CancellationToken)
Object.ToString()
Object.Equals(Object)
Object.Equals(Object, Object)
Object.ReferenceEquals(Object, Object)
Object.GetHashCode()
Object.GetType()
Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.ReturnLoss

Assembly: Santec.Hardware.dll

Syntax

```
public class RLM : SantecDevice, IRLM, ISantecDevice, IDiscretePowerMeter, IPowerMeter, IDetectorDevice,
ILaserSource, IWavelengthDevice, ISwitchController, IDevice, IDisposable, IFiberDevice
```

Properties

Detectors

Get the detectors connected to the device.

> **NOTE**
>
> The device must be initialized before this property contains the proper value.

Declaration

```
public IReadOnlyList<DeviceDetector> Detectors { get; }
```

Property Value

TYPE	DESCRIPTION
IReadOnlyList<DeviceDetector>	The list of the detectors connected to the device.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the property is accessed before the device has been initialized.

Fiber

Get the fiber information of the device.

> **NOTE**
>
> The device must be initialized before this property contains the proper value.

Declaration

```
public FiberInformation Fiber { get; }
```

Property Value

TYPE	DESCRIPTION
FiberInformation	The fiber information of the device.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the property is accessed before the device has been initialized.

Name

Get the name of the device.

Declaration

```
public override string Name { get; }
```

Property Value

TYPE	DESCRIPTION
String	The name of the device.

Overrides

PhysicalDevice.Name

Remarks

> **NOTE**
>
> This property will always return "RLM".

NumberOfDetectors

Get the number of detectors connected to the device.

> **NOTE**
>
> The device must be initialized before this property contains the proper value.

Declaration

```
public int NumberOfDetectors { get; }
```

Property Value

TYPE	DESCRIPTION
Int32	The number of detectors connected to the device.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the property is accessed before the device has been initialized.

NumberOfOutputs

Get the number of laser outputs.

> **NOTE**
>
> The device must be initialized before this property contains the proper value.

Declaration

```
public int NumberOfOutputs { get; }
```

Property Value

TYPE	DESCRIPTION
Int32	The number of device outputs.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the property is accessed before the device 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

```
public bool SupportsILResolution { get; }
```

Property Value

TYPE	DESCRIPTION
Boolean	`true` if the RLM supports changing the IL resolution; otherwise, `false`.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the property is accessed before the RLM has been initialized.

Switches

Get the switches connected to the device.

> **NOTE**
>
> The device must be initialized before this property contains the proper value.

Declaration

```
public List<DeviceSwitch> Switches { get; }
```

Property Value

TYPE	DESCRIPTION
List<DeviceSwitch>	The list of the switches connected to the device.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the property is accessed before the device has been initialized.

Wavelengths

Get the wavelengths supported by the device.

> **NOTE**
>
> The device must be initialized before this property contains the proper value.

Declaration

```
public List<int> Wavelengths { get; }
```

Property Value

TYPE	DESCRIPTION
List<Int32>	A list of the device's wavelengths.

Exceptions

TYPE	CONDITION
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

```
public Task ChangeMultipleSwitchesChannelsAsync(IEnumerable<DeviceSwitchAndChannelPair>
deviceSwitchesAndChannels, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
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.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device has no switch at any of the specified switch indexes.

TYPE	CONDITION
ArgumentException	Thrown when any the specified channels is out of range for the corresponding device switch.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

ChangeOutputChannelAsync(Int32, CancellationToken)

Asynchronously change the output channel of the laser source.

Declaration

```
public Task ChangeOutputChannelAsync(int output, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	output	The output channel to switch to.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION

TYPE	CONDITION
ArgumentException	Thrown when the specified output channel is out of range for the laser source.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the RLM.

ChangeSwitchChannelAsync(Int32, Int32, CancellationToken)

Asynchronously change the channel of a connected switch.

Declaration

```
public Task ChangeSwitchChannelAsync(int switchIndex, int channel, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	switchIndex	The switch number of the switch. `0` for the internal switch; `1` or `2` for external switches attached to the device.
Int32	channel	The channel to switch to.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device has no switch at the specified switch index.
ArgumentException	Thrown when the specified channel is out of range for the device switch.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

GetActiveLaserAsync(CancellationToken)

Asynchronously get the active laser.

Declaration

```
public Task<int> GetActiveLaserAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<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
InvalidOperationException	Thrown when the method is called on an uninitialized device.

TYPE	CONDITION
UnexpectedDeviceResponseException	Thrown when the device response does not correspond to a supported wavelength or to no active laser.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

GetDUTILAsync(CancellationToken)

Asynchronously get the DUT IL.

Declaration

```
public async Task<double> GetDUTILAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Double>	The task representing the operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized RLM.
UnexpectedDeviceResponseException	Thrown when the device DUT IL response is not a valid number.
TimeoutException	Thrown when a communication failure causes no response to be received from the RLM.

GetFactoryPowerAsync(Int32, Int32, CancellationToken)

Asynchronously get the factory power for the specified output channel and wavelength.

Declaration

```
public Task<double> GetFactoryPowerAsync(int output, int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	output	The output channel to get the factory power for.
Int32	wavelength	The wavelength to get the factory power for.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Double>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified output channel is out of range.
ArgumentException	Thrown when the device does not support the specified wavelength.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device factory power response is not a valid number.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

GetGainModeAsync(CancellationToken)

Asynchronously get the gain mode.

Declaration

```
public async Task<EGainMode> GetGainModeAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<EGainMode>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized RLM.
UnexpectedDeviceResponseException	Thrown when the device response does not correspond to a gain mode.
TimeoutException	Thrown when a communication failure causes no response to be received from the RLM.

GetILReferenceAsync(Int32, Int32, CancellationToken)

Asynchronously get the stored IL reference for a detector at a given wavelength.

Declaration

```
public Task<double> GetILReferenceAsync(int detector, int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	The detector to get the reference for.
Int32	wavelength	The wavelength to get the reference for.

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Double>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
ArgumentException	Thrown when the specified detector is out of range for the device.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

GetILResolutionAsync(CancellationToken)

Asynchronously get the IL reading resolution.

Declaration

```
public async Task<EResolution> GetILResolutionAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
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
InvalidOperationException	Thrown when the method is called on an uninitialized device.
InvalidOperationException	Thrown when the RLM does not support IL resolution operations.
UnexpectedDeviceResponseException	Thrown when the device response does not correspond to a resolution.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

GetLengthModeAsync(CancellationToken)

Asynchronously get the length mode.

Declaration

```
public async Task<ELengthMode> GetLengthModeAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<ELengthMode>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized RLM.
UnexpectedDeviceResponseException	Thrown when the device response does not correspond to a length mode.
TimeoutException	Thrown when a communication failure causes no response to be received from the RLM.

GetOutputChannelAsync(CancellationToken)

Asynchronously get the current output channel of the laser source.

Declaration

```
public Task<int> GetOutputChannelAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Int32>	The task representing the operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
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
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

GetPositionBModeAsync(CancellationToken)

Asynchronously get the RL position B mode.

Declaration

```
public async Task<EPositionBMode> GetPositionBModeAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<EPositionBMode>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the RLM.

GetRLModeAsync(CancellationToken)

Asynchronously get the RL sensitivity mode.

Declaration

```
public async Task<ERLMode> GetRLModeAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<ERLMode>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized RLM.
UnexpectedDeviceResponseException	Thrown when the device response does not correspond to an RL mode.
TimeoutException	Thrown when a communication failure causes no response to be received from the RLM.

GetRLReferenceAsync(CancellationToken)

Asynchronously get the stored RL reference.

Declaration

```
public async Task<double> GetRLReferenceAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Double>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the RLM.

GetSwitchChannelAsync(Int32, CancellationToken)

Asynchronously get the channel of a connected switch.

Declaration

```
public Task<int> GetSwitchChannelAsync(int switchIndex, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
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.
CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
Task<Int32>	The task representing the operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device has no switch at the specified switch index.

TYPE	CONDITION
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

GetTraceDiagnosticsAsync(CancellationToken)

Asynchronously get the trace diagnostics for the last RL measurement.

Declaration

```
public async Task<TraceDiagnostics> GetTraceDiagnosticsAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
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
InvalidOperationException	Thrown when the method is called on an uninitialized RLM.

TYPE	CONDITION
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the RLM.
NotSupportedException	Thrown when the method is called on an RLM through a USB connection.

InitializeAsync()

Asynchronously open the device connection and initialize the device settings.

Declaration

```
public override async Task InitializeAsync()
```

Returns

TYPE	DESCRIPTION
Task	The task representing the operation.

Overrides

SantecDevice.InitializeAsync()

Remarks

NOTE

This operation will not block.

Exceptions

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.

TYPE	CONDITION
TimeoutException	Thrown when a communication failure causes no response to be received from the RLM.

ReadILAsync(Int32, Int32, CancellationToken)

Asynchronously read the insertion loss from a detector at a given wavelength.

Declaration

```
public Task<double> ReadILAsync(int detector, int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	The detector number of the detector to read the insertion loss from.
Int32	wavelength	The wavelength to read the insertion loss at.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Double>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
ArgumentException	Thrown when the specified detector is out of range for the device.
InvalidOperationException	Thrown when the method is called on an uninitialized device.

TYPE	CONDITION
UnexpectedDeviceResponseException	Thrown when the device read IL response is not a valid number.
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

```
public Task<double> ReadMonitorPowerAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	wavelength	The wavelength to read the monitor power at.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Double>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device monitor power response is not a valid number.

TYPE	CONDITION
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

ReadMultipleILsAsync(IEnumerable<Int32>, Int32, CancellationToken)

Asynchronously read the insertion loss from multiple detectors at a given wavelength.

Declaration

```
public Task<List<double>> ReadMultipleILsAsync(IEnumerable<int> detectors, int wavelength, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
IEnumerable<Int32>	detectors	The detectors to read the insertion loss from.
Int32	wavelength	The wavelength to read the insertion loss at.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<List<Double>>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
ArgumentException	Thrown when any of the specified detectors are out of range for the device.
InvalidOperationException	Thrown when the method is called on an uninitialized device.

TYPE	CONDITION
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.
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

```
public Task<List<double>> ReadMultiplePowersAsync(IEnumerable<int> detectors, int wavelength,
CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
IEnumerable<Int32>	detectors	The detectors to read from.
Int32	wavelength	The wavelength to read the power at.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<List<Double>>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.

TYPE	CONDITION
ArgumentException	Thrown when any of the specified detectors are out of range for the device.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

ReadPowerAsync(Int32, Int32, CancellationToken)

Asynchronously read the power from an device detector at a given wavelength.

Declaration

```
public Task<double> ReadPowerAsync(int detector, int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	The detector number of the detector to read from.
Int32	wavelength	The wavelength to read the power at.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Double>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
ArgumentException	Thrown when the specified detector is out of range for the device.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device read power response is not a valid number.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

ReadRLAsync(Int32, Double, Double, CancellationToken)

Asynchronously read the return loss at a given wavelength using the specified reference lengths.

Declaration

```
public async Task<RLReading> ReadRLAsync(int wavelength, double lengthReferenceA, double lengthReferenceB,
CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	wavelength	The wavelength to read the return loss at.
Double	lengthReferenceA	The reference length to use for position A.
Double	lengthReferenceB	The reference length to use for position B.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
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
ArgumentException	Thrown when the RLM does not support the specified wavelength.
ArgumentException	Thrown when either RL reference length is less than `0`.
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.
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

```
public async Task<RLReading> ReadRLAsync(int wavelength, double lengthReference, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	wavelength	The wavelength to read the return loss at.
Double	lengthReference	The reference length to use for the reading.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
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
ArgumentException	Thrown when the RLM does not support the specified wavelength.
ArgumentException	Thrown when the RLM reference length is less than `0`.
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.
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

```
public async Task<RLReading> ReadRLAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	wavelength	The wavelength to read the return loss at.

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
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
ArgumentException	Thrown when the RLM does not support the specified wavelength.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the RLM.

ReferenceILAsync(Int32, Int32, Boolean, CancellationToken)

Asynchronously perform an insertion loss reference for all detectors at a given wavelength.

Declaration

```
public Task<double> ReferenceILAsync(int detector, int wavelength, bool applyReferenceToAllDetectors,
CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION

TYPE	NAME	DESCRIPTION
Int32	detector	The detector number of the detector to perform the reference for.
Int32	wavelength	The wavelength to perform the reference at.
Boolean	applyReferenceToAllDetectors	Indicate if the reference read should be applied to all detectors for the current channel.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Double>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
ArgumentException	Thrown when the specified detector is out of range for the device.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device reference IL response is not a valid number.
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

```
public Task<double> ReferenceILAsync(int detector, int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	The detector number of the detector to perform the reference for.
Int32	wavelength	The wavelength to perform the reference at.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Double>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
ArgumentException	Thrown when the specified detector is out of range for the device.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device reference IL response is not a valid number.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

ReferenceRLAsync(CancellationToken)

Asynchronously perform a return loss length reference.

Declaration

```
public async Task<double> ReferenceRLAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Double>	The task representing the operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized RLM.
UnexpectedDeviceResponseException	Thrown when the device reference RL response is not a valid number.
TimeoutException	Thrown when a communication failure causes no response to be received from the RLM.

RefreshAsync()

Asynchronously refresh the device settings.

Declaration

```
public override async Task RefreshAsync()
```

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Overrides

PhysicalDevice.RefreshAsync()

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized RLM.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the RLM.

SetDUTILAsync(Double, CancellationToken)

Asynchronously set the DUT IL.

Declaration

```
public async Task SetDUTILAsync(double il, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Double	il	The DUT IL value.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
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
ArgumentException	Thrown when the specified DUT IL is less than `0`.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the RLM.

SetGainModeAsync(EGainMode, CancellationToken)

Asynchronously set the gain mode.

Declaration

```
public async Task SetGainModeAsync(EGainMode gainMode, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
EGainMode	gainMode	The gain mode to change to.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized RLM.

TYPE	CONDITION
UnexpectedDeviceResponseException	Thrown when the device gain mode response does not match the expected response.
TimeoutException	Thrown when a communication failure causes no response to be received from the RLM.

SetILReferenceAsync(Int32, Int32, Double, CancellationToken)

Asynchronously set the IL reference for a detector at a given wavelength.

Declaration

```
public Task SetILReferenceAsync(int detector, int wavelength, double reference, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	The detector to set the reference for.
Int32	wavelength	The wavelength to set the reference for.
Double	reference	The IL reference value.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
ArgumentException	Thrown when the specified detector is out of range for the device.
InvalidOperationException	Thrown when the method is called on an uninitialized device.

TYPE	CONDITION
UnexpectedDeviceResponseException	Thrown when the device IL reference query response does not match the expected response.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

SetILResolutionAsync(EResolution, CancellationToken)

Asynchronously set the IL reading resolution.

Declaration

```
public async Task SetILResolutionAsync(EResolution resolution, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
EResolution	resolution	The resolution to set the detector to.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
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
ArgumentException	Thrown when the IL resolution is not supported by the RLM.

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized device.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

SetLengthModeAsync(ELengthMode, CancellationToken)

Asynchronously set the length mode.

Declaration

```
public async Task SetLengthModeAsync(ELengthMode lengthMode, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
ELengthMode	lengthMode	The length mode to change to.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized RLM.

TYPE	CONDITION
UnexpectedDeviceResponseException	Thrown when the device length mode response does not match the expected response.
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

```
public async Task SetPositionBModeAsync(EPositionBMode positionBMode, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
EPositionBMode	positionBMode	The position B mode to change to.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized RLM.
UnexpectedDeviceResponseException	Thrown when the device positionB mode response does not match the expected response.
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

```
public async Task SetRLModeAsync(ERLMode rlMode, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
ERLMode	rlMode	The RL mode to change to.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized RLM.
UnexpectedDeviceResponseException	Thrown when the device RL mode response does not match the expected response.
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

```
public async Task SetRLReferenceAsync(double reference, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Double	reference	The RL length reference value.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
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
ArgumentException	Thrown when the RL length reference value is less than `0`.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the RLM.

TurnOffLaserAsync(CancellationToken)

Asynchronously turn off the active laser.

Declaration

```
public Task TurnOffLaserAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device laser response does not match the expected response.
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

```
public Task TurnOnLaserAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	wavelength	The wavelength of the laser to turn on.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device laser response does not match the expected response.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

Uninitialize()

Synchronously uninitialize device settings and close the connection.

Declaration

```
public override void Uninitialize()
```

Overrides

PhysicalDevice.Uninitialize()

Implements

IRLM
ISantecDevice
IDiscretePowerMeter
IPowerMeter
IDetectorDevice
ILaserSource
IWavelengthDevice
ISwitchController
IDevice
System.IDisposable
IFiberDevice

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

Object

RLReading

Inherited Members

Object.ToString()

Object.Equals(Object)

Object.Equals(Object, Object)

Object.ReferenceEquals(Object, Object)

Object.GetHashCode()

Object.GetType()

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.
Double	length	The measured length of the fiber.

Properties

Length

The length of the fiber.

Declaration

```
public double Length { get; }
```

Property Value

TYPE	DESCRIPTION
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

Object
SimulatedSantecDevice
SimulatedRLM

Implements

IRLM
ISantecDevice
IDiscretePowerMeter
ISwitchController
IFiberDevice
IPowerMeter
IDetectorDevice
ILaserSource
IWavelengthDevice
IDevice
IDisposable

Inherited Members

SimulatedSantecDevice.creatingDevice
SimulatedSantecDevice.Dispose()
SimulatedSantecDevice.SerialNumber
SimulatedSantecDevice.FirmwareVersion
SimulatedSantecDevice.Address
SimulatedSantecDevice.IsInitialized
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)
Object.ToString()
Object.Equals(Object)
Object.Equals(Object, Object)
Object.ReferenceEquals(Object, Object)
Object.GetHashCode()
Object.GetType()
Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.ReturnLoss

Assembly: Santec.Hardware.dll

Syntax

```
public class SimulatedRLM : SimulatedSantecDevice, IRLM, ISantecDevice, IDiscretePowerMeter,
ISwitchController, IFiberDevice, IPowerMeter, IDetectorDevice, ILaserSource, IWavelengthDevice, 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

```
public IReadOnlyList<DeviceDetector> Detectors { get; }
```

Property Value

TYPE	DESCRIPTION
IReadOnlyList<DeviceDetector>	The list of the detectors connected to the device.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the property is accessed before the device has been initialized.

Fiber

Get the fiber information of the device.

> **NOTE**
>
> The device must be initialized before this property contains the proper value.

Declaration

```
public FiberInformation Fiber { get; }
```

Property Value

TYPE	DESCRIPTION
FiberInformation	The fiber information of the device.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the property is accessed before the device has been initialized.

ILMax

Declaration

```
public double ILMax { get; }
```

Property Value

TYPE	DESCRIPTION
Double	The maximum IL reading value. The default value is `0.15`.

ILMin

Declaration

```
public double ILMin { get; }
```

Property Value

TYPE	DESCRIPTION
Double	The minimum IL reading value. The default value is `0`.

ILReferenceMax

Declaration

```
public double ILReferenceMax { get; }
```

Property Value

TYPE	DESCRIPTION
Double	The maximum IL reference value. The default value is `5`.

ILReferenceMin

Declaration

```
public double ILReferenceMin { get; }
```

Property Value

TYPE	DESCRIPTION
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
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
Double	The minimum length reading value. The default value is `2.95`.

Remarks

Use ConfigureLength(Double, Double) to set this value.

MonitorPowerMax

Declaration

```
public double MonitorPowerMax { get; }
```

Property Value

TYPE	DESCRIPTION
Double	The maximum monitor power value. The default value is `-30`.

MonitorPowerMin

Declaration

```
public double MonitorPowerMin { get; }
```

Property Value

TYPE	DESCRIPTION
Double	The minimum monitor power value. The default value is `-30`.

Name

Get the name of the device.

Declaration

```
public override string Name { get; }
```

Property Value

TYPE	DESCRIPTION
String	The name of the device.

Overrides

SimulatedSantecDevice.Name

Remarks

> **NOTE**
>
> This property will always return "RLM".

NumberOfDetectors

Get the number of detectors connected to the device.

> **NOTE**
>
> The device must be initialized before this property contains the proper value.

Declaration

```
public int NumberOfDetectors { get; }
```

Property Value

TYPE	DESCRIPTION
Int32	The number of detectors connected to the device.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the property is accessed before the device has been initialized.

NumberOfOutputs

Get the number of laser outputs.

> **NOTE**
>
> The device must be initialized before this property contains the proper value.

Declaration

```
public int NumberOfOutputs { get; }
```

Property Value

TYPE	DESCRIPTION
Int32	The number of device outputs.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the property is accessed before the device has been initialized.

PowerMax

Declaration

```
public double PowerMax { get; }
```

Property Value

TYPE	DESCRIPTION
Double	The maximum power reading value. The default value is `-3`.

PowerMin

Declaration

```
public double PowerMin { get; }
```

Property Value

TYPE	DESCRIPTION
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
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
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
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
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
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
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
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
Double	The minimum RLtotal reading value. The default value is `55`.

Remarks

Use ConfigureRLtotal(Double, Double) to set this value.

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

```
public bool SupportsILResolution { get; }
```

Property Value

TYPE	DESCRIPTION
Boolean	`true` if the RLM supports changing the IL resolution; otherwise, `false`.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the property is accessed before the RLM has been initialized.

Switches

Get the switches connected to the device.

NOTE

The device must be initialized before this property contains the proper value.

Declaration

```
public List<DeviceSwitch> Switches { get; }
```

Property Value

TYPE	DESCRIPTION
List<DeviceSwitch>	The list of the switches connected to the device.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the property is accessed before the device has been initialized.

Wavelengths

Get the wavelengths supported by the device.

NOTE

The device must be initialized before this property contains the proper value.

Declaration

```
public List<int> Wavelengths { get; }
```

Property Value

TYPE	DESCRIPTION
List<Int32>	A list of the device's wavelengths.

Exceptions

TYPE	CONDITION
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

```
public async Task ChangeMultipleSwitchesChannelsAsync(IEnumerable<DeviceSwitchAndChannelPair>
deviceSwitchesAndChannels, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
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.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device has no switch at any of the specified switch indexes.
ArgumentException	Thrown when any the specified channels is out of range for the corresponding device switch.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

ChangeOutputChannelAsync(Int32, CancellationToken)

Asynchronously change the output channel of the laser source.

Declaration

```
public async Task ChangeOutputChannelAsync(int output, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	output	The output channel to switch to.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION

TYPE	CONDITION
ArgumentException	Thrown when the specified output channel is out of range for the laser source.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the RLM.

ChangeSwitchChannelAsync(Int32, Int32, CancellationToken)

Asynchronously change the channel of a connected switch.

Declaration

```
public async Task ChangeSwitchChannelAsync(int switchIndex, int channel, CancellationToken cancellationToken
= default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	switchIndex	The switch number of the switch. `0` for the internal switch; `1` or `2` for external switches attached to the device.
Int32	channel	The channel to switch to.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device has no switch at the specified switch index.
ArgumentException	Thrown when the specified channel is out of range for the device switch.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

ConfigureIL(Double, Double)

Declaration

```
public void ConfigureIL(double ilMin, double ilMax)
```

Parameters

TYPE	NAME	DESCRIPTION
Double	ilMin	
Double	ilMax	

ConfigureILReference(Double, Double)

Declaration

```
public void ConfigureILReference(double ilReferenceMin, double ilReferenceMax)
```

Parameters

TYPE	NAME	DESCRIPTION
Double	ilReferenceMin	
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
Double	lengthMin	The minimum length value.
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
Double	monitorPowerMin	
Double	monitorPowerMax	

ConfigurePower(Double, Double)

Declaration

```
public void ConfigurePower(double powerMin, double powerMax)
```

Parameters

TYPE	NAME	DESCRIPTION
Double	powerMin	
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
Double	rlAMin	The minimum RLa value.

TYPE	NAME	DESCRIPTION
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
Double	rlBMin	The minimum RLb value.
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
Double	rlReferenceMin	The minimum length value.
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
Double	rlTotalMin	The minimum RLtotal value.
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.

DarkAllDetectorsAsync(CancellationToken)

Declaration

```
public async Task DarkAllDetectorsAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
Task	

DarkDetectorAsync(Int32, CancellationToken)

Declaration

```
public async Task DarkDetectorAsync(int detector, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	
CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
Task	

GetActiveLaserAsync(CancellationToken)

Asynchronously get the active laser.

Declaration

```
public async Task<int> GetActiveLaserAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<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
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

GetDetectorDarkAsync(Int32, CancellationToken)

Declaration

```
public async Task<bool> GetDetectorDarkAsync(int detector, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
Task<Boolean>	

GetDUTILAsync(CancellationToken)

Asynchronously get the DUT IL.

Declaration

```
public async Task<double> GetDUTILAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Double>	The task representing the operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized RLM.
UnexpectedDeviceResponseException	Thrown when the device DUT IL response is not a valid number.
TimeoutException	Thrown when a communication failure causes no response to be received from the RLM.

GetFactoryPowerAsync(Int32, Int32, CancellationToken)

Asynchronously get the factory power for the specified output channel and wavelength.

Declaration

```
public async Task<double> GetFactoryPowerAsync(int channel, int wavelength, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	channel	
Int32	wavelength	The wavelength to get the factory power for.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Double>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified output channel is out of range.
ArgumentException	Thrown when the device does not support the specified wavelength.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device factory power response is not a valid number.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

GetGainModeAsync(CancellationToken)

Asynchronously get the gain mode.

Declaration

```
public async Task<EGainMode> GetGainModeAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<EGainMode>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized RLM.
UnexpectedDeviceResponseException	Thrown when the device response does not correspond to a gain mode.
TimeoutException	Thrown when a communication failure causes no response to be received from the RLM.

GetILReferenceAsync(Int32, Int32, CancellationToken)

Asynchronously get the stored IL reference for a detector at a given wavelength.

Declaration

```
public async Task<double> GetILReferenceAsync(int detector, int wavelength, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	The detector to get the reference for.
Int32	wavelength	The wavelength to get the reference for.

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Double>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
ArgumentException	Thrown when the specified detector is out of range for the device.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

GetILResolutionAsync(CancellationToken)

Asynchronously get the IL reading resolution.

Declaration

```
public async Task<EResolution> GetILResolutionAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
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
InvalidOperationException	Thrown when the method is called on an uninitialized device.
InvalidOperationException	Thrown when the RLM does not support IL resolution operations.
UnexpectedDeviceResponseException	Thrown when the device response does not correspond to a resolution.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

GetLengthModeAsync(CancellationToken)

Asynchronously get the length mode.

Declaration

```
public async Task<ELengthMode> GetLengthModeAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<ELengthMode>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized RLM.
UnexpectedDeviceResponseException	Thrown when the device response does not correspond to a length mode.
TimeoutException	Thrown when a communication failure causes no response to be received from the RLM.

GetOutputChannelAsync(CancellationToken)

Asynchronously get the current output channel of the laser source.

Declaration

```
public async Task<int> GetOutputChannelAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Int32>	The task representing the operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
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
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

GetPositionBModeAsync(CancellationToken)

Asynchronously get the RL position B mode.

Declaration

```
public async Task<EPositionBMode> GetPositionBModeAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<EPositionBMode>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the RLM.

GetRLModeAsync(CancellationToken)

Asynchronously get the RL sensitivity mode.

Declaration

```
public async Task<ERLMode> GetRLModeAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<ERLMode>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized RLM.
UnexpectedDeviceResponseException	Thrown when the device response does not correspond to an RL mode.
TimeoutException	Thrown when a communication failure causes no response to be received from the RLM.

GetRLReferenceAsync(CancellationToken)

Asynchronously get the stored RL reference.

Declaration

```
public async Task<double> GetRLReferenceAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Double>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the RLM.

GetSwitchChannelAsync(Int32, CancellationToken)

Asynchronously get the channel of a connected switch.

Declaration

```
public async Task<int> GetSwitchChannelAsync(int switchIndex, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
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.
CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
Task<Int32>	The task representing the operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device has no switch at the specified switch index.

TYPE	CONDITION
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

GetTraceDiagnosticsAsync(CancellationToken)

Asynchronously get the trace diagnostics for the last RL measurement.

Declaration

```
public Task<TraceDiagnostics> GetTraceDiagnosticsAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<TraceDiagnostics>	The task representing the operation.

Remarks

This is currently not implemented.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized RLM.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the RLM.

TYPE	CONDITION
NotSupportedException	Thrown when the method is called on an RLM through a USB connection.

ReadILAsync(Int32, Int32, CancellationToken)

Asynchronously read the insertion loss from a detector at a given wavelength.

Declaration

```
public async Task<double> ReadILAsync(int detector, int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	The detector number of the detector to read the insertion loss from.
Int32	wavelength	The wavelength to read the insertion loss at.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Double>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
ArgumentException	Thrown when the specified detector is out of range for the device.
InvalidOperationException	Thrown when the method is called on an uninitialized device.

TYPE	CONDITION
UnexpectedDeviceResponseException	Thrown when the device read IL response is not a valid number.
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

```
public async Task<double> ReadMonitorPowerAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	wavelength	The wavelength to read the monitor power at.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Double>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device monitor power response is not a valid number.

TYPE	CONDITION
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

ReadMultipleILsAsync(IEnumerable<Int32>, Int32, CancellationToken)

Asynchronously read the insertion loss from multiple detectors at a given wavelength.

Declaration

```
public async Task<List<double>> ReadMultipleILsAsync(IEnumerable<int> detectors, int wavelength,
CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
IEnumerable<Int32>	detectors	The detectors to read the insertion loss from.
Int32	wavelength	The wavelength to read the insertion loss at.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<List<Double>>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
ArgumentException	Thrown when any of the specified detectors are out of range for the device.
InvalidOperationException	Thrown when the method is called on an uninitialized device.

TYPE	CONDITION
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.
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

```
public async Task<List<double>> ReadMultiplePowersAsync(IEnumerable<int> detectors, int wavelength,
CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
IEnumerable<Int32>	detectors	The detectors to read from.
Int32	wavelength	The wavelength to read the power at.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<List<Double>>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.

TYPE	CONDITION
ArgumentException	Thrown when any of the specified detectors are out of range for the device.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

ReadPowerAsync(Int32, Int32, CancellationToken)

Asynchronously read the power from an device detector at a given wavelength.

Declaration

```
public async Task<double> ReadPowerAsync(int detector, int wavelength, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	The detector number of the detector to read from.
Int32	wavelength	The wavelength to read the power at.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Double>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
ArgumentException	Thrown when the specified detector is out of range for the device.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device read power response is not a valid number.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

ReadRLAsync(Int32, Double, Double, CancellationToken)

Asynchronously read the return loss at a given wavelength using the specified reference lengths.

Declaration

```
public async Task<RLReading> ReadRLAsync(int wavelength, double lengthReferenceA, double lengthReferenceB,
CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	wavelength	The wavelength to read the return loss at.
Double	lengthReferenceA	The reference length to use for position A.
Double	lengthReferenceB	The reference length to use for position B.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
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
ArgumentException	Thrown when the RLM does not support the specified wavelength.
ArgumentException	Thrown when either RL reference length is less than `0`.
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.
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

```
public async Task<RLReading> ReadRLAsync(int wavelength, double lengthReference, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	wavelength	The wavelength to read the return loss at.
Double	lengthReference	The reference length to use for the reading.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
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
ArgumentException	Thrown when the RLM does not support the specified wavelength.
ArgumentException	Thrown when the RLM reference length is less than `0`.
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.
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

```
public async Task<RLReading> ReadRLAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	wavelength	The wavelength to read the return loss at.

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
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
ArgumentException	Thrown when the RLM does not support the specified wavelength.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the RLM.

ReferenceILAsync(Int32, Int32, Boolean, CancellationToken)

Asynchronously perform an insertion loss reference for a detector at a given wavelength.

Declaration

```
public async Task<double> ReferenceILAsync(int detector, int wavelength, bool applyReferenceToAllDetectors,
CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION

TYPE	NAME	DESCRIPTION
Int32	detector	The detector number of the detector to perform the reference for.
Int32	wavelength	The wavelength to perform the reference at.
Boolean	applyReferenceToAllDetectors	Indicate if the reference read should be applied to all detectors for the current channel.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Double>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
ArgumentException	Thrown when the specified detector is out of range for the device.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device reference IL response is not a valid number.
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

```
public async Task<double> ReferenceILAsync(int detector, int wavelength, CancellationToken cancellationToken
= default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	The detector number of the detector to perform the reference for.
Int32	wavelength	The wavelength to perform the reference at.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Double>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
ArgumentException	Thrown when the specified detector is out of range for the device.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device reference IL response is not a valid number.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

ReferenceRLAsync(CancellationToken)

Asynchronously perform a return loss length reference.

Declaration

```
public async Task<double> ReferenceRLAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Double>	The task representing the operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized RLM.
UnexpectedDeviceResponseException	Thrown when the device reference RL response is not a valid number.
TimeoutException	Thrown when a communication failure causes no response to be received from the RLM.

SetDUTILAsync(Double, CancellationToken)

Asynchronously set the DUT IL.

Declaration

```
public async Task SetDUTILAsync(double il, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Double	il	The DUT IL value.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
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
ArgumentException	Thrown when the specified DUT IL is less than `0`.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the RLM.

SetGainModeAsync(EGainMode, CancellationToken)

Asynchronously set the gain mode.

Declaration

```
public async Task SetGainModeAsync(EGainMode gainMode, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
EGainMode	gainMode	The gain mode to change to.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the RLM.

SetILReferenceAsync(Int32, Int32, Double, CancellationToken)

Asynchronously set the IL reference for a detector at a given wavelength.

Declaration

```
public async Task SetILReferenceAsync(int detector, int wavelength, double reference, CancellationToken
cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	detector	The detector to set the reference for.
Int32	wavelength	The wavelength to set the reference for.
Double	reference	The IL reference value.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
ArgumentException	Thrown when the specified detector is out of range for the device.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

SetILResolutionAsync(EResolution, CancellationToken)

Asynchronously set the IL reading resolution.

Declaration

```
public async Task SetILResolutionAsync(EResolution resolution, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
EResolution	resolution	The resolution to set the detector to.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
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
ArgumentException	Thrown when the IL resolution is not supported by the RLM.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

SetLengthModeAsync(ELengthMode, CancellationToken)

Asynchronously set the length mode.

Declaration

```
public async Task SetLengthModeAsync(ELengthMode lengthMode, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
ELengthMode	lengthMode	The length mode to change to.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
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.
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

```
public async Task SetPositionBModeAsync(EPositionBMode positionBMode, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
EPositionBMode	positionBMode	The position B mode to change to.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized RLM.
UnexpectedDeviceResponseException	Thrown when the device positionB mode response does not match the expected response.

TYPE	CONDITION
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

```
public async Task SetRLModeAsync(ERLMode rlMode, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
ERLMode	rlMode	The RL mode to change to.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized RLM.
UnexpectedDeviceResponseException	Thrown when the device RL mode response does not match the expected response.
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

```
public async Task SetRLReferenceAsync(double reference, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Double	reference	The RL length reference value.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
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
ArgumentException	Thrown when the RL length reference value is less than `0`.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the RLM.

TurnOffLaserAsync(CancellationToken)

Asynchronously turn off the active laser.

Declaration

```
public async Task TurnOffLaserAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device laser response does not match the expected response.
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

```
public async Task TurnOnLaserAsync(int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	wavelength	The wavelength of the laser to turn on.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device laser response does not match the expected response.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

Implements

IRLM
ISantecDevice
IDiscretePowerMeter
ISwitchController
IFiberDevice
IPowerMeter
IDetectorDevice
ILaserSource
IWavelengthDevice
IDevice
System.IDisposable

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

Object
TraceDiagnosticInfo

Inherited Members

Object.ToString()
Object.Equals(Object)
Object.Equals(Object, Object)
Object.ReferenceEquals(Object, Object)
Object.GetHashCode()
Object.GetType()
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
Int32	The acquisition postDelay.

AcquisitionPreDelay

Get/Set the acquisition preDelay.

Declaration

```
[JsonPropertyName("acqPreDelay")]
public int AcquisitionPreDelay { get; set; }
```

Property Value

TYPE	DESCRIPTION
Int32	The acquisition preDelay.

AcquisitionSample

Get/Set the acquisition sample.

Declaration

```
[JsonPropertyName("acqSample")]
public int AcquisitionSample { get; set; }
```

Property Value

TYPE	DESCRIPTION
Int32	The acquisition sample.

APDBias

Get/Set the APD bias.

Declaration

```
public int APDBias { get; set; }
```

Property Value

TYPE	DESCRIPTION
Int32	The APD bias.

BaseIOR

Get/Set the baseIOR.

Declaration

```
[JsonPropertyName("baseIOR")]
public double BaseIOR { get; set; }
```

Property Value

TYPE	DESCRIPTION
Double	The baseIOR.

DeltaD

Get/Set the deltaD of the measurement.

Declaration

```
[JsonPropertyName("deltaD")]
public double DeltaD { get; set; }
```

Property Value

TYPE	DESCRIPTION
Double	The deltaD.

DeltaT

Get/Set the deltaT of the measurement.

Declaration

```
[JsonPropertyName("deltaT")]
public double DeltaT { get; set; }
```

Property Value

TYPE	DESCRIPTION
Double	The deltaT.

Dither

Get/Set the dither value.

Declaration

```
[JsonPropertyName("ditherVal")]
public int Dither { get; set; }
```

Property Value

TYPE	DESCRIPTION
Int32	The dither value.

DitherMode

Get/Set the dither mode.

Declaration

```
public int DitherMode { get; set; }
```

Property Value

TYPE	DESCRIPTION
Int32	The dither mode.

DUTil

Get/Set the DUT IL used in the measurement.

Declaration

```
public double DUTil { get; set; }
```

Property Value

TYPE	DESCRIPTION
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
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
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
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")]
public int EndOfFiberPeakIndex { get; set; }
```

Property Value

TYPE	DESCRIPTION
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
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
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
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
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
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

TYPE	DESCRIPTION
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
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
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
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
Int32	The number of phases.

OffPostDelay

Get/Set the offPostDelay.

Declaration

```
[JsonPropertyName("offPostDelay")]
public int OffPostDelay { get; set; }
```

Property Value

TYPE	DESCRIPTION
Int32	The offPostDelay.

OffPreDelay

Get/Set the offPreDelay.

Declaration

```
[JsonPropertyName("offPreDelay")]
public int OffPreDelay { get; set; }
```

Property Value

TYPE	DESCRIPTION
Int32	The offPreDelay.

OffSample

Get/Set the offSample.

Declaration

```
[JsonPropertyName("offSample")]
public int OffSample { get; set; }
```

Property Value

TYPE	DESCRIPTION
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
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
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
Double	The refPeakHeightStart.

RLa

Get/Set the RLa for the measurement.

Declaration

```
[JsonPropertyName("rlAValue")]
public double RLa { get; set; }
```

Property Value

TYPE	DESCRIPTION
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
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
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
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
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
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
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
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
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
Int32	The index of the RLb reference point.

RLMType

Get/Set the RLM type.

Declaration

```
[JsonPropertyName("moduleType")]
public ERLMType RLMType { get; set; }
```

Property Value

TYPE	DESCRIPTION

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
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
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
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
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
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
Int32	The start phase.

ZeroPositionIndex

Get/Set the index of the zero position.

Declaration

```
public int ZeroPositionIndex { get; set; }
```

Property Value

TYPE	DESCRIPTION
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

Object
TraceDiagnostics

Inherited Members

Object.ToString()
Object.Equals(Object)
Object.Equals(Object, Object)
Object.ReferenceEquals(Object, Object)
Object.GetHashCode()
Object.GetType()
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
List<Double>	The trace data.

Namespace Santec.Hardware.Devices.Source

Interfaces

ILaserSource

Defines the properties of a laser source and the operations that can be performed with a connection to the device.

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.ResetAsync()
IDevice.RefreshAsync()
IDevice.PingAsync()
IDevice.QueryAsync(String, CancellationToken)
IDevice.WriteAsync(String, CancellationToken)
IDevice.ReadAsync(CancellationToken)
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
Int32	The number of device outputs.

Exceptions

TYPE	CONDITION
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
Int32	output	The output channel to switch to.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified output channel is out of range for the laser source.
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.
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
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<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
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

GetFactoryPowerAsync(Int32, Int32, CancellationToken)

Asynchronously get the factory power for the specified output channel and wavelength.

Declaration

```
Task<double> GetFactoryPowerAsync(int output, int wavelength, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	output	The output channel to get the factory power for.
Int32	wavelength	The wavelength to get the factory power for.

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Double>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified output channel is out of range.
ArgumentException	Thrown when the device does not support the specified wavelength.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device factory power response is not a valid number.
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
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Int32>	The task representing the operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
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.
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
Int32	wavelength	The wavelength to read the monitor power at.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Double>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device monitor power response is not a valid number.
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
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device laser response does not match the expected response.

TYPE	CONDITION
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
Int32	wavelength	The wavelength of the laser to turn on.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device does not support the specified wavelength.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device laser response does not match the expected response.
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.

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.

SwitchModule

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.

Class DeviceSwitch

Represents a switch connected to a ISwitchController.

Inheritance

Object
DeviceSwitch
ExternalDeviceSwitch

Inherited Members

Object.ToString()
Object.Equals(Object)
Object.Equals(Object, Object)
Object.ReferenceEquals(Object, Object)
Object.GetHashCode()
Object.GetType()
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
Int32	The switch index.

NumberOfChannels

Get the number of the channels the switch has.

Declaration

```
public int NumberOfChannels { get; }
```

Property Value

TYPE	DESCRIPTION
Int32	The number of channels the switch has.

Class DeviceSwitchAndChannelPair

Represents a device switch and a channel on the switch.

Inheritance

Object
DeviceSwitchAndChannelPair

Inherited Members

Object.ToString()
Object.Equals(Object)
Object.Equals(Object, Object)
Object.ReferenceEquals(Object, Object)
Object.GetHashCode()
Object.GetType()
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
Int32	deviceSwitchIndex	
Int32	channel	

Properties

Channel

Get the switch channel.

Declaration

```
public int Channel { get; set; }
```

Property Value

TYPE	DESCRIPTION
Int32	The switch channel.

DeviceSwitchIndex

Get the switch index of the switch.

Declaration

```
public int DeviceSwitchIndex { get; set; }
```

Property Value

TYPE	DESCRIPTION
Int32	The switch index of the switch.

Class ExternalDeviceSwitch

Represents an external switch connected to a ISwitchController.

Inheritance

Object
DeviceSwitch
ExternalDeviceSwitch

Inherited Members

DeviceSwitch.Index
DeviceSwitch.NumberOfChannels
Object.ToString()
Object.Equals(Object)
Object.Equals(Object, Object)
Object.ReferenceEquals(Object, Object)
Object.GetHashCode()
Object.GetType()
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
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
IModuleController.GetSelectedModuleAsync(CancellationToken)
IModuleController.SetSelectedModuleAsync(Int32, CancellationToken)
IDevice.Name
IDevice.SerialNumber
IDevice.FirmwareVersion
IDevice.Address
IDevice.IsInitialized
IDevice.InitializeAsync()
IDevice.Uninitialize()
IDevice.ResetAsync()
IDevice.RefreshAsync()
IDevice.PingAsync()
IDevice.QueryAsync(String, CancellationToken)
IDevice.WriteAsync(String, CancellationToken)
IDevice.ReadAsync(CancellationToken)
IDisposable.Dispose()

Namespace: Santec.Hardware.Devices.Switching

Assembly: Santec.Hardware.dll

Syntax

```
public interface IOSX : ISantecDevice, IModuleController, 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<SwitchModule> Modules { get; }
```

Property Value

TYPE	DESCRIPTION
IReadOnlyList<SwitchModule>	A list of the modules the switch contains.

Exceptions

TYPE	CONDITION
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
Int32	channel	The channel to switch all modules to.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified channel is out of range for any of the OSX modules.
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
TimeoutException	Thrown when a communication failure causes no response to be received from the OSX.

ChangeChannelAsync(Int32, CancellationToken)

Asynchronously change the channel of the currently selected module.

Declaration

```
Task ChangeChannelAsync(int channel, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	channel	The channel to switch to.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified channel is out of range for the OSX.
InvalidOperationException	Thrown when the method is called on an uninitialized OSX.
UnexpectedDeviceResponseException	Thrown when the OSX channel response does not match the expected response.
TimeoutException	Thrown when a communication failure causes no response to be received from the OSX.

ChangeCommonAsync(Int32, CancellationToken)

Asynchronously change the common channel of the currently selected module.

Declaration

```
Task ChangeCommonAsync(int common, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	common	The common channel to switch to.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified common channel is out of range for the current OSX module.
InvalidOperationException	Thrown when the method is called on an uninitialized OSX.
UnexpectedDeviceResponseException	Thrown when the OSX common channel response does not match the expected response.
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
Int32	moduleIndex	The index of the module to change the channel of.
Int32	channel	The channel to switch to.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified module is out of range for the OSX.
ArgumentException	Thrown when the specified channel is out of range for the OSX module.
InvalidOperationException	Thrown when the method is called on an uninitialized OSX.
UnexpectedDeviceResponseException	Thrown when the OSX module channel response does not match the expected response.
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
Int32	moduleIndex	The index of the module to change the common channel of.
Int32	common	The common channel to switch to.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified module is out of range for the OSX.
ArgumentException	Thrown when the specified common channel is out of range for the OSX module.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the OSX.

GetChannelAsync(CancellationToken)

Asynchronously get the channel of the currently selected module.

Declaration

```
Task<int> GetChannelAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Int32>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized OSX.
UnexpectedDeviceResponseException	Thrown when the OSX channel response is not a valid channel.
TimeoutException	Thrown when a communication failure causes no response to be received from the OSX.

GetCommonAsync(CancellationToken)

Asynchronously get the common channel of the currently selected module.

Declaration

```
Task<int> GetCommonAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Int32>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized OSX.
UnexpectedDeviceResponseException	Thrown when the OSX common channel response is not a valid channel.
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
Int32	moduleIndex	The index of the module to get the current channel for.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Int32>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified module is out of range for the OSX.

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized OSX.
UnexpectedDeviceResponseException	Thrown when the OSX module channel response is not a valid module channel.
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
Int32	moduleIndex	The index of the module to get the current common channel for.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Int32>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified module is out of range for the OSX.
InvalidOperationException	Thrown when the method is called on an uninitialized OSX.

TYPE	CONDITION
UnexpectedDeviceResponseException	Thrown when the OSX module common channel response is not a valid module channel.
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.ResetAsync()
IDevice.RefreshAsync()
IDevice.PingAsync()
IDevice.QueryAsync(String, CancellationToken)
IDevice.WriteAsync(String, CancellationToken)
IDevice.ReadAsync(CancellationToken)
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
List<DeviceSwitch>	The list of the switches connected to the device.

Exceptions

TYPE	CONDITION
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
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.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device has no switch at any of the specified switch indexes.
ArgumentException	Thrown when any the specified channels is out of range for the corresponding device switch.
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.
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
Int32	switchIndex	The switch number of the switch. `0` for the internal switch; `1` or `2` for external switches attached to the device.
Int32	channel	The channel to switch to.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device has no switch at the specified switch index.
ArgumentException	Thrown when the specified channel is out of range for the device switch.
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.
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
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.
CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
Task<Int32>	The task representing the operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the device has no switch at the specified switch index.
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.
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

Object
PhysicalDevice
SantecDevice
OSX

Implements

IOSX
ISantecDevice
IModuleController
IDevice
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.Dispose()
PhysicalDevice.PingAsync()
PhysicalDevice.ResetAsync()
PhysicalDevice.QueryAsync(String, CancellationToken)
PhysicalDevice.WriteAsync(String, CancellationToken)
PhysicalDevice.ReadAsync(CancellationToken)
Object.ToString()
Object.Equals(Object)
Object.Equals(Object, Object)
Object.ReferenceEquals(Object, Object)
Object.GetHashCode()

Object.GetType()
Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.Switching

Assembly: Santec.Hardware.dll

Syntax

```
public class OSX : SantecDevice, IOSX, ISantecDevice, IModuleController, IDevice, IDisposable
```

Properties

Modules

Get the modules of the switch.

> **NOTE**
>
> The OSX must be initialized before this property contains the proper values.

Declaration

```
public IReadOnlyList<SwitchModule> Modules { get; }
```

Property Value

TYPE	DESCRIPTION
IReadOnlyList<SwitchModule>	A list of the modules the switch contains.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the property is accessed before the OSX has been initialized.

Name

Get the name of the device.

Declaration

```
public override string Name { get; }
```

Property Value

TYPE	DESCRIPTION
String	The name of the device.

Overrides

PhysicalDevice.Name

Remarks

> **NOTE**
>
> This property will always return "OSX".

NumberOfModules

Get the number of modules in the device.

> **NOTE**
>
> The device must be initialized before this property contains the proper value.

Declaration

```
public int NumberOfModules { get; }
```

Property Value

TYPE	DESCRIPTION
Int32	The number of modules in the device.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the property is accessed before the device has been initialized.

Methods

ChangeAllModuleChannelsAsync(Int32, CancellationToken)

Asynchronously change the channel of all modules.

Declaration

```
public async Task ChangeAllModuleChannelsAsync(int channel, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	channel	The channel to switch all modules to.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified channel is out of range for any of the OSX modules.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the OSX.

ChangeChannelAsync(Int32, CancellationToken)

Asynchronously change the channel of the currently selected module.

Declaration

```
public async Task ChangeChannelAsync(int channel, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	channel	The channel to switch to.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified channel is out of range for the OSX.
InvalidOperationException	Thrown when the method is called on an uninitialized OSX.
UnexpectedDeviceResponseException	Thrown when the OSX channel response does not match the expected response.
TimeoutException	Thrown when a communication failure causes no response to be received from the OSX.

ChangeCommonAsync(Int32, CancellationToken)

Asynchronously change the common channel of the currently selected module.

Declaration

```
public async Task ChangeCommonAsync(int common, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	common	The common channel to switch to.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified common channel is out of range for the current OSX module.

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized OSX.
UnexpectedDeviceResponseException	Thrown when the OSX common channel response does not match the expected response.
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

```
public async Task ChangeModuleChannelAsync(int moduleIndex, int channel, CancellationToken cancellationToken
= default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	moduleIndex	The index of the module to change the channel of.
Int32	channel	The channel to switch to.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified module is out of range for the OSX.
ArgumentException	Thrown when the specified channel is out of range for the OSX module.

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized OSX.
UnexpectedDeviceResponseException	Thrown when the OSX module channel response does not match the expected response.
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

```
public async Task ChangeModuleCommonAsync(int moduleIndex, int common, CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	moduleIndex	The index of the module to change the common channel of.
Int32	common	The common channel to switch to.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified module is out of range for the OSX.
ArgumentException	Thrown when the specified common channel is out of range for the OSX module.

TYPE	CONDITION
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the OSX.

GetChannelAsync(CancellationToken)

Asynchronously get the channel of the currently selected module.

Declaration

```
public async Task<int> GetChannelAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Int32>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized OSX.
UnexpectedDeviceResponseException	Thrown when the OSX channel response is not a valid channel.
TimeoutException	Thrown when a communication failure causes no response to be received from the OSX.

GetCommonAsync(CancellationToken)

Asynchronously get the common channel of the currently selected module.

Declaration

```
public async Task<int> GetCommonAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Int32>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized OSX.
UnexpectedDeviceResponseException	Thrown when the OSX common channel response is not a valid channel.
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

```
public async Task<int> GetModuleChannelAsync(int moduleIndex, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	moduleIndex	The index of the module to get the current channel for.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Int32>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified module is out of range for the OSX.
InvalidOperationException	Thrown when the method is called on an uninitialized OSX.
UnexpectedDeviceResponseException	Thrown when the OSX module channel response is not a valid module channel.
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

```
public async Task<int> GetModuleCommonAsync(int moduleIndex, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	moduleIndex	The index of the module to get the current common channel for.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Int32>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified module is out of range for the OSX.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the OSX.

GetSelectedModuleAsync(CancellationToken)

Asynchronously get the currently selected module.

Declaration

```
public Task<int> GetSelectedModuleAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Int32>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized device.

TYPE	CONDITION
UnexpectedDeviceResponseException	Thrown when the device selected module response is not a valid module.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

InitializeAsync()

Asynchronously open the device connection and initialize the device settings.

Declaration

```
public override async Task InitializeAsync()
```

Returns

TYPE	DESCRIPTION
Task	The task representing the operation.

Overrides

SantecDevice.InitializeAsync()

Remarks

NOTE

This operation will not block.

RefreshAsync()

Asynchronously refresh the device settings.

Declaration

```
public override async Task RefreshAsync()
```

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Overrides

PhysicalDevice.RefreshAsync()

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized RLM.
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.

SetSelectedModuleAsync(Int32, CancellationToken)

Asynchronously set the currently selected module.

Declaration

```
public Task SetSelectedModuleAsync(int moduleIndex, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	moduleIndex	The index of the module to select.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified module is out of range for the device.
InvalidOperationException	Thrown when the method is called on an uninitialized device.

TYPE	CONDITION
UnexpectedDeviceResponseException	Thrown when the device select module response does not match the expected response.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

Uninitialize()

Synchronously uninitialize device settings and close the connection.

Declaration

```
public override void Uninitialize()
```

Overrides

PhysicalDevice.Uninitialize()

Implements

IOSX
ISantecDevice
IModuleController
IDevice
System.IDisposable

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

Object
SimulatedSantecDevice
SimulatedOSX

Implements

IOSX
ISantecDevice
IModuleController
IDevice
IDisposable

Inherited Members

SimulatedSantecDevice.creatingDevice
SimulatedSantecDevice.Dispose()
SimulatedSantecDevice.SerialNumber
SimulatedSantecDevice.FirmwareVersion
SimulatedSantecDevice.Address
SimulatedSantecDevice.IsInitialized
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)
Object.ToString()
Object.Equals(Object)
Object.Equals(Object, Object)
Object.ReferenceEquals(Object, Object)
Object.GetHashCode()

Object.GetType()
Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.Switching

Assembly: Santec.Hardware.dll

Syntax

```
public class SimulatedOSX : SimulatedSantecDevice, IOSX, ISantecDevice, IModuleController, IDevice,
IDisposable
```

Properties

Modules

Get the modules of the switch.

> **NOTE**
>
> The OSX must be initialized before this property contains the proper values.

Declaration

```
public IReadOnlyList<SwitchModule> Modules { get; }
```

Property Value

TYPE	DESCRIPTION
IReadOnlyList<SwitchModule>	A list of the modules the switch contains.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the property is accessed before the OSX has been initialized.

Name

Get the name of the device.

Declaration

```
public override string Name { get; }
```

Property Value

TYPE	DESCRIPTION
String	The name of the device.

Overrides

SimulatedSantecDevice.Name

Remarks

NOTE

This property will always return "OSX".

NumberOfModules

Get the number of modules in the device.

NOTE

The device must be initialized before this property contains the proper value.

Declaration

```
public int NumberOfModules { get; }
```

Property Value

TYPE	DESCRIPTION
Int32	The number of modules in the device.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the property is accessed before the device has been initialized.

Methods

ChangeAllModuleChannelsAsync(Int32, CancellationToken)

Asynchronously change the channel of all modules.

Declaration

```
public async Task ChangeAllModuleChannelsAsync(int channel, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	channel	The channel to switch all modules to.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified channel is out of range for any of the OSX modules.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the OSX.

ChangeChannelAsync(Int32, CancellationToken)

Asynchronously change the channel of the currently selected module.

Declaration

```
public async Task ChangeChannelAsync(int channel, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	channel	The channel to switch to.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified channel is out of range for the OSX.
InvalidOperationException	Thrown when the method is called on an uninitialized OSX.
UnexpectedDeviceResponseException	Thrown when the OSX channel response does not match the expected response.
TimeoutException	Thrown when a communication failure causes no response to be received from the OSX.

ChangeCommonAsync(Int32, CancellationToken)

Asynchronously change the common channel of the currently selected module.

Declaration

```
public async Task ChangeCommonAsync(int common, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	common	The common channel to switch to.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified common channel is out of range for the current OSX module.

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized OSX.
UnexpectedDeviceResponseException	Thrown when the OSX common channel response does not match the expected response.
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

```
public async Task ChangeModuleChannelAsync(int moduleIndex, int channel, CancellationToken
cancellationToken)
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	moduleIndex	The index of the module to change the channel of.
Int32	channel	The channel to switch to.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified module is out of range for the OSX.
ArgumentException	Thrown when the specified channel is out of range for the OSX module.

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized OSX.
UnexpectedDeviceResponseException	Thrown when the OSX module channel response does not match the expected response.
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

```
public async Task ChangeModuleCommonAsync(int moduleIndex, int common, CancellationToken cancellationToken)
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	moduleIndex	The index of the module to change the common channel of.
Int32	common	The common channel to switch to.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified module is out of range for the OSX.
ArgumentException	Thrown when the specified common channel is out of range for the OSX module.

TYPE	CONDITION
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the OSX.

GetChannelAsync(CancellationToken)

Asynchronously get the channel of the currently selected module.

Declaration

```
public async Task<int> GetChannelAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Int32>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized OSX.
UnexpectedDeviceResponseException	Thrown when the OSX channel response is not a valid channel.
TimeoutException	Thrown when a communication failure causes no response to be received from the OSX.

GetCommonAsync(CancellationToken)

Asynchronously get the common channel of the currently selected module.

Declaration

```
public async Task<int> GetCommonAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Int32>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized OSX.
UnexpectedDeviceResponseException	Thrown when the OSX common channel response is not a valid channel.
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

```
public async Task<int> GetModuleChannelAsync(int moduleIndex, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	moduleIndex	The index of the module to get the current channel for.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Int32>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified module is out of range for the OSX.
InvalidOperationException	Thrown when the method is called on an uninitialized OSX.
UnexpectedDeviceResponseException	Thrown when the OSX module channel response is not a valid module channel.
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

```
public async Task<int> GetModuleCommonAsync(int moduleIndex, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	moduleIndex	The index of the module to get the current common channel for.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Int32>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified module is out of range for the OSX.
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.
TimeoutException	Thrown when a communication failure causes no response to be received from the OSX.

GetSelectedModuleAsync(CancellationToken)

Asynchronously get the currently selected module.

Declaration

```
public async Task<int> GetSelectedModuleAsync(CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task<Int32>	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
InvalidOperationException	Thrown when the method is called on an uninitialized device.

TYPE	CONDITION
UnexpectedDeviceResponseException	Thrown when the device selected module response is not a valid module.
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

SetSelectedModuleAsync(Int32, CancellationToken)

Asynchronously set the currently selected module.

Declaration

```
public async Task SetSelectedModuleAsync(int moduleIndex, CancellationToken cancellationToken =
default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	moduleIndex	The index of the module to select.
CancellationToken	cancellationToken	The cancellation token for the asynchronous operation. The default value is `None`.

Returns

TYPE	DESCRIPTION
Task	The task representing the asynchronous operation.

Remarks

NOTE

This operation will not block.

Exceptions

TYPE	CONDITION
ArgumentException	Thrown when the specified module is out of range for the device.
InvalidOperationException	Thrown when the method is called on an uninitialized device.
UnexpectedDeviceResponseException	Thrown when the device select module response does not match the expected response.

TYPE	CONDITION
TimeoutException	Thrown when a communication failure causes no response to be received from the device.

Implements

IOSX
ISantecDevice
IModuleController
IDevice
System.IDisposable

Class SwitchModule

Inheritance

Object

SwitchModule

Inherited Members

Object.ToString()

Object.Equals(Object)

Object.Equals(Object, Object)

Object.ReferenceEquals(Object, Object)

Object.GetHashCode()

Object.GetType()

Object.MemberwiseClone()

Namespace: Santec.Hardware.Devices.Switching

Assembly: Santec.Hardware.dll

Syntax

```
public class SwitchModule
```

Constructors

SwitchModule(Int32, Int32)

Declaration

```
public SwitchModule(int numberOfCommons, int numberOfChannels)
```

Parameters

TYPE	NAME	DESCRIPTION
Int32	numberOfCommons	
Int32	numberOfChannels	

Properties

NumberOfChannels

Declaration

```
public int NumberOfChannels { get; }
```

Property Value

TYPE	DESCRIPTION
Int32	

NumberOfCommons

Declaration

```
public int NumberOfCommons { get; }
```

Property Value

TYPE	DESCRIPTION
Int32	

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.

PolarityMappingFailedException

The exception that is thrown when a device cannot determine a mapping of outputs to inputs.

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

Object
Exception
AmbientLightDetectedException

Implements

ISerializable
_Exception

Inherited Members

Exception.GetBaseException()
Exception.ToString()
Exception.GetObjectData(SerializationInfo, StreamingContext)
Exception.GetType()
Exception.Message
Exception.Data
Exception.InnerException
Exception.TargetSite
Exception.StackTrace
Exception.HelpLink
Exception.Source
Exception.HResult
Exception.SerializeObjectState
Object.Equals(Object)
Object.Equals(Object, Object)
Object.ReferenceEquals(Object, Object)
Object.GetHashCode()
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
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
String	message	The error message.
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
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

Object
Exception
CameraNotFoundException

Implements

ISerializable
_Exception

Inherited Members

Exception.GetBaseException()
Exception.ToString()
Exception.GetObjectData(SerializationInfo, StreamingContext)
Exception.GetType()
Exception.Message
Exception.Data
Exception.InnerException
Exception.TargetSite
Exception.StackTrace
Exception.HelpLink
Exception.Source
Exception.HResult
Exception.SerializeObjectState
Object.Equals(Object)
Object.Equals(Object, Object)
Object.ReferenceEquals(Object, Object)
Object.GetHashCode()
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
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
String	message	The error message.
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

Object
Exception
CompositeDeviceHardwareException

Implements

ISerializable
_Exception

Inherited Members

Exception.GetBaseException()
Exception.ToString()
Exception.GetObjectData(SerializationInfo, StreamingContext)
Exception.GetType()
Exception.Message
Exception.Data
Exception.InnerException
Exception.TargetSite
Exception.StackTrace
Exception.HelpLink
Exception.Source
Exception.HResult
Exception.SerializeObjectState
Object.Equals(Object)
Object.Equals(Object, Object)
Object.ReferenceEquals(Object, Object)
Object.GetHashCode()
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
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
String	message	The error message.
Exception	innerException	The inner 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

Object
Exception
PolarityMappingFailedException

Implements

ISerializable
_Exception

Inherited Members

Exception.GetBaseException()
Exception.ToString()
Exception.GetObjectData(SerializationInfo, StreamingContext)
Exception.GetType()
Exception.Message
Exception.Data
Exception.InnerException
Exception.TargetSite
Exception.StackTrace
Exception.HelpLink
Exception.Source
Exception.HResult
Exception.SerializeObjectState
Object.Equals(Object)
Object.Equals(Object, Object)
Object.ReferenceEquals(Object, Object)
Object.GetHashCode()
Object.MemberwiseClone()

Namespace: Santec.Hardware.Exceptions

Assembly: Santec.Hardware.dll

Syntax

```
public class PolarityMappingFailedException : Exception, ISerializable, _Exception
```

Constructors

PolarityMappingFailedException(IEnumerable<Int32>, Mat)

Initializes a new instance of the PolarityMappingFailedException class.

Declaration

```
public PolarityMappingFailedException(IEnumerable<int> detectedFibers, Mat polarityMappingImage)
```

Parameters

TYPE	NAME	DESCRIPTION
IEnumerable<Int32>	detectedFibers	The fibers that were detected during the polarity mapping operation.

TYPE	NAME	DESCRIPTION
OpenCvSharp.Mat	polarityMappingImage	A composite image of the polarity mapping operation.

PolarityMappingFailedException(String, IEnumerable<Int32>, Mat)

Initializes a new instance of the PolarityMappingFailedException class with the specified error message.

Declaration

```
public PolarityMappingFailedException(string message, IEnumerable<int> detectedFibers, Mat
polarityMappingImage)
```

Parameters

TYPE	NAME	DESCRIPTION
String	message	The error message.
IEnumerable<Int32>	detectedFibers	The fibers that were detected during the polarity mapping operation.
OpenCvSharp.Mat	polarityMappingImage	A composite image of the polarity mapping operation.

PolarityMappingFailedException(String, Exception, IEnumerable<Int32>, 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, Mat polarityMappingImage)
```

Parameters

TYPE	NAME	DESCRIPTION
String	message	The error message.
Exception	innerException	The inner exception.
IEnumerable<Int32>	detectedFibers	The fibers that were detected during the polarity mapping operation.
OpenCvSharp.Mat	polarityMappingImage	A composite image of the polarity mapping operation.

Properties

DetectedFibers

Get the fibers that were detected during the polarity mapping operation.

Declaration

```
public IReadOnlyList<int> DetectedFibers { get; }
```

Property Value

TYPE	DESCRIPTION
IReadOnlyList<Int32>	A list of the fibers that were detected during the polarity mapping operation.

PolarityMappingImage

Get the composite image of the polarity mapping operation.

Declaration

```
public Mat PolarityMappingImage { 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.

Methods

Finalize()

Free the resources of the PolarityMappingFailedException.

Declaration

```
protected void Finalize()
```

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

Object
Exception
UnexpectedDeviceResponseException

Implements

ISerializable
_Exception

Inherited Members

Exception.GetBaseException()
Exception.ToString()
Exception.GetObjectData(SerializationInfo, StreamingContext)
Exception.GetType()
Exception.Message
Exception.Data
Exception.InnerException
Exception.TargetSite
Exception.StackTrace
Exception.HelpLink
Exception.Source
Exception.HResult
Exception.SerializeObjectState
Object.Equals(Object)
Object.Equals(Object, Object)
Object.ReferenceEquals(Object, Object)
Object.GetHashCode()
Object.MemberwiseClone()

Namespace: Santec.Hardware.Exceptions

Assembly: Santec.Hardware.dll

Syntax

```
public class UnexpectedDeviceResponseException : Exception, ISerializable, _Exception
```

Constructors

UnexpectedDeviceResponseException()

Initializes a new instance of the UnexpectedDeviceResponseException class.

Declaration

```
public UnexpectedDeviceResponseException()
```

UnexpectedDeviceResponseException(String)

Initializes a new instance of the UnexpectedDeviceResponseException class with the specified error message.

Declaration

```
public UnexpectedDeviceResponseException(string message)
```

Parameters

TYPE	NAME	DESCRIPTION
String	message	The error message.

UnexpectedDeviceResponseException(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(string message, Exception innerException)
```

Parameters

TYPE	NAME	DESCRIPTION
String	message	The error message.
Exception	innerException	The inner 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

Object
Exception
VisaNotFoundException

Implements

ISerializable
_Exception

Inherited Members

Exception.GetBaseException()
Exception.ToString()
Exception.GetObjectData(SerializationInfo, StreamingContext)
Exception.GetType()
Exception.Message
Exception.Data
Exception.InnerException
Exception.TargetSite
Exception.StackTrace
Exception.HelpLink
Exception.Source
Exception.HResult
Exception.SerializeObjectState
Object.Equals(Object)
Object.Equals(Object, Object)
Object.ReferenceEquals(Object, Object)
Object.GetHashCode()
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
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
String	message	The error message.
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

Object
ConverterFactory

Inherited Members

Object.ToString()
Object.Equals(Object)
Object.Equals(Object, Object)
Object.ReferenceEquals(Object, Object)
Object.GetHashCode()
Object.GetType()
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

Object

HardwareDetectionService

Implements

IHardwareDetectionService

Inherited Members

Object.ToString()

Object.Equals(Object)

Object.Equals(Object, Object)

Object.ReferenceEquals(Object, Object)

Object.GetHashCode()

Object.GetType()

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

Get whether the hardware detection service can find a VISA implementation and detect USB devices.

Declaration

```
public bool CanDetectUSBDevices { get; }
```

Property Value

TYPE	DESCRIPTION
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

```
public bool CheckForVISA()
```

Returns

TYPE	DESCRIPTION
Boolean	`true` if a VISA implementation is found; otherwise, `false`.

DetectHardwareAsync(EConnectionTypesToDetect)

Asynchronously detect Santec Canada hardware devices.

Declaration

```
public async 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
Task<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

```
public async 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
Task<List<T>>	The task representing the asynchronous operation.

Type Parameters

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

```
public async Task<(bool Success, IDevice DetectedDevice)> TryDetectIPDeviceAsync(IPAddress ipAddress)
```

Parameters

TYPE	NAME	DESCRIPTION
IPAddress	ipAddress	The IP address to probe for the device.

Returns

TYPE	DESCRIPTION
Task<(T1, T2)<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

```
public async Task<(bool Success, T DetectedDevice)> TryDetectIPDeviceAsync<T>(IPAddress ipAddress)
    where T : class, IDevice
```

Parameters

TYPE	NAME	DESCRIPTION
IPAddress	ipAddress	The IP address to probe for the device.

Returns

TYPE	DESCRIPTION
Task<(T1, T2)<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.

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
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
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
Task<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
Task<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
IPAddress	ipAddress	The IP address to probe for the device.

Returns

TYPE	DESCRIPTION
Task<(T1, T2)<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
IPAddress	ipAddress	The IP address to probe for the device.

Returns

TYPE	DESCRIPTION
Task<(T1, T2)<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

Object
SimulatedHardwareDetectionService

Implements

IHardwareDetectionService

Inherited Members

Object.ToString()
Object.Equals(Object)
Object.Equals(Object, Object)
Object.ReferenceEquals(Object, Object)
Object.GetHashCode()
Object.GetType()
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

Get whether the hardware detection service can find a VISA implementation and detect USB devices.

Declaration

```
public bool CanDetectUSBDevices { get; }
```

Property Value

TYPE	DESCRIPTION
Boolean	`true` if the service can detect USB devices; otherwise, `false`.

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
Boolean	`true` if the service will detect real devices; otherwise, `false`.

Methods

CheckForVISA()

Check for a VISA implementation on the system and update CanDetectUSBDevices accordingly.

Declaration

```
public bool CheckForVISA()
```

Returns

TYPE	DESCRIPTION
Boolean	`true` if a VISA implementation is found; otherwise, `false`.

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
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
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
Boolean	overrideUSBDetection	If `true`, the service will be configured to override the USB detection responses; otherwise, USB detection will occur normally.
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)

Asynchronously detect Santec Canada hardware devices.

Declaration

```
public async 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
Task<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

```
public async 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
Task<List<T>>	The task representing the asynchronous operation.

Type Parameters

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

```
public async Task<(bool Success, IDevice DetectedDevice)> TryDetectIPDeviceAsync(IPAddress ipAddress)
```

Parameters

TYPE	NAME	DESCRIPTION
IPAddress	ipAddress	The IP address to probe for the device.

Returns

TYPE	DESCRIPTION
Task<(T1, T2)<Boolean, IDevice>>	The task representing the asynchronous operation.

Remarks

> **NOTE**
>
> This operation will not block.

> **NOTE**
>
> This method will never return success if DetectRealHardware is `false`.

TryDetectIPDeviceAsync<T>(IPAddress)

Asynchronously try to detect a Santec Canada device of a specific type at a specific IP address.

Declaration

```
public async Task<(bool Success, T DetectedDevice)> TryDetectIPDeviceAsync<T>(IPAddress ipAddress)
    where T : class, IDevice
```

Parameters

TYPE	NAME	DESCRIPTION
IPAddress	ipAddress	The IP address to probe for the device.

Returns

TYPE	DESCRIPTION
Task<(T1, T2)<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.

NOTE

This method will never return success if DetectRealHardware is `false`.

Implements

IHardwareDetectionService