SASS Documentation

Release 0.9.0-SNAPSHOT

Marcel Stefko, Kyle M. Douglass

Jun 05, 2018

Contents

Super-resolution Acquisition Simulation Software

CHAPTER 1

Quickstart

Contents

1.1 Installation

SASS is both a standalone application and a Fiji plugin.

1.1.1 Standalone

1. Download the latest .jar file from the SASS releases page.
2. You will also need to download the latest ALICA_ACPack .jar, which contains the run-time components for control systems simulations.
3. Place both .jars in the folder of your choosing.

1.1.2 Fiji

1. Download the latest .jar file from the SASS releases page.
2. You will also need to download the latest ALICA_ACPack .jar, which contains the run-time components for control systems simulations.
3. Copy the SASS .jar file into your *~/Fiji.app/plugins/* folder, where *Fiji.app* is root directory for your Fiji installation.
4. Copy the ALICA_ACPack .jar file into your *~/Fiji.app/jars* folder.
5. Restart Fiji.

You should now see *SASS* appear as a menu item in the the *Plugins* menu.

1.2 Run a simulation

1.2.1 Standalone

Before starting, make sure that you have a copy of the file example_random_2d_fluorophores.bsh from the SASS respository's *scripts* folder. When using SASS in standalone mode, it is most commonly used as a command line application.

1. From the command line, navigate to the folder where you placed the SASS .jar file that you downloaded in the installation step.
2. Enter the command *java -jar SASS_-<VERSION>.jar -s example_random_2d_fluorophores.bsh*.
3. If you want to save the simulation's output, ensure that any call to the *saveStack(...)* method is uncommented inside the script and rerun the simulation.

1.2.2 Fiji

1. Launch Fiji. (If you're launch Fiji from the command line, ensure that you are first in the Fiji root directory.)
2. Navigate to *Plugins* > *SASS* > *Simulator*.
3. Ensure that **Manual** is selected in the *Controller* drop-down box.
4. Click the *Initialize* button.
5. Rearrange the windows so that you can find the dialog with the controller set point and the *Start* and *Stop* buttons.
6. Click *Start* to start the simulation. You should see images begin streaming into the simulation's image stack.
7. Click the *Stop* button to pause the simulation.
8. Change the *Controller setpoint* value and click *Start* again to resume the simulation with a new laser power.

CHAPTER 2

Simulation Models

2.1 Fluorescence dynamics

The fluorescence dynamics in SASS are modeled as memoryless state systems. Such systems are comprised of two or more states that a fluorophore may occupy at any given time. During the course of an experiment, the fluorophore may randomly transition from its current state m to a new state n, and the probability with which this transition occurs is determined partly by the so-called rate constant k_{mn}.

Memorylessness means that the probability to transition to any accessible state does not depend on the time that the fluorophore has already spent in its current state. This assumption is well-founded: it is unlikely that a fluorescent molecule possesses some mechanism to keep track of time. Under the assumption of memorylessness, the length of the time interval t that is spent by a fluorophore in its current state S_m before making a transition to state S_n is given by an exponential probability density function

$$p_{mn}(t) = k_{mn} e^{-k_{mn} t}$$

When multiple states are accessible from S_m, then it may be shown that the probability that the fluorophore will have transitioned to the specific state S_n is

$$P(S_n, t = \infty | S_m, t = 0) = \frac{k_{mn}}{K}$$

where $K \equiv \sum_n k_{mn}$. Thus, the rate constants determine the relative probabilities of the transitions to different states.

2.1.1 Algorithm for state system simulations

The algorithm for simulating the state transitions proceeds as follows:

1. The fluorescent molecule is assigned a pre-defined starting state S_m.
2. Next, a random transition time from the molecule's current state is drawn for each accessible state n from an exponential distribution, $\forall n : t_{mn} \sim \text{Exp}(\tau_{mn})$ where $\tau_{mn} \equiv 1/k_{mn}$ is the average of the distribution.
3. The smallest value from this set of transition times is computed and stored as the molecule's transition time $T \equiv \text{Min}(t_{mn})$. The corresponding molecular state S_n is stored for use in the next step.

4. The simulation time is advanced one time step. If, during this time, a total amount of time has elapsed that is greater than the previously calculated transition time T, then the molecule is transitioned into its next state. The new next state and its transition time are generated and stored in the manner just described.
5. This process is repeated as the simulation continues until a pre-determined number of time steps have occurred or it is stopped by the user.

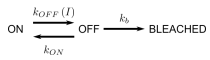
2.1.2 Non-stationary state transitions

In PALM/STORM type experiments, one or more rate constants depend on the light irradiance (power per area) of one or more light sources. Indeed, adjusting the power during an acquisition is a common way to optimize the quality of datasets derived from such experiments because it offers a direct way to tune the density of fluorophores in a light-emitting state.

When the laser irradiance varies with time, so too do the rate constants and, therefore, the relative numbers of the fluorophores found in each state. Fortunately, the memorylessness property makes it easy to adapt the above algorithm to account for a changing irradiance. At each time step of the simulation, a check is performed to see whether the laser irradiance has changed. If it has, new rate constants are computed and a new transition time and state are derived from the algorithm described above.

2.1.3 State system representations

As an example of how state systems are represented in SASS, consider the simplified three-state fluorophore model pictured below.



In this simple model, the fluorophore may be in a fluorescence emitting (ON) state, a non-emitting (OFF) state, and an irreversibly bleached state from which it may never recover. (This model is perhaps too simplistic as it does not account for the typically numerous non-emitting states that real fluorophores possess. It does, however, capture the essential behavior in a SMLM experiment.)

The transition rate from OFF to ON is a constant, k_{ON}, as is the rate k_b from the OFF to the BLEACHED state. The ON to OFF rate k_{OFF} is a function of the irradiance and may be expanded as

$$k_{OFF}(I) = k_{OFF,0} + k_{OFF,1}I + k_{OFF,2}I^2 + \cdots$$

Let's assume that k_{OFF} is at most linear with the irradiance. Then, the full dynamics of the fluorophore may be

specified by a $3 \times 3 \times 2$ matrix M

$$M_{:,:,1} = \begin{bmatrix} k_{OFF,0} \\ 0 \\ k_{ON,0} \\ 0 \\ k_{b,0} \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$M_{:,:,2} = \begin{bmatrix} k_{OFF,1} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

(Note that some browsers may not render the first elements of the above matrices. Both elements are 0.)

The rows of each matrix represent the state being *transitioned from* (ON, OFF, and BLEACHED states respectively), while the columns represent the state that is *transitioned to* (in the same order). For example, the first row of $M_{:,:,1}$ indicates that $k_{OFF,0}$ is the zero-order term for the rate coefficient polynomial expansion in I from the ON state to the OFF state. Here, row number one corresponds to the ON state and column number 2 corresponds to the OFF state. The corresponding element in the second matrix $M_{:,:,2}$ is $k_{OFF,1}$ and indicates that the rate coefficient is linearly proportional to the irradiance. If there were a third matrix $M_{:,:,3}$ with a $k_{OFF,2}$ element, then this would indicate a second-order polynomial term for the dependence of k on I. Zeros for all the remaining elements in $M_{:,:,2}$ indicate that no other rates depend on the irradiance.

Any fluorophore state system may be implemented in SASS by specifying the matrix M.

2.2 Shot noise and sensor noise

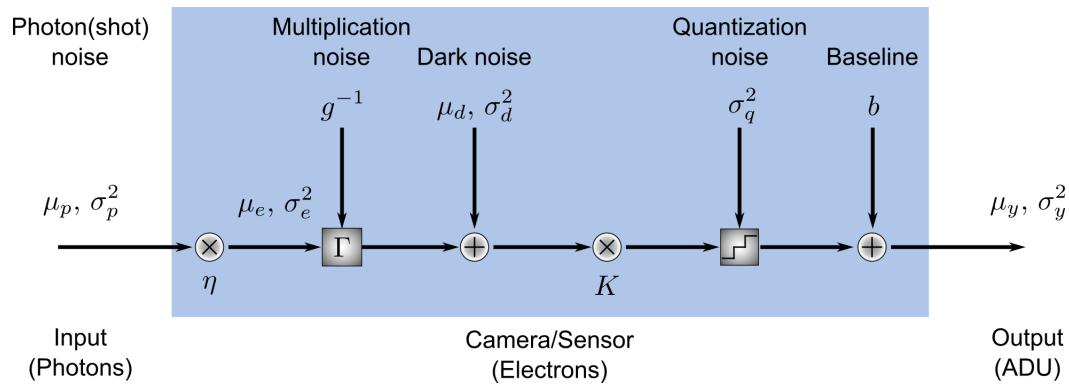
There are two noise models employed by SASS: photon shot noise—which accounts for the quantum nature of fluorescence emission—and sensor noise. Sensor noise is based on the models described in these two documents:

- Basden, Haniff, and Mackay, "Photon counting strategies with low-light-level CCDs," Mon. Not. R. Aston. Soc. 345, 1187-1197 (2003)
- The EMVA 1288 Standard

Sensor noise models in SASS currently do not account for spatial non-uniformities or defect pixels; each pixel is assumed independent from all other pixels. Furthermore, each pixel has identical statistical properties to all other pixels.

Additional assumptions employed in SASS include:

- The sensor is linear.
- Noise sources are wide sense stationary with respect to time and space.
- Only quantum efficiency is wavelength-dependent.
- Only dark current is temperature dependent.



2.2.1 Shot noise

Photon shot noise (or just shot noise) represents fluctuations in the number of photons incident on a pixel between different frame exposures. It is due to the quantum nature of fluorescence emission and is not dependent upon any properties of the image sensor.

Let μ_p represent the mean number of photons incident upon a pixel during the exposure of a given frame. The number of photoelectrons μ_e generated by these photons is given by

$$\mu_e = \eta \mu_p$$

where η is the quantum efficiency of the sensor and, in general, depends on the wavelength of the light.

Fluorescence emission is well-modeled as a Poisson process. Under this condition, the mean number of photoelectrons will be equivalent to the variance $sigma_e^2$ of the number of photoelectrons generated over time.

$$\sigma_e^2 = \mu_e$$

2.2.2 Sensor temporal noise

Within the sensor, photoelectrons are converted to analog-to-digital units (ADU) through a step-wise process involving

1. the amplification of the signal and the addition of multiplication noise (for cameras possessing a multiplication register),
2. the addition of dark noise, which consists of readout noise and dark current noise,
3. the conversion of electrons to voltages by multiplication with a constant system gain factor,
4. and quantization of the voltage to discrete ADU values and summation with a constant baseline value.

The number of photoelectrons that is generated within the pixels of an electron multiplying CCD (EMCCD) is amplified within a serial register via electron avalanche multiplication. This process is random and introduces a multiplicative noise that is modeled as a gamma distribution $\Gamma\left(\mu_e, g^{-1}\right)$ where g^{-1} is the inverse value of the camera's EM gain. (Note that in some notations the second parameter of the gamma distribution is denoted directly by the gain, not its inverse.) Sensors such as sCMOS cameras that lack a serial multiplication register are modeled in SASS by setting the EM gain value to 0.

Following the multiplication register, dark current noise is added to the signal to account for thermally excited electrons within the pixels. Dark current is modeled as a zero-mean Gaussian distribution whose standard deviation is a free parameter. Typically, the value for this parameter is found by assuming that dark current is also a Poisson process whose variance is equivalent to the mean number of dark current electrons $\mu_I t_{exp}$. Here, μ_I is the dark current in electrons per time and t_{exp} is the exposure time of the frame. μ_I is dependent on temperature in general. Dark current is often negligible in microscopy experiments, so it may often be safely ignored.

The total number of amplified photoelectrons and dark current electrons are then readout as a voltage, which introduces a readout noise. Readout noise is modeled as a zero-mean Gaussian distribution whose standard deviation is also a free parameter. The value for this parameter is often given on camera specification sheets as a median or root-mean-square (RMS) number of electrons. (RMS readout noise is preferred for sCMOS cameras because of pixel-to-pixel variation in the values.) Some camera manufacturers will combine dark current and readout noise into a single noise source known as dark noise with mean μ_d and variance σ_d^2.

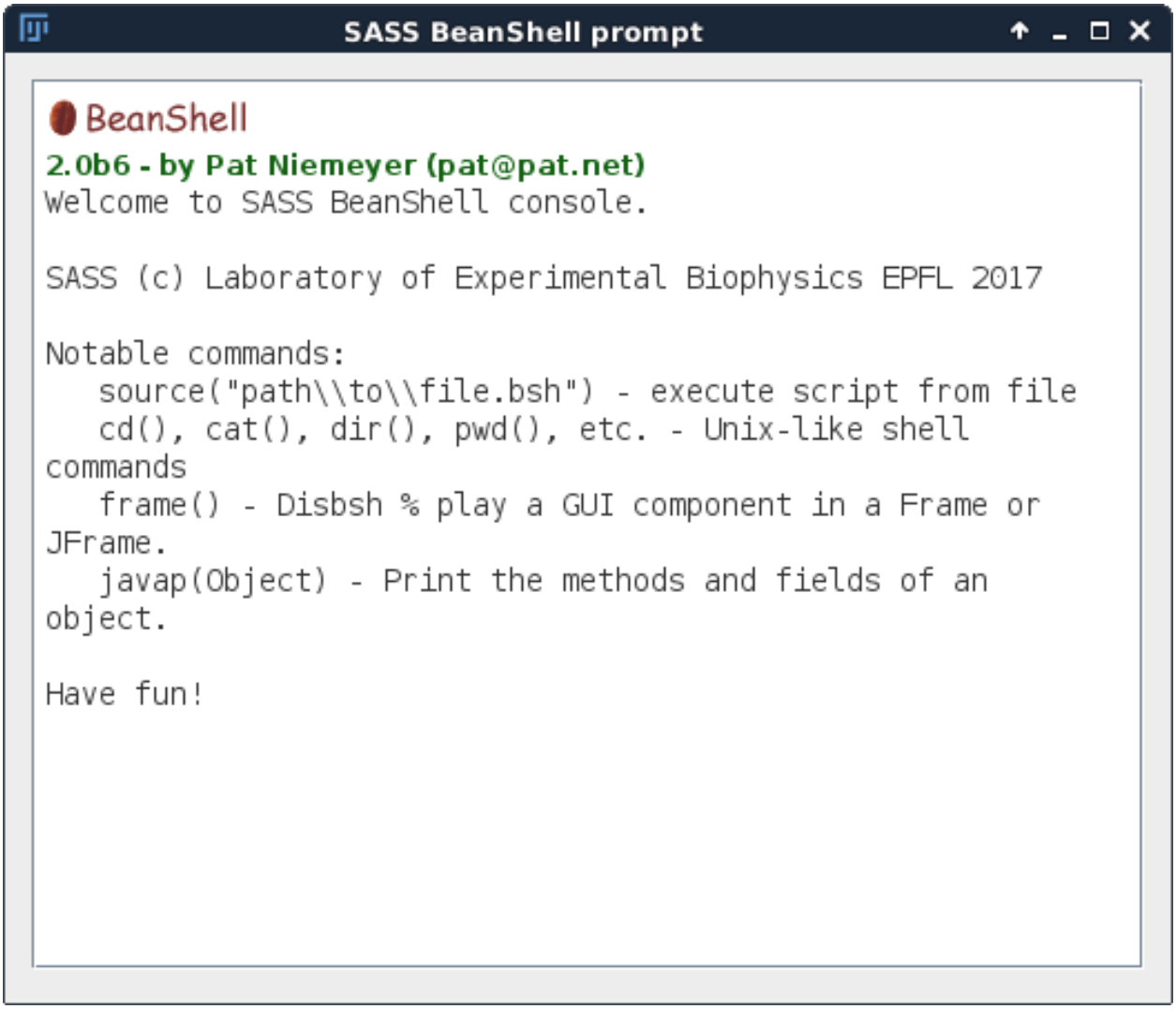
After addition of the readout noise, the voltage signal is amplified by another free parameter found on camera specification sheets, the system gain K. Finally, the signal is quantized into discrete ADUs and optionally summed with a constant baseline b to prevent negative pixel values. This baseline is often about 100 ADU. The quantization noise is a uniform distribution with variance $\sigma_q^2 = \frac{1}{12}\,ADU^2$. It is automatically accounted for in the code by converting from double to integer data types.

CHAPTER 3

Scripting Interface

3.1 The SASS Command Prompt

SASS includes a beanshell scripting interface that supports the execution of either pre-made or *ad hoc* scripts for easily running and repeating simulations. The interface is accessed through the Plugins menu bar via *Plugins > SASS > Command Prompt*.



Inside this prompt you have access to a few Unix-like shell commands by appending `();` to the command name. For example:

```
pwd();
```

prints the current working directory.

3.2 Running Beanshell Scripts

Beanshell scripts that setup and launch localization microscopy simulations may be run with the `source()` command. For security reasons, you will want to ensure that the file you are sourcing is trustworthy because the Beanshell interpreter will run whatever code is contained within the file.

Here is how one would launch the **example_run_generator.bsh** example script from within the command prompt and which launches a basic PALM simulation:

```
source("/path/to/examples/example_run_generator.bsh");
```

Please be sure to change the path argument above to one for your specific machine, which includes changing / to \ if you are using Windows.

3.2.1 From the shell/command line

To better facilitate batch processing and complex workflows, we made it possible to run a Beanshell script directly from the command line by invoking the SASS .jar directly through the Java Virtual Machine:

```
java -jar path/to/SASS/SASS.jar -s path/to/examples/example_run_generator.bsh
```

As you can see, you only need to pass the path to the .jar file on your machine and a **-s** argument followed by the path to the Beanshell script.

3.3 Example Scripts

Example scripts for performing 2D and 3D simulations with PALM and STORM models may be found in the examples folder in the SASS parent directory.

https://github.com/LEB-EPFL/SASS/tree/master/scripts

CHAPTER 4

Remote Procedure Calls and the SASS Server

4.1 Introduction

It is possible to control a SASS simulation from a programming language other than Java or even remotely over a network. This feature is enabled by the SASS remote procedure call (RPC) server. The idea of the RPC server is simple: it listens on a network port for commands sent by other languages and/or computers. When it receives a command, it performs the requested operation and returns any data that is associated with the command.

For example, after initializing a simulation and starting the server, a Python script on the same PC could adjust the laser power on the simulated microscope. It could then ask the server to simulate a number of new images and return them to the Python interpreter for further processing.

As another example, a C++ program could run a simulation by connecting to the server remotely over a network. The details of setting up your network, such as ensuring the correct ports are open in your firewall, are beyond the scope of this documentation.

The RPC service was created using Apache Thrift.

4.2 Starting the server

There are three ways to start the server: via the command line, inside the ImageJ GUI, and via a Beanshell script.

4.2.1 Command line

Enter the following command in a console window to start the server from the command line

```
java -jar PATH_TO_SASS_JAR -r CONFIGURATION_FILE
```

The above command requires two arguments. **PATH_TO_SASS_JAR** is the path and name of the SASS .jar file, which can be downloaded from the releases page of the GitHub repository. **CONFIGURATION_FILE** is a file that specifies the simulation configuration. This file can be created created and saved from inside the SASS ImageJ GUI.

The command will start the server on the default port, which was 9090 at the time of this writing. If instead you wish to specify the port number, use

```
java -jar PATH_TO_SASS_JAR -p PORT -r CONFIGURATION_FILE
```

4.2.2 ImageJ

1. Open the server configuration dialog from the menu bar by clicking **Plugins > SASS > Server**.
2. Enter the port number you wish to use for communications with the server. Usually the default (9090) is fine.
3. Next, you will need a configuration file that defines your simulation parameters. This should be a *.sass* file containing the simulation details. You can create one by navigating to **Plugins > SASS > Simulator**, adjusting the simulation parameters as desired, then clicking the **Save...** button.
4. Once you have a configuration file, click the **Select configuration...** button, navigate to your file, and open it.
5. The **Start** button should now be enabled. Click it and the simulation will initialize. (This may take a few seconds depending on the size of your simulation.)
6. When the server has started, you should see the **Server running** message in the status field.
7. To stop the server, either click the **Stop** button or exit the server control window.

If you are using Fiji, then you can see status updates from the server by navigating to **Window >> Console** on the menu bar.

4.2.3 Beanshell script

There is an example script called **example_server.bsh** in the scripts folder of SASS that demonstrates how to launch the server through a Beanshell script. After creating a Microscope instance named *microscope*, simply create and launch the server with these lines

```
RPCServer server = new RPCServer(microscope, 9090);
server.serve();
```

Note that you will need to first import RPCServer with the command

```
import ch.epfl.leb.sass.server.RPCServer;
```

This code will initialize the server to listen on port 9090 and launch it. If you run the script from the command line, then you can kill the server by typing **Ctrl-C**.

4.3 Server communications

4.3.1 Services

The RPC server works by providing clearly-defined services to clients. Roughly speaking, a service is just a command made by a client that changes the simulation state and/or returns some data. A client must therefore know what services are provided by the server.

The SASS RPC server is implemented using Apache Thrift. The types of services that are provided by the server are defined in the RPCServer.thrift file in the *thrift* folder of the SASS root directory. Here is what RPCServer.thrift file looked like at the time of this writing (comments are omitted)

```
namespace java ch.epfl.leb.sass.server
namespace py remotesim

exception ImageGenerationException { }
exception UnknownSimulationIdException { }

service RemoteSimulationService {


  i32 createSimulation(),

  void deleteSimulation(1: i32 id) throws (1: UnknownSimulationIdException ex),

  double getControlSignal(1: i32 id) throws (1: UnknownSimulationIdException ex),

  string getFluorescenceJsonName(1: i32 id) throws (1: UnknownSimulationIdException␣
↪ex),

  double getFovSize(1: i32 id) throws (1: UnknownSimulationIdException ex),

  i32 getImageCount(1: i32 id) throws (1: UnknownSimulationIdException ex),

  binary getNextImage(1: i32 id) throws(1: ImageGenerationException ex,
                                        2: UnknownSimulationIdException ex2),

  double getObjectSpacePixelSize(1: i32 id) throws (1: UnknownSimulationIdException␣
↪ex),

  string getServerStatus(),

  string getShortTrueSignalDescription(1: i32 id) throws (1:␣
↪UnknownSimulationIdException ex),

  double getTrueSignal(1: i32 id, 2: i32 imageNum) throws (1:␣
↪UnknownSimulationIdException ex),

  void incrementTimeStep(1: i32 id) throws (1: UnknownSimulationIdException ex),

  void setControlSignal(1: i32 id, 2: double power) throws (1:␣
↪UnknownSimulationIdException ex)

  string toJsonMessages(1: i32 id) throws (1: UnknownSimulationIdException ex),

  string toJsonState(1: i32 id) throws (1: UnknownSimulationIdException ex),

}
```

This file first defines the package names for Java and Python code, respectively, and a few exceptions that the server will return when something goes wrong. After that, it then defines the service that the server provides. There are a number of method calls such as:

1. **setControlSignal()** - Adjusts the simulation's laser power.
2. **getNextImage()** - Simulates a new image.
3. **toJsonMessages()** - Dump the simulation message cache.
4. **toJsonState()** - Get information on the current state of the simulation's components.

To turn this script into code, it must be compiled by the Thrift compiler. An example of how to do this for Java is

located in the compile.sh file inside the thrift folder. Compilation produces files that enable the server in your target language.

Note that the SASS RPC server sends images as tif-encoded byte strings and the simulation state as JSON strings. You will need to decode this information after its received in your target language.

4.3.2 A Python client

The general problem of setting up a client to interact with the simulation is not so much a SASS problem but is rather more within the scope of working with Apache Thrift. There are many excellent tutorials on their website on how to do this in a number of different languages.

To get you started, we provide here a basic workflow to setup a rudimentary Python client to control a SASS simulation.

1. Get Apache Thrift.
2. Navigate into the folder containing the RPCServer.thrift file and open it. Add the namespace for your target lanuage. For Python, this has already been done for you.
3. Compile the thrift file into Python with the command *thrift -r –gen py RPCServer.thrift*.
4. Install the Thrift bindings for Python, preferably inside a virtual environment. *pip install thrift*
4. Enter the folder **gen-py** (or move it to a convenient directory).
5. Create an emtpy file named client.py.

Inside the client.py file, you will need to add the following code

```
from thrift import Thrift
from thrift.transport import TSocket
from thrift.transport import TTransport
from thrift.protocol import TBinaryProtocol
from remotesim import RemoteSimulationService
from PIL import Image
from io import BytesIO

def main():
    # Make socket
    transport = TSocket.TSocket('localhost', 9090)

    # Buffering is critical. Raw sockets are very slow
    transport = TTransport.TBufferedTransport(transport)

    # Wrap in a protocol
    protocol = TBinaryProtocol.TBinaryProtocol(transport)

    # Create a client to use the protocol encoder
    client = RemoteSimulationService.Client(protocol)

    # Connect!
    transport.open()

    try:
         x = client.getNextImage()
         img = Image.open(BytesIO(x))
         img.load()
         img.show()
    finally:
```

(continues on next page)

(continued from previous page)

```
        transport.close()

if __name__ == '__main__':
    main()
```

This will create the client and request the next image from the simulation. **By default, the RPC Server will return images as tif-encoded byte strings.** You therefore will need the libtiff library in your target language to decode them. In Python, this can be provided by pillow.

4.3.3 A Java client

A simple Java client for the SASS RPC server **'is already included in SASS'_**.

CHAPTER 5

Frequently Asked Questions

Contents

5.1 General

5.1.1 What are the advantages of SASS over other SMLM simulators?

- Complete integration with ImageJ/Fiji.
- Incorporates automated control systems into the simulation environment.
- Allows for dynamic adjustment of the illumination *during* a simulation.
- Easy-to-use script interface via Beanshell and the example scripts.
- Interfaces are available for extending simulation attributes, such as PSF generation, background, fiducial markers, and custom fluorophore photophysics.

5.1.2 What does SASS stand for?

SMLM Acquisition Simulation Software.

(SMLM stands for Single Molecule Localization Microscopy.)

5.2 Running Simulations

5.2.1 How is the coordinate system in SASS defined?

Coordinates in SASS are typically in units of pixels unless otherwise noted in the documentation. **Please note that the origin of the Cartesian coordinate system lies at the center of a pixel, not at a corner.**

5.2.2 How are custom fluorophore position lists formatted?

Customized fluorophore positions are imported into SASS from an externally-generated file that you create yourself. This file should contain two columns (*optionally three*) of comma-separated numerical values (for example, a .csv file). Each row represents the position of one fluorophore; the first column represents the fluorophore's x-position, while the second column represents the fluorophore's y-position. If you want to do 3D simulations, there should be a third column for the z-position. The file should contain no header or comments.

Example

The column labels **should not** be included in the file; they are illustrated here only to indicate which columns correspond to x and y.

x, pixels	y, pixels	(optional) z, arb. units
1.2376	4.2340	0.0000
2.7300	3.7105	0.0000
2.4360	1.2887	0.0000
...	...	...

The units of the values are in pixels, and, **for imports from CSV files only**, the origin is in the upper left-hand corner of the generated image stacks, not the center of the upper left pixel. After import into SASS, there is an implicit subtraction of half a pixel from the x- and y-coordinate values which shifts the coordinate system into the one used by SASS. This is done to preserve the same relative pixel locations when importing from the same file into SASS or ThunderSTORM.

For example, a fluorophore with a position in the CSV file at (15.5, 15.5) will lie at the center of a pixel in ThunderSTORM. To get it to lie at the center of a pixel in SASS, 0.5 is subtracted from each coordinate to make the resulting position (15, 15). Because the origin is at a pixel center in SASS, so to will be this fluorophore's position.

Check out ThunderSTORM for more information.

5.2.3 What are the units for the axial (z) direction?

The units of the values in the z-column of the fluorophore position lists can be any unit that you want, so long as you are consistent in your choice of units for the properties of the various simulation components.

For example, if the you specify the fluorophore z-positions in microns, then you should use microns for the fluorescence wavelength, stage displacement, and other values that require a length.

5.2.4 How is the stage z-displacement property used?

The z-displacement of the stage is used for some 3D point spread functions that depend on the emitter's distance from the coverslip.

- z = 0 corresponds to the coverslip surface.
- Negative z-positions correspond to moving the stage downwards on an inverted microscope. For example, a stage z-position of -2 microns corresponds to a focal volume that is located +2 microns above the coverslip surface.

CHAPTER 6

Javadoc

6.1 ch.epfl.leb.sass

6.1.1 IntegrationTest

public interface **`IntegrationTest`**

Category annotation for IntegrationTests.

Author Kyle M. Douglass

6.2 ch.epfl.leb.sass.client

6.2.1 RPCClient

public class **`RPCClient`**

A client for interaction with the SASS RPCServer.

Author Kyle M. Douglass

Constructors

RPCClient

public **`RPCClient`** (String *hostUrl*, int *port*)

Creates a RPCClient instance for communications with the RPCServer. For RPCServers running on the same machine, use "localhost" for the hostUrl argument.

Parameters

- **`hostUrl`** – The URL of the RPCServer.

- **port** – The port that the RPCServer is listening on.

Methods

close

public void **close**()

Closes the transport layer to the server. This method must be called before the program terminates.

finalize

protected void **finalize**()

Safety check that the transport layer is properly closed.

getClient

public *RemoteSimulationService.Client* **getClient**()

Returns a copy of service client. Use the client to make calls to the RPC server.

Returns A copy of the RemoteSimulationService client.

6.3 ch.epfl.leb.sass.commandline

6.3.1 BeanShellConsole

public class **BeanShellConsole** extends PlugInFrame

BeanShell console for execution of SASS simulation scripts

Author Marcel Stefko

Constructors

BeanShellConsole

public **BeanShellConsole**(String *title*)

Initialize the new frame

Parameters

- **title** – name of the frame

Methods

getInterpreter

public Interpreter **getInterpreter**()

Returns BeanShell interpreter associated with this BeanShellConsole

6.3.2 CommandLineInterface

public final class **CommandLineInterface**
Main class of the project, launches the BeanShell script interface.

Author Marcel Stefko

Methods

constructOptions

public static Options **constructOptions** ( )

Returns all understood options for ALICA execution

main

public static void **main** (String[] *args*)
Shows help, launches the interpreter and executes scripts according to input args.

Parameters

- **args** – input arguments

printWelcomeText

public static void **printWelcomeText** (PrintStream *out*)
Reads the welcome_text file and prints it to a PrintStream.

Parameters

- **out** – stream to print to

6.4 ch.epfl.leb.sass.ijplugin

6.4.1 App

public class **App** extends *ImageJSimulator*
Backend for the FIJI plugin GUI

Author Marcel Stefko

Constructors

App

public **App** (*Microscope microscope*, Analyzer *analyzer*, Controller *controller*, int *controller_tickrate*)
Assemble the App from custom components.

Parameters

- **microscope** – The microscope to be simulated.

- **analyzer** – An analyzer for processing images from the microscope.
- **controller** – A controller that adjusts the state of the microscope.

Methods

getAnalyzerOutput

public ArrayList<Double> **getAnalyzerOutput** ()

getControllerOutput

public ArrayList<Double> **getControllerOutput** ()

getControllerSetpoint

public ArrayList<Double> **getControllerSetpoint** ()

getControllerTickrate

public int **getControllerTickrate** ()

getGeneratorTrueSignal

public ArrayList<Double> **getGeneratorTrueSignal** ()

getStatusFrame

public *SimulatorStatusFrame* **getStatusFrame** ()
Return the handle for the status frame.

Returns Plots with the simulation history.

setSetpoint

public void **setSetpoint** (double *value*)
Set new setpoint for the controller

Parameters

- **value** – new setpoint value

startSimulating

public void **startSimulating** ()
Start continuously generating new images until stopped.

stopSimulating

public void **stopSimulating**()
Stop generating new images.

6.4.2 ButtonGroupUtils

public class **ButtonGroupUtils**
Utilities for working with button groups. See https://stackoverflow.com/questions/201287/how-do-i-get-which-jradiobutton-is-selected-from-a-buttongroup#13232816

Author Kyle M. Douglass

Methods

getSelectedButtonText

public static String **getSelectedButtonText**(ButtonGroup *buttonGroup*)
Determines the label of the current selected button.

Parameters

- **buttonGroup** –

Returns The text label of the selected button.

selectButtonModelFromText

public static void **selectButtonModelFromText**(ButtonGroup *buttonGroup*, String *text*)
Selects the button in a button group based on its text label.

Parameters

- **buttonGroup** –
- **text** – The text label of the desired button to select.

6.4.3 CommandPrompt

public class **CommandPrompt** implements PlugIn
Wrapper for initialization of BeanShell console

Author Marcel Stefko

Constructors

CommandPrompt

public **CommandPrompt**()
Initializes new BeanShell console

Methods

run

public void **run** (String *string*)
Set input and output streams, and print welcome text.

Parameters

- **string** –

6.4.4 GUI

public class **GUI** extends PlugInFrame
Main FIJI plugin frame.

Author Marcel Stefko

Fields

app

App **app**

Constructors

GUI

public **GUI** (String *title*)
Creates new form MainFrame

Parameters

- **title** – title of the window

GUI

public **GUI** ()
Initialize the new frame

Methods

run

public void **run** (String *arg*)
Show the frame and initialize backend.

Parameters

- **arg** –

setApp

public void **setApp** (*App app*)
Set the App which this GUI should control

Parameters

- **app** –

6.4.5 IJPluginModel

public class **IJPluginModel** implements Serializable
IJPluginModel for the InitializeSimulation window.

Author Kyle M. Douglass

Methods

build

public *Microscope* **build** ()
Builds a microscope from the model parameters.

Returns A new microscope built from the model parameters.

getAnalyzerCurrentSelection

public String **getAnalyzerCurrentSelection** ()

getBackgroundCurrentSelection

public String **getBackgroundCurrentSelection** ()

getBackgroundRandomButtonText

public String **getBackgroundRandomButtonText** ()

getBackgroundRandomFeatureSize

public double **getBackgroundRandomFeatureSize** ()

getBackgroundRandomMaxValue

public float **getBackgroundRandomMaxValue** ()

getBackgroundRandomMinValue

public float **getBackgroundRandomMinValue** ()

getBackgroundRandomSeed

public int **getBackgroundRandomSeed**()

getBackgroundTifFile

public String **getBackgroundTifFile**()

getBackgroundTifFileButtonText

public String **getBackgroundTifFileButtonText**()

getBackgroundUniformButtonText

public String **getBackgroundUniformButtonText**()

getBackgroundUniformSignal

public float **getBackgroundUniformSignal**()

getCameraAduPerElectron

public double **getCameraAduPerElectron**()

getCameraBaseline

public int **getCameraBaseline**()

getCameraDarkCurrent

public double **getCameraDarkCurrent**()

getCameraEmGain

public int **getCameraEmGain**()

getCameraNX

public int **getCameraNX**()

getCameraNY

public int **getCameraNY**()

getCameraPixelSize

public double **getCameraPixelSize**()

getCameraQuantumEfficiency

public double **getCameraQuantumEfficiency**()

getCameraReadoutNoise

public double **getCameraReadoutNoise**()

getCameraThermalNoise

public double **getCameraThermalNoise**()

getControllerCurrentSelection

public String **getControllerCurrentSelection**()

getEmitters3DCheckBoxEnabled

public boolean **getEmitters3DCheckBoxEnabled**()

getEmitters3DMaxZ

public double **getEmitters3DMaxZ**()

getEmitters3DMinZ

public double **getEmitters3DMinZ**()

getEmittersCsvFile

public String **getEmittersCsvFile**()

getEmittersCsvFileButtonText

public String **getEmittersCsvFileButtonText**()

getEmittersCurrentSelection

public String **getEmittersCurrentSelection**()

getEmittersGridButtonText

public String **getEmittersGridButtonText**()

getEmittersGridSpacing

public int **getEmittersGridSpacing**()

getEmittersRandomButtonText

public String **getEmittersRandomButtonText**()

getEmittersRandomNumber

public int **getEmittersRandomNumber**()

getFiducialsNumber

public int **getFiducialsNumber**()

getFiducialsSignal

public double **getFiducialsSignal**()

getFluorophoreCurrentSelection

public String **getFluorophoreCurrentSelection**()

getFluorophorePalmText

public String **getFluorophorePalmText**()

getFluorophoreSignal

public double **getFluorophoreSignal**()

getFluorophoreSimpleText

public String **getFluorophoreSimpleText**()

getFluorophoreStormText

public String **getFluorophoreStormText**()

getFluorophoreTBl

public double **getFluorophoreTBl**()

getFluorophoreTOff

public double **getFluorophoreTOff**()

getFluorophoreTOn

public double **getFluorophoreTOn**()

getFluorophoreWavelength

public double **getFluorophoreWavelength**()

getLaserCurrentPower

public double **getLaserCurrentPower**()

getLaserMaxPower

public double **getLaserMaxPower**()

getLaserMinPower

public double **getLaserMinPower**()

getObjectiveMag

public double **getObjectiveMag**()

getObjectiveNa

public double **getObjectiveNa**()

getPalmKA

public double **getPalmKA**()

getPalmKB

public double **getPalmKB**()

getPalmKD1

public double **getPalmKD1** ( )

getPalmKD2

public double **getPalmKD2** ( )

getPalmKR1

public double **getPalmKR1** ( )

getPalmKR2

public double **getPalmKR2** ( )

getPalmSignal

public double **getPalmSignal** ( )

getPalmWavelength

public double **getPalmWavelength** ( )

getPsfCurrentSelection

public String **getPsfCurrentSelection** ( )

getPsfGaussian2dText

public String **getPsfGaussian2dText** ( )

getPsfGaussian3dText

public String **getPsfGaussian3dText** ( )

getPsfGibsonLanniMaxRadius

public int **getPsfGibsonLanniMaxRadius** ( )

getPsfGibsonLanniNg

public double **getPsfGibsonLanniNg** ( )

getPsfGibsonLanniNg0

public double **getPsfGibsonLanniNg0**()

getPsfGibsonLanniNi

public double **getPsfGibsonLanniNi**()

getPsfGibsonLanniNi0

public double **getPsfGibsonLanniNi0**()

getPsfGibsonLanniNs

public double **getPsfGibsonLanniNs**()

getPsfGibsonLanniNumBasis

public int **getPsfGibsonLanniNumBasis**()

getPsfGibsonLanniNumSamples

public int **getPsfGibsonLanniNumSamples**()

getPsfGibsonLanniOversampling

public int **getPsfGibsonLanniOversampling**()

getPsfGibsonLanniResPsf

public double **getPsfGibsonLanniResPsf**()

getPsfGibsonLanniResPsfAxial

public double **getPsfGibsonLanniResPsfAxial**()

getPsfGibsonLanniSizeX

public int **getPsfGibsonLanniSizeX**()

getPsfGibsonLanniSizeY

public int **getPsfGibsonLanniSizeY**()

getPsfGibsonLanniSolver

public String **getPsfGibsonLanniSolver**()

getPsfGibsonLanniText

public String **getPsfGibsonLanniText**()

getPsfGibsonLanniTg

public double **getPsfGibsonLanniTg**()

getPsfGibsonLanniTg0

public double **getPsfGibsonLanniTg0**()

getPsfGibsonLanniTi0

public double **getPsfGibsonLanniTi0**()

getStageX

public double **getStageX**()

getStageY

public double **getStageY**()

getStageZ

public double **getStageZ**()

getStormKBl

public double **getStormKBl**()

getStormKDark

public double **getStormKDark**()

getStormKDarkRecovery

public double **getStormKDarkRecovery**()

getStormKDarkRecoveryConstant

public double **getStormKDarkRecoveryConstant** ( )

getStormKTriplet

public double **getStormKTriplet** ( )

getStormKTripletRecovery

public double **getStormKTripletRecovery** ( )

getStormSignal

public double **getStormSignal** ( )

getStormWavelength

public double **getStormWavelength** ( )

read

public static *IJPluginModel* **read** (FileInputStream *fileIn*)

Loads a model from a file.

Parameters

- **fileIn** – The input stream from the file.

setAnalyzerCurrentSelection

public void **setAnalyzerCurrentSelection** (String *text*)

setBackgroundCurrentSelection

public void **setBackgroundCurrentSelection** (String *currentSelection*)

setBackgroundRandomButtonText

public void **setBackgroundRandomButtonText** (String *text*)

setBackgroundRandomFeatureSize

public void **setBackgroundRandomFeatureSize** (double *featureSize*)

setBackgroundRandomMaxValue

public void **setBackgroundRandomMaxValue** (float *maxValue*)

setBackgroundRandomMinValue

public void **setBackgroundRandomMinValue** (float *minValue*)

setBackgroundRandomSeed

public void **setBackgroundRandomSeed** (int *seed*)

setBackgroundTifFile

public void **setBackgroundTifFile** (String *filename*)

setBackgroundTifFileButtonText

public void **setBackgroundTifFileButtonText** (String *text*)

setBackgroundUniformButtonText

public void **setBackgroundUniformButtonText** (String *text*)

setBackgroundUniformSignal

public void **setBackgroundUniformSignal** (float *signal*)

setCameraAduPerElectron

public void **setCameraAduPerElectron** (double *aduPerElectron*)

setCameraBaseline

public void **setCameraBaseline** (int *baseline*)

setCameraDarkCurrent

public void **setCameraDarkCurrent** (double *darkCurrent*)

setCameraEmGain

public void **setCameraEmGain** (int *emGain*)

setCameraNX

public void **setCameraNX** (int *nX*)

setCameraNY

public void **setCameraNY** (int *nY*)

setCameraPixelSize

public void **setCameraPixelSize** (double *pixelSize*)

setCameraQuantumEfficiency

public void **setCameraQuantumEfficiency** (double *quantumEfficiency*)

setCameraReadoutNoise

public void **setCameraReadoutNoise** (double *readoutNoise*)

setCameraThermalNoise

public void **setCameraThermalNoise** (double *thermalNoise*)

setControllerCurrentSelection

public void **setControllerCurrentSelection** (String *text*)

setEmitters3DCheckBoxEnabled

public void **setEmitters3DCheckBoxEnabled** (boolean *enabled*)

setEmitters3DMaxZ

public void **setEmitters3DMaxZ** (double *max*)

setEmitters3DMinZ

public void **setEmitters3DMinZ** (double *min*)

setEmittersCsvFile

public void **setEmittersCsvFile** (String *filename*)

setEmittersCsvFileButtonText

public void **setEmittersCsvFileButtonText** (String *text*)

setEmittersCurrentSelection

public void **setEmittersCurrentSelection** (String *currentSelection*)

setEmittersGridButtonText

public void **setEmittersGridButtonText** (String *text*)

setEmittersGridSpacing

public void **setEmittersGridSpacing** (int *spacing*)

setEmittersRandomButtonText

public void **setEmittersRandomButtonText** (String *text*)

setEmittersRandomNumber

public void **setEmittersRandomNumber** (int *number*)

setFiducialsNumber

public void **setFiducialsNumber** (int *number*)

setFiducialsSignal

public void **setFiducialsSignal** (double *signal*)

setFluorophoreCurrentSelection

public void **setFluorophoreCurrentSelection** (String *text*)

setFluorophorePalmText

public void **setFluorophorePalmText** (String *text*)

setFluorophoreSignal

public void **setFluorophoreSignal** (double *signal*)

setFluorophoreSimpleText

public void **setFluorophoreSimpleText** (String *text*)

setFluorophoreStormText

public void **setFluorophoreStormText** (String *text*)

setFluorophoreTBl

public void **setFluorophoreTBl** (double *tBl*)

setFluorophoreTOff

public void **setFluorophoreTOff** (double *tOff*)

setFluorophoreTOn

public void **setFluorophoreTOn** (double *tOn*)

setFluorophoreWavelength

public void **setFluorophoreWavelength** (double *wavelength*)

setLaserCurrentPower

public void **setLaserCurrentPower** (double *currentPower*)

setLaserMaxPower

public void **setLaserMaxPower** (double *maxPower*)

setLaserMinPower

public void **setLaserMinPower** (double *minPower*)

setObjectiveMag

public void **setObjectiveMag** (double *mag*)

setObjectiveNa

public void **setObjectiveNa** (double *na*)

setPalmKA

public void **setPalmKA** (double *kA*)

setPalmKB

public void **setPalmKB** (double *kB*)

setPalmKD1

public void **setPalmKD1** (double *kD1*)

setPalmKD2

public void **setPalmKD2** (double *kD2*)

setPalmKR1

public void **setPalmKR1** (double *kR1*)

setPalmKR2

public void **setPalmKR2** (double *kR2*)

setPalmSignal

public void **setPalmSignal** (double *signal*)

setPalmWavelength

public void **setPalmWavelength** (double *wavelength*)

setPsfCurrentSelection

public void **setPsfCurrentSelection** (String *text*)

setPsfGaussian2dText

public void **setPsfGaussian2dText** (String *text*)

setPsfGaussian3dText

public void **setPsfGaussian3dText** (String *text*)

setPsfGibsonLanniMaxRadius

public void **setPsfGibsonLanniMaxRadius** (int *maxRadius*)

setPsfGibsonLanniNg

public void **setPsfGibsonLanniNg** (double *ng*)

setPsfGibsonLanniNg0

public void **setPsfGibsonLanniNg0** (double *ng0*)

setPsfGibsonLanniNi

public void **setPsfGibsonLanniNi** (double *ni*)

setPsfGibsonLanniNi0

public void **setPsfGibsonLanniNi0** (double *ni0*)

setPsfGibsonLanniNs

public void **setPsfGibsonLanniNs** (double *ns*)

setPsfGibsonLanniNumBasis

public void **setPsfGibsonLanniNumBasis** (int *numBasis*)

setPsfGibsonLanniNumSamples

public void **setPsfGibsonLanniNumSamples** (int *numSamples*)

setPsfGibsonLanniOversampling

public void **setPsfGibsonLanniOversampling** (int *oversampling*)

setPsfGibsonLanniResPsf

public void **setPsfGibsonLanniResPsf** (double *resPsf*)

setPsfGibsonLanniResPsfAxial

public void **setPsfGibsonLanniResPsfAxial** (double *resPsfAxial*)

setPsfGibsonLanniSizeX

public void **setPsfGibsonLanniSizeX** (int *sizeX*)

setPsfGibsonLanniSizeY

public void **setPsfGibsonLanniSizeY** (int *sizeY*)

setPsfGibsonLanniSolver

public void **setPsfGibsonLanniSolver** (String *solver*)

setPsfGibsonLanniText

public void **setPsfGibsonLanniText** (String *text*)

setPsfGibsonLanniTg

public void **setPsfGibsonLanniTg** (double *tg*)

setPsfGibsonLanniTg0

public void **setPsfGibsonLanniTg0** (double *tg0*)

setPsfGibsonLanniTi0

public void **setPsfGibsonLanniTi0** (double *ti0*)

setStageX

public void **setStageX** (double *x*)

setStageY

public void **setStageY** (double *y*)

setStageZ

public void **setStageZ** (double *z*)

setStormKBl

public void **setStormKBl** (double *kBl*)

setStormKDark

public void **setStormKDark** (double *kDark*)

setStormKDarkRecovery

public void **setStormKDarkRecovery** (double *kDarkRecovery*)

setStormKDarkRecoveryConstant

public void **setStormKDarkRecoveryConstant** (double *kDarkRecoveryConstant*)

setStormKTriplet

public void **setStormKTriplet** (double *kTriplet*)

setStormKTripletRecovery

public void **setStormKTripletRecovery** (double *kTripletRecovery*)

setStormSignal

public void **setStormSignal** (double *signal*)

setStormWavelength

public void **setStormWavelength** (double *wavelength*)

write

public void **write** (FileOutputStream *fileOut*)

Saves the model's state to a file.

Parameters

- **fileOut** – The output stream to the file.

6.4.6 InitializeSimulation

public class **InitializeSimulation** extends java.awt.Dialog

Frame for basic setup of a simulation.

Author Marcel Stefko

Fields

backgroundTifFile

File **backgroundTifFile**

emittersCsvFile

File **emittersCsvFile**

main

GUI **main**

model

IJPluginModel **model**

Constructors

InitializeSimulation

public **InitializeSimulation** (java.awt.Frame *parent*, boolean *modal*, *GUI main*)

Assemble the frame and display it

Parameters

- **parent** –
- **modal** – should the window be persistent
- **main** – GUI to notify

6.4.7 InteractionWindow

public class **InteractionWindow** extends javax.swing.JFrame

Author stefko

Constructors

InteractionWindow

public **InteractionWindow** (Analyzer *analyzer*, Controller *controller*)

Creates new form InteractionWindow

6.4.8 ModelTest

public class **ModelTest**

Author Kyle M. Douglass

Constructors

ModelTest

public **ModelTest** ()

Methods

testGetAnalyzerCurrentSelection

public void **testGetAnalyzerCurrentSelection** ()
Test of getAnalyzerCurrentSelection method, of class IJPluginModel.

testGetBackgroundCurrentSelection

public void **testGetBackgroundCurrentSelection** ()
Test of getBackgroundCurrentSelection method, of class IJPluginModel.

testGetBackgroundRandomButtonText

public void **testGetBackgroundRandomButtonText** ()
Test of getBackgroundRandomButtonText() {

testGetBackgroundRandomFeatureSize

public void **testGetBackgroundRandomFeatureSize** ()
Test of getBackgroundRandomFeatureSize method, of class IJPluginModel.

testGetBackgroundRandomMaxValue

public void **testGetBackgroundRandomMaxValue** ()
Test of getBackgroundRandomMaxValue method, of class IJPluginModel.

testGetBackgroundRandomMinValue

public void **testGetBackgroundRandomMinValue** ()
Test of getBackgroundRandomMinValue method, of class IJPluginModel.

testGetBackgroundRandomSeed

public void **testGetBackgroundRandomSeed**()
Test of getBackgroundRandomFeatureSize method, of class IJPluginModel.

testGetBackgroundTifFile

public void **testGetBackgroundTifFile**()
Test of getBackgroundTifFile method, of class IJPluginModel.

testGetBackgroundTifFileButtonText

public void **testGetBackgroundTifFileButtonText**()
Test of getBackgroundTifFileButtonText method, of class IJPluginModel.

testGetBackgroundUniformButtonText

public void **testGetBackgroundUniformButtonText**()
Test of getBackgroundUniformButtonText method, of class IJPluginModel.

testGetBackgroundUniformSignal

public void **testGetBackgroundUniformSignal**()
Test of getBackgroundUniformSignal method, of class IJPluginModel.

testGetCameraAduPerElectron

public void **testGetCameraAduPerElectron**()
Test of getCameraAduPerElectron method, of class IJPluginModel.

testGetCameraBaseline

public void **testGetCameraBaseline**()
Test of getCameraBaseline method, of class IJPluginModel.

testGetCameraDarkCurrent

public void **testGetCameraDarkCurrent**()
Test of getCameraDarkCurrent method, of class IJPluginModel.

testGetCameraEmGain

public void **testGetCameraEmGain**()
Test of getCameraEmGain method, of class IJPluginModel.

testGetCameraNX

public void **testGetCameraNX**()
Test of getCameraNX method, of class IJPluginModel.

testGetCameraNY

public void **testGetCameraNY**()
Test of getCameraNY method, of class IJPluginModel.

testGetCameraPixelSize

public void **testGetCameraPixelSize**()
Test of getCameraPixelSize method, of class IJPluginModel.

testGetCameraQuantumEfficiency

public void **testGetCameraQuantumEfficiency**()
Test of getCameraQuantumEfficiency method, of class IJPluginModel.

testGetCameraReadoutNoise

public void **testGetCameraReadoutNoise**()
Test of getCameraReadoutNoise method, of class IJPluginModel.

testGetCameraThermalNoise

public void **testGetCameraThermalNoise**()
Test of getCameraThermalNoise method, of class IJPluginModel.

testGetControllerCurrentSelection

public void **testGetControllerCurrentSelection**()
Test of getControllerCurrentSelection method, of class IJPluginModel.

testGetEmitters3DCheckBoxEnabled

public void **testGetEmitters3DCheckBoxEnabled**()
Test of getEmitters3DCheckBoxEnabled method, of class IJPluginModel.

testGetEmitters3DMaxZ

public void **testGetEmitters3DMaxZ**()
Test of getEmitters3DMaxZ method, of class IJPluginModel.

testGetEmitters3DMinZ

public void **testGetEmitters3DMinZ**()
Test of getEmitters3DMinZ method, of class IJPluginModel.

testGetEmittersCsvFile

public void **testGetEmittersCsvFile**()
Test of getEmittersCsvFile method, of class IJPluginModel.

testGetEmittersCsvFileButtonText

public void **testGetEmittersCsvFileButtonText**()
Test of getEmittersCsvFileButtonText method, of class IJPluginModel.

testGetEmittersCurrentSelection

public void **testGetEmittersCurrentSelection**()
Test of getEmittersCurrentSelection method, of class IJPluginModel.

testGetEmittersGridButtonText

public void **testGetEmittersGridButtonText**()
Test of getEmittersGridButtonText method, of class IJPluginModel.

testGetEmittersGridSpacing

public void **testGetEmittersGridSpacing**()
Test of getEmittersGridSpacing method, of class IJPluginModel.

testGetEmittersRandomButtonText

public void **testGetEmittersRandomButtonText**()
Test of getEmittersRandomButtonText method, of class IJPluginModel.

testGetEmittersRandomNumber

public void **testGetEmittersRandomNumber**()
Test of getEmittersRandomNumber method, of class IJPluginModel.

testGetFiducialsNumber

public void **testGetFiducialsNumber**()
Test of getFiducialsNumber method, of class IJPluginModel.

testGetFiducialsSignal

public void **testGetFiducialsSignal**()
Test of getFiducialsSignal method, of class IJPluginModel.

testGetFluorophoreCurrentSelection

public void **testGetFluorophoreCurrentSelection**()
Test of getFluorophoreCurrentSelection method, of class IJPluginModel.

testGetFluorophorePalmText

public void **testGetFluorophorePalmText**()
Test of getFluorophorePalmText method, of class IJPluginModel.

testGetFluorophoreSignal

public void **testGetFluorophoreSignal**()
Test of getFluorophoreSignal method, of class IJPluginModel.

testGetFluorophoreSimpleText

public void **testGetFluorophoreSimpleText**()
Test of getFluorophoreSimpleText method, of class IJPluginModel.

testGetFluorophoreStormText

public void **testGetFluorophoreStormText**()
Test of getFluorophoreStormText method, of class IJPluginModel.

testGetFluorophoreTBl

public void **testGetFluorophoreTBl**()
Test of getFluorophoreTBl method, of class IJPluginModel.

testGetFluorophoreTOff

public void **testGetFluorophoreTOff**()
Test of getFluorophoreTOff method, of class IJPluginModel.

testGetFluorophoreTOn

public void **testGetFluorophoreTOn**()
Test of getFluorophoreTOn method, of class IJPluginModel.

testGetFluorophoreWavelength

public void **testGetFluorophoreWavelength**()
Test of getFluorophoreWavelength method, of class IJPluginModel.

testGetLaserCurrentPower

public void **testGetLaserCurrentPower**()
Test of getLaserCurrentPower method, of class IJPluginModel.

testGetLaserMaxPower

public void **testGetLaserMaxPower**()
Test of getLaserMaxPower method, of class IJPluginModel.

testGetLaserMinPower

public void **testGetLaserMinPower**()
Test of getLaserMinPower method, of class IJPluginModel.

testGetObjectiveMag

public void **testGetObjectiveMag**()
Test of getObjectiveMag method, of class IJPluginModel.

testGetObjectiveNa

public void **testGetObjectiveNa**()
Test of getObjectiveNa method, of class IJPluginModel.

testGetPalmKA

public void **testGetPalmKA**()
Test of getPalmKA method, of class IJPluginModel.

testGetPalmKB

public void **testGetPalmKB**()
Test of getPalmKB method, of class IJPluginModel.

testGetPalmKD1

public void **testGetPalmKD1**()
Test of getPalmKD1 method, of class IJPluginModel.

testGetPalmKD2

public void **testGetPalmKD2** ()
Test of getPalmKD2 method, of class IJPluginModel.

testGetPalmKR1

public void **testGetPalmKR1** ()
Test of getPalmKR1 method, of class IJPluginModel.

testGetPalmKR2

public void **testGetPalmKR2** ()
Test of getPalmKR2 method, of class IJPluginModel.

testGetPalmSignal

public void **testGetPalmSignal** ()
Test of getPalmSignal method, of class IJPluginModel.

testGetPalmWavelength

public void **testGetPalmWavelength** ()
Test of getPalmWavelength method, of class IJPluginModel.

testGetPsfCurrentSelection

public void **testGetPsfCurrentSelection** ()
Test of getPsfCurrentSelection method, of class IJPluginModel.

testGetPsfGaussian2dText

public void **testGetPsfGaussian2dText** ()
Test of getPsfGaussian2dText method, of class IJPluginModel.

testGetPsfGaussian3dText

public void **testGetPsfGaussian3dText** ()
Test of getPsfGaussian3dText method, of class IJPluginModel.

testGetPsfGibsonLanniMaxRadius

public void **testGetPsfGibsonLanniMaxRadius** ()
Test of getPsfGibsonLanniMaxRadius, of class IJPluginModel.

testGetPsfGibsonLanniNg

public void **testGetPsfGibsonLanniNg**()
Test of getPsfGibsonLanniNg, of class IJPluginModel.

testGetPsfGibsonLanniNg0

public void **testGetPsfGibsonLanniNg0**()
Test of getPsfGibsonLanniNg0, of class IJPluginModel.

testGetPsfGibsonLanniNi

public void **testGetPsfGibsonLanniNi**()
Test of getPsfGibsonLanniNi, of class IJPluginModel.

testGetPsfGibsonLanniNi0

public void **testGetPsfGibsonLanniNi0**()
Test of getPsfGibsonLanniNi0, of class IJPluginModel.

testGetPsfGibsonLanniNs

public void **testGetPsfGibsonLanniNs**()
Test of getPsfGibsonLanniNs, of class IJPluginModel.

testGetPsfGibsonLanniNumBasis

public void **testGetPsfGibsonLanniNumBasis**()
Test of getPsfGibsonLanniNumBasis, of class IJPluginModel.

testGetPsfGibsonLanniNumSamples

public void **testGetPsfGibsonLanniNumSamples**()
Test of getPsfGibsonLanniNumSamples, of class IJPluginModel.

testGetPsfGibsonLanniOversampling

public void **testGetPsfGibsonLanniOversampling**()
Test of getPsfGibsonLanniOversampling, of class IJPluginModel.

testGetPsfGibsonLanniResPsf

public void **testGetPsfGibsonLanniResPsf**()
Test of getPsfGibsonLanniResPsf, of class IJPluginModel.

testGetPsfGibsonLanniResPsfAxial

public void **testGetPsfGibsonLanniResPsfAxial**()
Test of getPsfGibsonLanniResPsfAxial, of class IJPluginModel.

testGetPsfGibsonLanniSizeX

public void **testGetPsfGibsonLanniSizeX**()
Test of getPsfGibsonLanniSizeX, of class IJPluginModel.

testGetPsfGibsonLanniSizeY

public void **testGetPsfGibsonLanniSizeY**()
Test of getPsfGibsonLanniSizeY, of class IJPluginModel.

testGetPsfGibsonLanniSolver

public void **testGetPsfGibsonLanniSolver**()
Test of getPsfGibsonLanniSolver, of class IJPluginModel.

testGetPsfGibsonLanniTg

public void **testGetPsfGibsonLanniTg**()
Test of getPsfGibsonLanniTg, of class IJPluginModel.

testGetPsfGibsonLanniTg0

public void **testGetPsfGibsonLanniTg0**()
Test of getPsfGibsonLanniTg0, of class IJPluginModel.

testGetPsfGibsonLanniTi0

public void **testGetPsfGibsonLanniTi0**()
Test of getPsfGibsonLanniTi0, of class IJPluginModel.

testGetStageX

public void **testGetStageX**()
Test of getStageX method, of class IJPluginModel.

testGetStageY

public void **testGetStageY**()
Test of getStageY method, of class IJPluginModel.

testGetStageZ

public void **testGetStageZ**()
Test of getStageZ method, of class IJPluginModel.

testGetStormKBl

public void **testGetStormKBl**()
Test of getStormKBl method, class IJPluginModel.

testGetStormKDark

public void **testGetStormKDark**()
Test of getStormKDark method, class IJPluginModel.

testGetStormKDarkRecovery

public void **testGetStormKDarkRecovery**()
Test of getStormKDarkRecovery method, class IJPluginModel.

testGetStormKDarkRecoveryConstant

public void **testGetStormKDarkRecoveryConstant**()
Test of getStormKDarkRecoveryConstant method, class IJPluginModel.

testGetStormKTriplet

public void **testGetStormKTriplet**()
Test of getStormKTriplet method, class IJPluginModel.

testGetStormKTripletRecovery

public void **testGetStormKTripletRecovery**()
Test of getStormKTripletRecovery method, class IJPluginModel.

6.4.9 Server

public class **Server** extends PlugInFrame
The form for configuring the SASS server from within ImageJ.

Author Kyle M. Douglass

Constructors

Server

public **Server** (String *title*)
Creates new form Server

Parameters

- **title** – The title of the form.

Server

public **Server** ()
Creates new form Server

Methods

run

public void **run** (String *arg*)
Show the frame and initialize backend.

Parameters

- **arg** –

6.4.10 ServerModel

public class **ServerModel**
Contains the GUI form data for the SASS server.

Author Kyle M. Douglass

Methods

getConfigFile

public String **getConfigFile** ()

getPort

public int **getPort** ()

getPortTextEnabled

public boolean **getPortTextEnabled** ()

getSelectConfigButtonEnabled

public boolean **getSelectConfigButtonEnabled**()

getServer

public *RPCServer* **getServer**()

getSimulationModel

public *IJPluginModel* **getSimulationModel**()

getStartButtonEnabled

public boolean **getStartButtonEnabled**()

getStopButtonEnabled

public boolean **getStopButtonEnabled**()

setConfigFile

public void **setConfigFile**(String *filename*)

setPort

public void **setPort**(int *port*)

setPortTextEnabled

public void **setPortTextEnabled**(boolean *enabled*)

setSelectConfigButtonEnabled

public void **setSelectConfigButtonEnabled**(boolean *enabled*)

setServer

public void **setServer**(*RPCServer server*)

setSimulationModel

public void **setSimulationModel**(*IJPluginModel simulationModel*)

setStartButtonEnabled

public void **setStartButtonEnabled**(boolean *enabled*)

setStopButtonEnabled

public void **setStopButtonEnabled**(boolean *enabled*)

6.4.11 SimulatorStatusFrame

public class **SimulatorStatusFrame** extends javax.swing.JFrame

Frame that displays the current status and recent history of the simulation. The layout for the status frame was inspired by Karl Bellve's pgFocus GUI: http://big.umassmed.edu/wiki/index.php/PgFocus

Author Kyle M. Douglass

Fields

SUBPLOT_COUNT

public final int **SUBPLOT_COUNT**

Constructors

SimulatorStatusFrame

public **SimulatorStatusFrame**(String *groundTruthYLabel*, String *analyzerYLabel*, String *setpointYLabel*, String *outputYLabel*)

Creates a new status frame.

Parameters

- **groundTruthYLabel** – The y-axis label for the ground truth signal.
- **analyzerYLabel** – The units output by the analyzer.
- **setpointYLabel** – The units of the controller setpoint.
- **outputYLabel** – The units output by the controller.

Methods

updateGraph

public void **updateGraph**(int *frame*, double *trueCount*, double *estimate*, double *setpoint*, double *laser*)

Adds a single new time point to the plot.

Parameters

- **frame** – The frame number
- **trueCount** – The true number of emitting molecules.
- **estimate** – Analyzer's estimate of the number of emitting molecules.

- **setpoint** – The controller's setpoint value.
- **laser** – The output of the laser.

6.4.12 Worker

class **Worker** extends Thread

Fields

stop

public boolean **stop**

Constructors

Worker

public **Worker** (*App app*, Controller *controller*, Analyzer *active_analyzer*, *ImageS imp*)

Methods

run

public void **run** ()

6.5 ch.epfl.leb.sass.logging

6.5.1 Listener

public interface **Listener**

Defines common methods for listeners, i.e. objects that track `Observables`.

Author Kyle M. Douglass

Methods

update

public void **update** (Object *data*)

This method is called by an Observable when its state has changed.

Parameters

- **data** – The data object that is passed from the Observable.

6.5.2 Message

public interface **Message** extends Serializable

Defines methods that all logging messages should posses.

Author Kyle M. Douglass

Methods

getType

public *MessageType* **getType** ()

A unique identifier for the message type.

Returns The message type.

toJson

public JsonElement **toJson** ()

The message as a JSON element.

Returns A JsonElement represented as a message.

6.5.3 MessageType

public enum **MessageType**

Author Kyle M. Douglass

Enum Constants

FLUOROPHORE

public static final *MessageType* **FLUOROPHORE**

The type of messages that are sent by individual fluorophores.

LASER_POWER_CHANGE

public static final *MessageType* **LASER_POWER_CHANGE**

This message indicates a change in laser power.

6.5.4 Observable

public interface **Observable**

Defines interface methods for logging changes in a microscope component.

Author Kyle M. Douglass

Methods

addListener

public void **addListener** (*Listener listener*)

Adds a new listener to the list of subscribed listeners.

Parameters

- **listener** – The listener to add to the list of subscribed listeners.

deleteListener

public void **deleteListener** (*Listener listener*)

Deletes a listener from the list of subscribed listeners.

Parameters

- **listener** – The listener to delete from the list of listeners.

notifyListeners

public void **notifyListeners** ()

Notifies all subscribed listeners to a change in the Observable's state. This method should only be called if setChanged() has been called.

notifyListeners

public void **notifyListeners** (Object *data*)

Notifies all subscribed listeners of a state change and pushes the data.

Parameters

- **data** – The data object to push to the listeners.

setChanged

public void **setChanged** ()

Indicates that the state of this Observable has been changed.

6.5.5 WrongMessageTypeException

public class **WrongMessageTypeException** extends Exception

Raised when a Listener receives an unexpected message type.

Author Kyle M. Douglass

Constructors

WrongMessageTypeException

public **WrongMessageTypeException**()

WrongMessageTypeException

public **WrongMessageTypeException**(String *msg*)

Parameters

- **msg** – An error message describing what raised this exception.

6.6 ch.epfl.leb.sass.logging.internal

6.6.1 AbstractObservable

public abstract class **AbstractObservable** implements *Observable*

Provides functionality common to all Observables.

Author Kyle M. Douglass

Fields

LOGGER

protected static final Logger **LOGGER**

changed

protected boolean **changed**

A flag indicating whether the state of this object has changed. This flag is used only when notifying listeners of a state change.

listeners

protected ArrayList<*Listener*> **listeners**

The list of listeners that are tracking this object.

Methods

addListener

public void **addListener**(*Listener listener*)

Adds a new listener to the list of subscribed listeners.

deleteListener

public void **deleteListener** (*Listener listener*)
Deletes a listener from the list of subscribed listeners.

notifyListeners

public void **notifyListeners** ()
Notifies all subscribed listeners to a change in the Observable's state. This method should only be called if setChanged() has been called.

notifyListeners

public void **notifyListeners** (Object *data*)
Notifies all subscribed listeners of a state change and pushes the data.

Parameters

- **data** – The data object to push to the listeners.

setChanged

public void **setChanged** ()
Indicates that the state of this Observable has been changed.

6.6.2 FluorophoreStateTransition

public class **FluorophoreStateTransition** implements *Message*
A message containing information about a fluorophore state transition.

Author Kyle M. Douglass

Fields

CURRENT_STATE

public final int **CURRENT_STATE**

ID

public final int **ID**

NEXT_STATE

public final int **NEXT_STATE**

TIME_ELAPSED

public final double **TIME_ELAPSED**

TYPE

public final *MessageType* **TYPE**

Constructors

FluorophoreStateTransition

public **FluorophoreStateTransition** (int *id*, double *timeElapsed*, int *currentState*, int *nextState*)

Methods

getType

public *MessageType* **getType** ()
An identifier that indicates where this message originated from.

Returns The message type.

toJson

public JsonElement **toJson** ()
Returns the the message as a JSON string.

Returns The properties of the fluorophore as a JSON string.

6.6.3 FluorophoreStateTransition.FluorophoreStateTransitionSerializer

class **FluorophoreStateTransitionSerializer** implements JsonSerializer<*FluorophoreStateTransition*>

Methods

serialize

public JsonElement **serialize** (*FluorophoreStateTransition* *src*, Type *typeOfSrc*, JsonSerializationContext *context*)

6.6.4 FluorophoreStateTransitionTest

public class **FluorophoreStateTransitionTest**
Tests of the FluorophoreStateTransition class.

Author Kyle M. Douglass

Constructors

FluorophoreStateTransitionTest

public **FluorophoreStateTransitionTest** ( )

Methods

testToJson

public void **testToJson** ( )
Test of toJson method, of class FluorophoreStateTransition.

6.6.5 LaserPowerChange

public class **LaserPowerChange** implements *Message*
A message containing information about a fluorophore state transition.

Author Kyle M. Douglass

Fields

POWER

public final double **POWER**

TYPE

public final *MessageType* **TYPE**

Constructors

LaserPowerChange

public **LaserPowerChange** (double *power*)

Methods

getType

public *MessageType* **getType** ( )
An identifier that indicates where this message originated from.

Returns The message type.

toJson

public JsonElement **toJson** ( )
Returns the the message as a JSON string.

Returns The properties of the fluorophore as a JSON string.

6.6.6 LaserPowerChange.LaserPowerChangeSerializer

class **LaserPowerChangeSerializer** implements JsonSerializer<*LaserPowerChange*>

Methods

serialize

public JsonElement **serialize** (*LaserPowerChange src*, Type *typeOfSrc*, JsonSerializationContext *context*)

6.6.7 LaserPowerChangeTest

public class **LaserPowerChangeTest**
Unit tests for the LaserPowerChange Message.

Author Kyle M. Douglass

Constructors

LaserPowerChangeTest

public **LaserPowerChangeTest** ( )

Methods

testGetType

public void **testGetType** ( )
Test of getType method, of class LaserPowerChange.

testToJson

public void **testToJson** ( )
Test of toJson method, of class LaserPowerChange.

6.7 ch.epfl.leb.sass.models

6.7.1 Microscope

public class **Microscope** implements Serializable

Integrates all the components into one microscope.

Constructors

Microscope

public **Microscope** (*DefaultCamera.Builder cameraBuilder*, *DefaultLaser.Builder laserBuilder*, *DefaultObjective.Builder objectiveBuilder*, *PSFBuilder psfBuilder*, *DefaultStage.Builder stageBuilder*, *FluorophoreCommandBuilder fluorBuilder*, *FluorophoreDynamicsBuilder fluorDynamicsBuilder*, *ObstructorCommandBuilder obstructorBuilder*, *BackgroundCommandBuilder backgroundBuilder*, *IlluminationBuilder illuminationBuilder*)

Initializes the microscope for simulations.

Parameters

- **cameraBuilder** –
- **laserBuilder** –
- **objectiveBuilder** –
- **psfBuilder** –
- **stageBuilder** –
- **fluorBuilder** – Positions fluorophore's within the field of view.
- **fluorDynamicsBuilder** –
- **obstructorBuilder** – Creates the obstructors, e.g. fiducials.
- **backgroundBuilder** – Creates the background signal on the image.
- **illuminationBuilder** – Creates the illumination profile.

Methods

getFluorophores

public List<*Fluorophore*> **getFluorophores** ()

Returns references to the fluorophores in the sample.

Returns The sample's Fluorophore objects.

getFovSize

public double **getFovSize** ()

Returns the size of the field-of-view in object space units.

Returns size of current FOV in object space units.

getLaserPower

public double **getLaserPower**()

Return current power of the laser.

Returns laser power

getObjectSpacePixelSize

public double **getObjectSpacePixelSize**()

The size of a pixel after division by the objective magnification.

Returns Length of one pixel side in object space units

getOnEmitterCount

public double **getOnEmitterCount**()

Returns the number of currently active emitters.

Returns number of shining emitters

getResolution

public int[] **getResolution**()

Return the number of camera pixels in x and y.

Returns 2D array with number of pixels in x and y.

setLaserPower

public void **setLaserPower**(double *laserPower*)

Modifies the laser power to desired value.

Parameters

- **laserPower** – new laser power

simulateFrame

public *ImageS* **simulateFrame**()

Generates a new frame and moves the device state forward. First the obstructors are drawn on the frame, then the fluorophores, and finally noise.

Returns A simulated image of the next camera frame.

toJsonCamera

public JsonElement **toJsonCamera**()

Returns information about the camera.

Returns A JsonElement containing information about the camera.

toJsonFluorescence

public JsonElement **toJsonFluorescence**()
Returns information about the sample fluorophores.

Returns A JsonObject containing information about the fluorophores.

toJsonLaser

public JsonElement **toJsonLaser**()
Returns information about the laser.

Returns A JsonElement containing information about the laser.

toJsonObjective

public JsonElement **toJsonObjective**()
Returns information about the objective.

Returns A JsonElement containing information about the stage.

toJsonStage

public JsonElement **toJsonStage**()
Returns information about the stage.

Returns A JsonElement containing information about the stage.

6.7.2 MicroscopeIT

public class **MicroscopeIT**
Integration tests for the Microscope class.

Author Kyle M. Douglass

See also: `https://stackoverflow.com/questions/2606572/junit-splitting-integration-test-and-unit-tests`

Methods

setUp

public void **setUp**()
Sets up a basic Microscope for an acquisition simulation.

testGetFluorophores

public void **testGetFluorophores**()
Test of getFluorophores method, of class Microscope.

testGetFovSize

public void **testGetFovSize**()
Test of getFovSize method, of class Microscope.

testGetObjectSpacePixelSize

public void **testGetObjectSpacePixelSize**()
Test of getObjectSpacePixelSize method, of class Microscope.

testGetOnEmitterCount

public void **testGetOnEmitterCount**()
Test of getOnEmitterCount method, of class Microscope.

testGetResolution

public void **testGetResolution**()
Test of getResolution method, of class Microscope.

testGetSetLaserPower

public void **testGetSetLaserPower**()
Test of setLaserPower and getLaserPower methods, of class Microscope.

testSimulateFrame

public void **testSimulateFrame**()
Test of simulateFrame method, of class Microscope.

testToJsonCamera

public void **testToJsonCamera**()
Test of toJsonCamera method, of class Microscope.

testToJsonFluorescence

public void **testToJsonFluorescence**()
Test of toJsonFluorescence method, of class Microscope.

testToJsonLaser

public void **testToJsonLaser**()
Test of toJsonLaser method, of class Microscope.

testToJsonObjective

public void **testToJsonObjective**()
Test of toJsonObjective method, of class Microscope.

testToJsonStage

public void **testToJsonStage**()
Test of toJsonStage method, of class Microscope.

6.7.3 Model

public interface **Model** extends Serializable
Defines common methods possessed by all models employed by the microscope.

Author Kyle M. Douglass

Methods

toJson

public JsonElement **toJson**()
Outputs the model's properties as a JSON element.

Returns A JSON tree describing the model's properties.

6.8 ch.epfl.leb.sass.models.backgrounds

6.8.1 BackgroundCommand

public interface **BackgroundCommand** extends Serializable
Commands for creating a background in an image.

Author Kyle M. Douglass

Methods

generateBackground

public float[][] **generateBackground**()

6.8.2 BackgroundCommandBuilder

public interface **BackgroundCommandBuilder**
Interface BackgroundCommand builders.

Author Kyle M. Douglass

Methods

build

public *BackgroundCommand* **build**()

nX

public *BackgroundCommandBuilder* **nX**(int *nX*)

Sets the number of pixels of the images in the x-direction.

Parameters

- **nX** – Number of pixels in x.

Returns The very same builder object.

nY

public *BackgroundCommandBuilder* **nY**(int *nY*)

Sets the number of pixels of the images in the y-direction.

Parameters

- **nY** – Number of pixels in y.

Returns The very same builder object.

6.9 ch.epfl.leb.sass.models.backgrounds.internal.commands

6.9.1 GenerateBackgroundFromFile

public final class **GenerateBackgroundFromFile** implements *BackgroundCommand*

Constant overlay loaded from a tif image.

Author Marcel Stefko

Methods

generateBackground

public float[][] **generateBackground**()

Creates the background image.

Returns The background image.

6.9.2 GenerateBackgroundFromFile.Builder

public static class **Builder** implements *BackgroundCommandBuilder*

Methods

build

public *GenerateBackgroundFromFile* **build**()

file

public *Builder* **file**(File *file*)

nX

public *Builder* **nX**(int *nX*)

nY

public *Builder* **nY**(int *nY*)

6.9.3 GenerateBackgroundFromFileTest

public class **GenerateBackgroundFromFileTest**

Tests for generating a constant background from a .tif file.

Author Kyle M. Douglass

Fields

tempDir

public TemporaryFolder **tempDir**

Constructors

GenerateBackgroundFromFileTest

public **GenerateBackgroundFromFileTest**()

Methods

setUp

public void **setUp**()

Creates a test .tif file as an example background.

testGenerateBackground

public void **testGenerateBackground**()
Test of generateBackground method, of class GenerateBackgroundFromFile.

6.9.4 GenerateRandomBackground

public class **GenerateRandomBackground** implements *BackgroundCommand*
Generates random background patterns from a simplex noise generator.

Author Kyle M. Douglass

Methods

generateBackground

public float[][] **generateBackground**()
Create the random background signal.

Returns A 2D array of background photons for each pixel.

6.9.5 GenerateRandomBackground.Builder

public static class **Builder** implements *BackgroundCommandBuilder*

Methods

build

public *GenerateRandomBackground* **build**()

featureSize

public *Builder* **featureSize**(double *featureSize*)

max

public *Builder* **max**(float *max*)

min

public *Builder* **min**(float *min*)

nX

public *Builder* **nX**(int *nX*)

nY

public *Builder* **nY** (int *nY*)

seed

public *Builder* **seed** (int *seed*)

6.9.6 GenerateRandomBackgroundTest

public class **GenerateRandomBackgroundTest**

Author Kyle M. Douglass

Constructors

GenerateRandomBackgroundTest

public **GenerateRandomBackgroundTest** ()

Methods

testGenerateBackground

public void **testGenerateBackground** ()
Test of generateBackground method, of class GenerateRandomBackground.

6.9.7 GenerateUniformBackground

public final class **GenerateUniformBackground** implements *BackgroundCommand*

Author Kyle M. Douglass

Methods

generateBackground

public float[][] **generateBackground** ()
Create the background signal.

Returns A 2D array of background photons for each pixel.

6.9.8 GenerateUniformBackground.Builder

public static class **Builder** implements *BackgroundCommandBuilder*
Creates the command to generate a uniform background.

Methods

backgroundSignal

public *Builder* **backgroundSignal** (float *backgroundSignal*)

build

public *GenerateUniformBackground* **build** ()
Builds the command.

Returns The command to build a uniform background.

nX

public *Builder* **nX** (int *nX*)

nY

public *Builder* **nY** (int *nY*)

6.9.9 OpenSimplexNoise

public class **OpenSimplexNoise**

Constructors

OpenSimplexNoise

public **OpenSimplexNoise** ()

OpenSimplexNoise

public **OpenSimplexNoise** (short[] *perm*)

OpenSimplexNoise

public **OpenSimplexNoise** (long *seed*)

Methods

eval

public double **eval** (double *x*, double *y*)

eval

public double **eval** (double *x*, double *y*, double *z*)

eval

public double **eval** (double *x*, double *y*, double *z*, double *w*)

6.10 ch.epfl.leb.sass.models.components

6.10.1 Camera

public interface **Camera** extends *Model*

Common methods of the Camera interface.

Author Kyle M. Douglass

Methods

getAduPerElectron

public double **getAduPerElectron** ()

The number analog-to-digital units generated by each electron.

Returns The number analog-to-digital units (ADUs) generated by each electron.

getBaseline

public int **getBaseline** ()

The offset added to each pixel in analog-to-digital units (ADUs).

Returns The offset added to each pixel in analog-to-digital units (ADUs).

getDarkCurrent

public double **getDarkCurrent** ()

The camera's dark current in electrons/second/pixel.

Returns The camera's dark current in electrons/second/pixel.

getEmGain

public int **getEmGain** ()

The electron-multiplication gain. This will be zero if there is no EM gain of the camera.

Returns The electron-multiplication gain.

getNX

public int **getNX**()

Returns The number of pixels in x.

getNY

public int **getNY**()

Returns The number of pixels in y.

getPixelSize

public double **getPixelSize**()

The physical size of a pixel. This is the image space, NOT object space, pixel size, i.e. it does not depend on the optics.

Returns The physical size of a pixel.

getQuantumEfficiency

public double **getQuantumEfficiency**()

The camera's quantum efficiency. This number determines how many photons are converted into electrons on average per pixel.

Returns The camera's quantum efficiency.

getReadoutNoise

public double **getReadoutNoise**()

The readout noise per pixel in units of electrons (standard deviation).

Returns The readout nosie per pixel.

getThermalNoise

public double **getThermalNoise**()

The camera's thermal noise units of electrons/frame/pixel.

Returns The camera's thermal noise.

6.10.2 DefaultCameraTest

public class **DefaultCameraTest**

Unit tests for the DefaultCamera class.

Author Kyle M. Douglass

Constructors

DefaultCameraTest

public **DefaultCameraTest** ()

Methods

testGetAduPerElectron

public void **testGetAduPerElectron** ()
Test of getAduPerElectron method, of class DefaultCamera.

testGetBaseline

public void **testGetBaseline** ()
Test of getBaseline method, of class DefaultCamera.

testGetDarkCurrent

public void **testGetDarkCurrent** ()
Test of getDarkCurrent method, of class DefaultCamera.

testGetEmGain

public void **testGetEmGain** ()
Test of getEmGain method, of class DefaultCamera.

testGetNX

public void **testGetNX** ()
Test of getNX method, of class DefaultCamera.

testGetNY

public void **testGetNY** ()
Test of getNY method, of class DefaultCamera.

testGetPixelSize

public void **testGetPixelSize** ()
Test of getPixelSize method, of class DefaultCamera.

testGetQuantumEfficiency

public void **testGetQuantumEfficiency**()
Test of getQuantumEfficiency method, of class DefaultCamera.

testGetReadoutNoise

public void **testGetReadoutNoise**()
Test of getReadoutNoise method, of class DefaultCamera.

testGetThermalNoise

public void **testGetThermalNoise**()
Test of getThermalNoise method, of class DefaultCamera.

testToJson

public void **testToJson**()
Test of toJson method, of class DefaultCamera.

6.10.3 DefaultLaserTest

public class **DefaultLaserTest**

Author Kyle M. Douglass

Constructors

DefaultLaserTest

public **DefaultLaserTest**()

Methods

setUp

public void **setUp**()

testGetPower

public void **testGetPower**()
Test of getPower method, of class DefaultLaser.

testGetWavelength

public void **testGetWavelength**()
Test of getWavelength method, of class DefaultLaser.

testSetPower

public void **testSetPower**()
Test of setPower method, of class DefaultLaser.

testToJson

public void **testToJson**()
Test of toJson method, of class DefaultLaser.

6.10.4 DefaultObjectiveTest

public class **DefaultObjectiveTest**

Author Kyle M. Douglass

Constructors

DefaultObjectiveTest

public **DefaultObjectiveTest**()

Methods

testAiryFWHM

public void **testAiryFWHM**()
Test of psfFWHM method, of class DefaultObjective.

testAiryRadius

public void **testAiryRadius**()
Test of psfFWHM method, of class DefaultObjective.

testGetMag

public void **testGetMag**()
Test of getMag method, of class DefaultObjective.

testGetNA

public void **testGetNA**()
Test of getNA method, of class DefaultObjective.

testToJson

public void **testToJson**()
Test of toJson method, of class DefaultLaser.

6.10.5 DefaultStageTest

public class **DefaultStageTest**

Author Kyle M. Douglass

Constructors

DefaultStageTest

public **DefaultStageTest**()

Methods

setUp

public void **setUp**()

testGetX

public void **testGetX**()
Test of getX method, of class DefaultStage.

testGetY

public void **testGetY**()
Test of getY method, of class DefaultStage.

testGetZ

public void **testGetZ**()
Test of getZ method, of class DefaultStage.

testSetX

public void **testSetX**()
Test of setX method, of class DefaultStage.

testSetY

public void **testSetY**()
Test of setY method, of class DefaultStage.

testSetZ

public void **testSetZ**()
Test of setZ method, of class DefaultStage.

testToJson

public void **testToJson**()
Test of toJson method, of class DefaultLaser.

6.10.6 Laser

public interface **Laser** extends *Model*, *Observable*
Defines methods common to Lasers.

Author Kyle M. Douglass

Methods

getPower

public double **getPower**()
Returns the current power if the laser.

Returns The current laser power.

getWavelength

public double **getWavelength**()
Returns the wavelength of the laser.

Returns The laser's wavelength.

setPower

public void **setPower**(double *newPower*)
Sets the light source's power.

Parameters

- **newPower** – The power of the light source.

6.10.7 Objective

public interface **Objective** extends *Model*

Methods common to all microscope objectives.

Author Kyle M. Douglass

Methods

airyFWHM

public double **airyFWHM** (double *wavelength*)

Computes the full width at half maximum of the Airy disk. Units are the same as those of wavelength.

Parameters

- **wavelength** –

Returns Full width at half maximum size of the Airy disk.

airyRadius

public double **airyRadius** (double *wavelength*)

Computes the radius of the Airy disk. Units are the same as those of wavelength.

Parameters

- **wavelength** –

Returns Distance from center of Airy disk to first minimum.

getMag

public double **getMag** ()

Returns the objective's magnification.

Returns The objective's magnification.

getNA

public double **getNA** ()

Returns the objective's numerical aperture.

Returns The objective's numerical aperture

6.10.8 Stage

public interface **Stage** extends *Model*

Defines methods common to all Stages.

Author Kyle M. Douglass

Methods

getX

public double **getX**()
Returns the stage's x-position.

Returns The stage's x-position.

getY

public double **getY**()
Returns the stage's y-position.

Returns The stage's y-position.

getZ

public double **getZ**()
Returns the stage's z-position.

Returns The stage's z-position.

setX

public void **setX**(double *x*)
Set the stage's x-position.

Parameters

- **x** –

setY

public void **setY**(double *y*)
Set the stage's y-position.

Parameters

- **y** –

setZ

public void **setZ**(double *z*)
Set the stage's z-position.

Parameters

- **z** –

6.11 ch.epfl.leb.sass.models.components.internal

6.11.1 DefaultCamera

public final class **`DefaultCamera`** implements *Camera*

Represents the parameters of the camera.

Methods

getAduPerElectron

public double **`getAduPerElectron`**()

getBaseline

public int **`getBaseline`**()

getDarkCurrent

public double **`getDarkCurrent`**()

getEmGain

public int **`getEmGain`**()

getNX

public int **`getNX`**()

Returns The number of pixels in x.

getNY

public int **`getNY`**()

Returns The number of pixels in y.

getPixelSize

public double **`getPixelSize`**()

getQuantumEfficiency

public double **`getQuantumEfficiency`**()

getReadoutNoise

public double **getReadoutNoise**()

getThermalNoise

public double **getThermalNoise**()

toJson

public JsonElement **toJson**()

Outputs the camera's properties as a JSON element.

Returns A JSON tree describing the camera's properties.

6.11.2 DefaultCamera.Builder

public static class **Builder**

Methods

aduPerElectron

public *Builder* **aduPerElectron**(double *aduPerElectron*)

baseline

public *Builder* **baseline**(int *baseline*)

build

public *DefaultCamera* **build**()

darkCurrent

public *Builder* **darkCurrent**(double *darkCurrent*)

emGain

public *Builder* **emGain**(int *emGain*)

nX

public *Builder* **nX**(int *nX*)

nY

public *Builder* **nY** (int *nY*)

pixelSize

public *Builder* **pixelSize** (double *pixelSize*)

quantumEfficiency

public *Builder* **quantumEfficiency** (double *quantumEfficiency*)

readoutNoise

public *Builder* **readoutNoise** (double *readoutNoise*)

thermalNoise

public *Builder* **thermalNoise** (double *thermalNoise*)

6.11.3 DefaultCamera.DefaultCameraSerializer

class **DefaultCameraSerializer** implements JsonSerializer<*DefaultCamera*>

Methods

serialize

public JsonElement **serialize** (*DefaultCamera* *src*, Type *typeOfSrc*, JsonSerializationContext *context*)

6.11.4 DefaultLaser

public class **DefaultLaser** extends *AbstractObservable* implements *Laser*

A source of light for illuminating the sample.

Methods

getPower

public double **getPower** ()

Returns the current power.

Returns current laser power

getWavelength

public double **getWavelength**()

Returns the wavelength of the laser.

Returns The laser's wavelength.

setPower

public void **setPower**(double *newPower*)

Sets the light source's power. If the value is not within the limits, set it to the the closest allowed value.

Parameters

- **newPower** – The power of the light source.

toJson

public JsonElement **toJson**()

Outputs the laser's properties as a JSON element.

Returns A JSON tree describing the laser's properties.

6.11.5 DefaultLaser.Builder

public static class **Builder**

Methods

build

public *DefaultLaser* **build**()

currentPower

public *Builder* **currentPower**(double *currentPower*)

maxPower

public *Builder* **maxPower**(double *maxPower*)

minPower

public *Builder* **minPower**(double *minPower*)

wavelength

public *Builder* **wavelength**(double *wavelength*)

6.11.6 DefaultLaser.DefaultLaserSerializer

class **DefaultLaserSerializer** implements JsonSerializer<*DefaultLaser*>

Methods

serialize

public JsonElement **serialize** (*DefaultLaser src*, Type *typeOfSrc*, JsonSerializationContext *context*)

6.11.7 DefaultObjective

public final class **DefaultObjective** implements *Objective*

Properties related to the microscope objective.

Author Kyle M. Douglass

Methods

airyFWHM

public double **airyFWHM** (double *wavelength*)

Computes the full width at half maximum of the Airy disk. Units are the same as those of wavelength.

Parameters

- **wavelength** –

Returns Full width at half maximum size of the Airy disk.

airyRadius

public double **airyRadius** (double *wavelength*)

Computes the radius of the Airy disk. Units are the same as those of wavelength.

Parameters

- **wavelength** –

Returns Distance from center of Airy disk to first minimum.

getMag

public double **getMag** ()

Returns The objective's magnification.

getNA

public double **getNA** ()

Returns The objective's numerical aperture

toJson

public JsonElement **toJson** ( )

Outputs the laser's properties as a JSON element.

Returns A JSON tree describing the laser's properties.

6.11.8 DefaultObjective.Builder

public static class **Builder**

Methods

NA

public *Builder* **NA** (double *NA*)

build

public *DefaultObjective* **build** ( )

mag

public *Builder* **mag** (double *mag*)

6.11.9 DefaultObjective.DefaultObjectiveSerializer

class **DefaultObjectiveSerializer** implements JsonSerializer<*DefaultObjective*>

Methods

serialize

public JsonElement **serialize** (*DefaultObjective* *src*, Type *typeOfSrc*, JsonSerializationContext *context*)

6.11.10 DefaultStage

public class **DefaultStage** implements *Stage*

The sample stage.

Author Kyle M. Douglass

Methods

getX

public double **getX**()

Returns The stage's x-position.

getY

public double **getY**()

Returns The stage's y-position.

getZ

public double **getZ**()

Returns The stage's z-position.

setX

public void **setX**(double *x*)
Set the stage's x-position.

Parameters

- **x** –

setY

public void **setY**(double *y*)
Set the stage's y-position.

Parameters

- **y** –

setZ

public void **setZ**(double *z*)
Set the stage's z-position.

Parameters

- **z** –

toJson

public JsonElement **toJson**()
Outputs the laser's properties as a JSON element.

Returns A JSON tree describing the laser's properties.

6.11.11 DefaultStage.Builder

public static class **Builder**

Builder for creating stage instances.

Methods

build

public *DefaultStage* **build** ( )

x

public *Builder* **x** (double *x*)

y

public *Builder* **y** (double *y*)

z

public *Builder* **z** (double *z*)

6.11.12 DefaultStage.DefaultStageSerializer

class **DefaultStageSerializer** implements JsonSerializer<*DefaultStage*>

Methods

serialize

public JsonElement **serialize** (*DefaultStage* *src*, Type *typeOfSrc*, JsonSerializationContext *context*)

6.12 ch.epfl.leb.sass.models.emitters

6.12.1 AbstractEmitterTest

public class **AbstractEmitterTest**

Author douglass

Constructors

AbstractEmitterTest

public **AbstractEmitterTest** ( )

Methods

testGetPixelsWithinRadiusLessThanOne

public void **testGetPixelsWithinRadiusLessThanOne**()

Test of getPixelsWithinRadius method, of class Camera. Tests that only the pixel containing the point is returned if the radius is less than one.

testGetPixelsWithinRadiusOfOrigin

public void **testGetPixelsWithinRadiusOfOrigin**()

Test of getPixelsWithinRadius method, of class Camera. Tests that all pixels within a certain radius of the origin are correctly returned.

6.13 ch.epfl.leb.sass.models.emitters.internal

6.13.1 AbstractEmitter

public abstract class **AbstractEmitter** extends Point2D.Double

A point source of light and tools to compute its signature on a digital detector. Emitters are general point sources of light that are imaged by an optical system and recorded by a digital sensor. The AbstractEmitter class contains tools for generating the digital images of point sources without any regard for the dynamics of the of the signal (apart from photon shot noise). Classes that extend the AbstractEmitter class are intended to implement the dynamics of the source's signal.

Author Marcel Stefko, Kyle M. Douglass

Fields

builder

protected *PSFBuilder* **builder**

A builder for creating/updating the emitter PSF.

id

protected int **id**

A unique ID assigned to this emitter.

numberOfEmitters

protected static int **numberOfEmitters**

Running total of the number of emitters.

pixel_list

protected ArrayList<*Pixel*> **pixel_list**
List of pixels which are affected by this emitter's light (these pixels need to be updated when the emitter is on).

poisson

protected Poisson **poisson**
Poisson RNG for flickering simulation.

psf

protected *PSF* **psf**
The PSF model that's created by the emitter.

z

public double **z**
The emitter's z-position.

Constructors

AbstractEmitter

public **AbstractEmitter** (double *x*, double *y*, double *z*, *PSFBuilder psfBuilder*)
Creates the emitter at given position, and calculates its image from the PSF and camera.

Parameters

- **x** – x-position in image [pixels, with sub-pixel precision]
- **y** – y-position in image [pixels, with sub-pixel precision]
- **z** – z-position in image [pixels, with sub-pixel precision]
- **psfBuilder** – Builder for creating the emitter's PSF.

Methods

applyTo

public void **applyTo** (float[][] *pixels*)
Simulates the brightness pattern of this emitter for the next frame duration, and renders the emitter onto the image.

Parameters

- **pixels** – image to be drawn on

flicker

protected double **flicker** (double *baseBrightness*)

Applies Poisson statistics to simulate flickering of an emitter.

Parameters

- **baseBrightness** – mean of Poisson distribution to draw from

Returns actual brightness of this emitter for this frame

generate_signature_for_pixel

protected double **generate_signature_for_pixel** (int *x*, int *y*, double *camera_fwhm_digital*)

Returns the signature that this emitter leaves on a given pixel (what fraction of this emitter's photons hits this particular pixel).

Parameters

- **x** – pixel x-position
- **y** – pixel y-position
- **camera_fwhm_digital** – camera fwhm value

Throws

- **MathException** –

Returns signature value for this pixel

getId

public int **getId** ()

Returns the emitter's ID.

Returns The unique integer identifying the emitter.

getPSF

public *PSF* **getPSF** ()

Returns the emitter's PSF model.

Returns The PSF model used to create the image of this emitter.

getPixelList

public ArrayList<*Pixel*> **getPixelList** ()

Returns list of pixels which need to be drawn on the image to accurately render the emitter.

Returns list of Pixels

getPixelsWithinRadius

public static final ArrayList<*Pixel*> **getPixelsWithinRadius** (Point2D *point*, double *radius*)

Returns a list of pixels within a certain radius from a point. This method locates all the pixels within a circular area surrounding a given two-dimensional point whose center lies at (x, y). The coordinate of a pixel is assumed to lie at the pixel's center, and a pixel is within a given radius of another if the pixel's center lies within this circle.

Parameters

- **point** –
- **radius** – radius value [pixels]

Returns list of Pixels with pre-calculated signatures

get_pixels_within_radius

protected final ArrayList<*Pixel*> **get_pixels_within_radius** (double *radius*, double *camera_fwhm_digital*)

Returns a list of pixels within a certain radius from this emitter (so that their signature is precalculated). Pixels outside this radius are considered to have negligible signature.

Parameters

- **radius** – radius value [pixels]
- **camera_fwhm_digital** – camera fwhm value

Returns list of Pixels with precalculated signatures

setPSF

public void **setPSF** (*PSF psf*)

Change the emitter's PSF model.

Parameters

- **psf** – The PSF model used to create the image of this emitter.

simulateBrightness

protected abstract double **simulateBrightness** ()

Simulates the state evolution of the emitter for the next frame, and returns the integrated brightness of this emitter for this frame.

Returns brightness of emitter in this frame [photons emitted]

6.13.2 Pixel

public class **Pixel**

Representation of a single pixel signature caused by a single emitter.

Author Marcel Stefko

Fields

x

public final int **x**
X-position of pixel in image.

y

public final int **y**
Y-position of pixel in image.

Constructors

Pixel

public **Pixel** (int *x*, int *y*, double *signature*)
Initialize new pixel with position and signature.

Parameters

- **x** – x-position [px]
- **y** – y-position [px]
- **signature** – relative brightness of this pixel due to emitter [-]

Methods

distance_to

public double **distance_to** (*Pixel p*)
Calculates euclidean distance to another pixel

Parameters

- **p** – another Pixel

Returns euclidean distance between these pixels [px]

distance_to_sq

public int **distance_to_sq** (*Pixel p*)
Calculates squared distance to another pixel

Parameters

- **p** – another Pixel

Returns squared distance between these pixels [px2]

getSignature

public double **getSignature**()
Returns this pixel's signature

Returns relative brightness of this pixel due to an emitter [-]

setSignature

public void **setSignature**(double *signature*)
Set's the pixel's signature.

6.14 ch.epfl.leb.sass.models.fluorophores

6.14.1 Fluorophore

public interface **Fluorophore** extends *Model*, *Observable*
A single fluorophore including its position and photophysical properties.

Author Kyle M. Douglass

Methods

applyTo

public void **applyTo**(float[][] *pixels*)
Renders the fluorophore onto an array of pixels.

Parameters

- **pixels** – Image on which the fluorophore's signature will be drawn.

getIlluminationListener

public *Listener* **getIlluminationListener**()
Returns the listener that listens for changes in the illumination.

Returns The fluorophore's illumination listener.

getX

public double **getX**()
Return the x-position of the fluorophore.

Returns The fluorophore's x-position.

getY

public double **getY**()

Return the y-position of the fluorophore.

Returns The fluorophore's y-position.

getZ

public double **getZ**()

Return the z-position of the fluorophore.

Returns The fluorophore's z-position.

isBleached

public boolean **isBleached**()

Has the fluorophore been bleached? If so, it can never return to a fluorescence-emitting state.

Returns A true/false value describing whether the fluorophore is bleached.

isOn

public boolean **isOn**()

Describes whether the fluorophore is emitting light or is in a dark state.

Returns A true/false value describing whether the fluorophore is emitting.

recalculateLifetimes

public void **recalculateLifetimes**(double *laserPower*)

This method recalculates the lifetimes of the fluorophore's state system based on the laser power.

Parameters

- **laserPower** – The new power of the laser.

6.15 ch.epfl.leb.sass.models.fluorophores.commands

6.15.1 FluorophoreCommand

public interface **FluorophoreCommand**

Executes a command for generating fluorophores. A fluorophore command is a tool for generating a new set of Fluorophore instances. It handles the job of both creating the instances and connecting the correct Listeners to their Observables.

Author Kyle M. Douglass

Methods

generateFluorophores

public List<*Fluorophore*> **generateFluorophores**()

Creates a new set of Fluorophore instances.

Returns A new list of Fluorophore instances.

6.15.2 FluorophoreCommandBuilder

public interface **FluorophoreCommandBuilder**

Interface for populating the field with fluorophores.

Author Kyle M. Douglass

Methods

build

public *FluorophoreCommand* **build**()

Creates a new Command instance for generating a fluorophore distribution.

Returns The new instance of a FluorophoreCommand.

camera

public *FluorophoreCommandBuilder* **camera**(*Camera camera*)

Sets the builder's Camera instance.

Parameters

- **camera** – The camera used for building Fluorophore distributions.

Returns A new copy of the builder.

fluorDynamics

public *FluorophoreCommandBuilder* **fluorDynamics**(*FluorophoreDynamics fluorDynamics*)

Sets the FluorescenceDynamics.

Parameters

- **fluorDynamics** – The fluorescence dynamical system.

Returns A new copy of the builder.

illumination

public *FluorophoreCommandBuilder* **illumination**(*Illumination illumination*)

Sets the illumination profile on the sample.

Parameters

- **illumination** – The illumination profile.

Returns A new copy of the builder.

psfBuilder

public *FluorophoreCommandBuilder* **psfBuilder** (*PSFBuilder psfBuilder*)

Sets the PSF builder that will create the fluorophores' PSFs.

Parameters

- **psfBuilder** – The PSF builder.

Returns A new copy of the builder.

6.16 ch.epfl.leb.sass.models.fluorophores.commands.internal

6.16.1 FluorophoreReceiver

public class **FluorophoreReceiver**

Populates a field of view with fluorophores. The FluorophoreGenerator contains a number of methods for creating actual fluorophore instances and in different arrangements, such as placing them on a grid, randomly distributing them in the FOV, and placing them according to input from a text file.

Author Marcel Stefko, Kyle M. Douglass

Methods

generateFluorophoresFromCSV

public static ArrayList<*Fluorophore*> **generateFluorophoresFromCSV** (File *file*, *Camera camera*, *Illumination illumination*, *PSFBuilder psfBuilder*, *FluorophoreDynamics fluorDynamics*, boolean *rescale*)

Parse a CSV file and generate fluorophores from it.

Parameters

- **file** – The CSV file. If this is null, then a dialog is opened.
- **camera** – The camera for determining the size of the field of view.
- **illumination** – The illumination profile on the sample.
- **psfBuilder** – Builder for calculating microscope PSFs.
- **fluorDynamics** – The fluorophore dynamics properties.
- **rescale** – if true, positions are rescaled to fit into frame, otherwise positions outside of frame are cropped

Throws

- **IOException** –
- **FileNotFoundException** –

Returns list of fluorophores.

generateFluorophoresGrid2D

public static ArrayList<*Fluorophore*> **generateFluorophoresGrid2D**(int *spacing*, *Camera camera*, *Illumination illumination*, *PSFBuilder psfBuilder*, *FluorophoreDynamics fluorDynamics*)

Generate a rectangular grid of fluorophores.

Parameters

- **spacing** – The distance along the grid between nearest neighbors.
- **camera** – The camera for determining the size of the field of view.
- **illumination** – The illumination profile on the sample.
- **psfBuilder** – Builder for calculating microscope PSFs.
- **fluorDynamics** – The fluorophore dynamics properties.

Returns The list of fluorophores.

generateFluorophoresGrid3D

public static ArrayList<*Fluorophore*> **generateFluorophoresGrid3D**(int *spacing*, double *zLow*, double *zHigh*, *Camera camera*, *Illumination illumination*, *PSFBuilder psfBuilder*, *FluorophoreDynamics fluorDynamics*)

Create fluorophores on a 2D grid and step-wise in the axial direction.

Parameters

- **spacing** – The distance along the grid between nearest neighbors.
- **zLow** – The lower bound on the range in z in units of pixels.
- **zHigh** – The upper bound on the range in z in units of pixels.
- **camera** – The camera for determining the size of the field of view.
- **illumination** – The illumination profile on the sample.
- **psfBuilder** – Builder for calculating microscope PSFs.
- **fluorDynamics** – The fluorophore dynamics properties.

Returns The list of fluorophores.

generateFluorophoresRandom2D

public static ArrayList<*Fluorophore*> **generateFluorophoresRandom2D**(int *numFluors*, *Camera camera*, *Illumination illumination*, *PSFBuilder psfBuilder*, *FluorophoreDynamics fluorDynamics*)

Randomly populate the field of view with fluorophores.

Parameters

- **numFluors** – The number of fluorophores to add to the field of view.
- **camera** – The camera for determining the size of the field of view.
- **illumination** – The illumination profile on the sample.
- **psfBuilder** – Builder for calculating microscope PSFs.
- **fluorDynamics** – The fluorophore dynamics properties.

Returns The list of fluorophores.

generateFluorophoresRandom3D

public static ArrayList<*Fluorophore*> **generateFluorophoresRandom3D**(int *numFluors*, double *zLow*, double *zHigh*, *Camera camera*, *Illumination illumination*, *PSFBuilder psfBuilder*, *FluorophoreDynamics fluorDynamics*)

Randomly populate the field of view with fluorophores in three dimensions.

Parameters

- **numFluors** – The number of fluorophores to add to the field of view.
- **zLow** – The lower bound on the range in z in units of pixels
- **zHigh** – The upper bound on the range in z in units of pixels
- **camera** – The camera for determining the size of the field of view.
- **illumination** – The illumination profile on the sample.
- **psfBuilder** – Builder for calculating microscope PSFs.
- **fluorDynamics** – The fluorophore dynamics properties.

Returns The list of fluorophores.

6.16.2 FluorophoreReceiverIT

public class **FluorophoreReceiverIT**

Integration tests for the FluorophoreReceiver class.

Author Kyle M. Douglass

Methods

setUp

public void **setUp** ( )

testGenerateFluorophoresFromCSV

public void **testGenerateFluorophoresFromCSV** ( )
Test of generateFluorophoresFromCSV method, of class FluorophoreReceiver.

testGenerateFluorophoresGrid2D

public void **testGenerateFluorophoresGrid2D** ( )
Test of generateFluorophoresGrid2D method, of class FluorophoreReceiver.

testGenerateFluorophoresGrid3D

public void **testGenerateFluorophoresGrid3D** ( )
Test of generateFluorophoresGrid3D method, of class FluorophoreReceiver.

testGenerateFluorophoresRandom2D

public void **testGenerateFluorophoresRandom2D** ( )
Test of generateFluorophoresRandom2D method, of class FluorophoreReceiver.

testGenerateFluorophoresRandom3D

public void **testGenerateFluorophoresRandom3D** ( )
Test of generateFluorophoresRandom3D method, of class FluorophoreReceiver.

6.16.3 GenerateFluorophoresFromCSV

public final class **GenerateFluorophoresFromCSV** implements *FluorophoreCommand*
This serves as the Invoker of a DefaultFluorophore command.

Author Kyle M.Douglass

Methods

generateFluorophores

public List<*Fluorophore*> **generateFluorophores** ( )
Executes the command that generates the fluorophores.

Returns The list of Fluorophores.

6.16.4 GenerateFluorophoresFromCSV.Builder

public static class **Builder** implements *FluorophoreCommandBuilder*
A builder for creating this command for fluorophore generation.

Methods

build

public *FluorophoreCommand* **build**()

camera

public *Builder* **camera**(*Camera* *camera*)

file

public *Builder* **file**(File *file*)

fluorDynamics

public *Builder* **fluorDynamics**(*FluorophoreDynamics* *fluorDynamics*)

illumination

public *Builder* **illumination**(*Illumination* *illumination*)

psfBuilder

public *Builder* **psfBuilder**(*PSFBuilder* *psfBuilder*)

rescale

public *Builder* **rescale**(boolean *rescale*)

6.16.5 GenerateFluorophoresGrid2D

public final class **GenerateFluorophoresGrid2D** implements *FluorophoreCommand*
This serves as the Invoker of a DefaultFluorophore command.

Author Kyle M.Douglass

Methods

generateFluorophores

public List<*Fluorophore*> **generateFluorophores**()
Executes the command that generates the fluorophores.

Returns The list of fluorophores.

6.16.6 GenerateFluorophoresGrid2D.Builder

public static class **Builder** implements *FluorophoreCommandBuilder*
A builder for creating this command for fluorophore generation.

Methods

build

public *FluorophoreCommand* **build**()

camera

public *Builder* **camera**(*Camera camera*)

fluorDynamics

public *Builder* **fluorDynamics**(*FluorophoreDynamics fluorDynamics*)

illumination

public *Builder* **illumination**(*Illumination illumination*)

psfBuilder

public *Builder* **psfBuilder**(*PSFBuilder psfBuilder*)

spacing

public *Builder* **spacing**(int *spacing*)

6.16.7 GenerateFluorophoresGrid3D

public final class **GenerateFluorophoresGrid3D** implements *FluorophoreCommand*
This serves as the Invoker of a DefaultFluorophore command.

Author Kyle M.Douglass

Methods

generateFluorophores

public List<*Fluorophore*> **generateFluorophores**()
Executes the command that generates the fluorophores.

Returns The list of Fluorophores.

6.16.8 GenerateFluorophoresGrid3D.Builder

public static class **Builder** implements *FluorophoreCommandBuilder*
A builder for creating this command for fluorophore generation.

Methods

build

public *FluorophoreCommand* **build**()

camera

public *Builder* **camera**(*Camera camera*)

fluorDynamics

public *Builder* **fluorDynamics**(*FluorophoreDynamics fluorDynamics*)

illumination

public *Builder* **illumination**(*Illumination illumination*)

psfBuilder

public *Builder* **psfBuilder**(*PSFBuilder psfBuilder*)

spacing

public *Builder* **spacing**(int *spacing*)

zHigh

public *Builder* **zHigh**(double *zHigh*)

zLow

public *Builder* **zLow** (double *zLow*)

6.16.9 GenerateFluorophoresRandom2D

public final class **GenerateFluorophoresRandom2D** implements *FluorophoreCommand*

This serves as the Invoker of a DefaultFluorophore command.

Author Kyle M.Douglass

Methods

generateFluorophores

public List<*Fluorophore*> **generateFluorophores** ()

Executes the command that generates the fluorophores.

Returns The list of fluorophores.

6.16.10 GenerateFluorophoresRandom2D.Builder

public static class **Builder** implements *FluorophoreCommandBuilder*

A builder for creating this command for fluorophore generation.

Methods

build

public *FluorophoreCommand* **build** ()

camera

public *Builder* **camera** (*Camera* *camera*)

fluorDynamics

public *Builder* **fluorDynamics** (*FluorophoreDynamics* *fluorDynamics*)

illumination

public *Builder* **illumination** (*Illumination* *illumination*)

numFluors

public *Builder* **numFluors** (int *numFluors*)

psfBuilder

public *Builder* **psfBuilder** (*PSFBuilder psfBuilder*)

6.16.11 GenerateFluorophoresRandom3D

public final class **GenerateFluorophoresRandom3D** implements *FluorophoreCommand*

This serves as the Invoker of a DefaultFluorophore command.

Author Kyle M.Douglass

Methods

generateFluorophores

public List<*Fluorophore*> **generateFluorophores** ()

Executes the command that generates the fluorophores.

Returns The list of Fluorophores.

6.16.12 GenerateFluorophoresRandom3D.Builder

public static class **Builder** implements *FluorophoreCommandBuilder*

A builder for creating this command for fluorophore generation.

Methods

build

public *FluorophoreCommand* **build** ()

camera

public *Builder* **camera** (*Camera camera*)

fluorDynamics

public *Builder* **fluorDynamics** (*FluorophoreDynamics fluorDynamics*)

illumination

public *Builder* **illumination** (*Illumination illumination*)

numFluors

public *Builder* **numFluors** (int *numFluors*)

psfBuilder

public *Builder* **psfBuilder** (*PSFBuilder psfBuilder*)

zHigh

public *Builder* **zHigh** (double *zHigh*)

zLow

public *Builder* **zLow** (double *zLow*)

6.17 ch.epfl.leb.sass.models.fluorophores.internal

6.17.1 DefaultFluorophore

public class **DefaultFluorophore** extends *AbstractEmitter* implements *Fluorophore*

A general fluorescent molecule which emits light. This class directly implements the methods of Observables, rather than extending AbstractObservable, because Java does not support multiple inheritance.

Author Marcel Stefko, Kyle M. Douglass

Constructors

DefaultFluorophore

public **DefaultFluorophore** (*PSFBuilder psfBuilder*, *Illumination illumination*, double *signal*, *StateSystem state_system*, int *start_state*, double *x*, double *y*, double *z*)

Initialize fluorophore and calculate its pattern on camera

Parameters

- **psfBuilder** – The Builder for calculating microscope PSFs.
- **illumination** – The illumination profile on the sample.
- **signal** – Number of photons per frame.
- **state_system** – Internal state system for this fluorophore
- **start_state** – Initial state number
- **x** – x-position in pixels
- **y** – y-position in pixels
- **z** – z-position in pixels

Methods

addListener

public void **addListener** (*Listener listener*)
Adds a new listener to the list of subscribed listeners.

deleteListener

public void **deleteListener** (*Listener listener*)
Deletes a listener from the list of subscribed listeners.

getCurrentState

public int **getCurrentState** ()
Returns the id of the fluorophore state system's current state.

Returns The id of the current state of the fluorophore's state system.

getIlluminationListener

public *Listener* **getIlluminationListener** ()
Returns the Listener that is attached to the illumination profile.

Returns The illumination Listener.

getOnTimeThisFrame

public double **getOnTimeThisFrame** ()
Returns the time spent in the emitting state during the previous frame. This time is the proportion of the frame's duration; 1 corresponds to having spent the entirety of the frame in the emitting state.

Returns The time spent in the emitting state.

getPhotonsThisFrame

public double **getPhotonsThisFrame** ()
Returns the number of photons emitted during the previous frame.

Returns The number of photons emitted during the previous frame.

getSignal

public double **getSignal** ()
Returns the fluorophore's number of photons per frame.

Returns The number of photons per frame emitted by the fluorophore.

getX

public double **getX**()

Return the x-position of the fluorophore.

Returns The fluorophore's x-position.

getY

public double **getY**()

Return the y-position of the fluorophore.

Returns The fluorophore's y-position.

getZ

public double **getZ**()

Return the z-position of the fluorophore.

Returns The fluorophore's z-position.

isBleached

public boolean **isBleached**()

Informs if this emitter switched into the irreversible bleached state.

Returns boolean, true if emitter is bleached

isOn

public boolean **isOn**()

Returns the current state of the emitter (on or off), but does not inform if this emitter is also bleached!

Returns true-emitter is on, false-emitter is off

nextExponential

protected final double **nextExponential**(double *mean*)

Sample an random number from an exponential distribution

Parameters

- **mean** – mean of the distribution

Returns random number from this distribution

notifyListeners

public void **notifyListeners**()

Notifies all subscribed listeners to a change in the Observable's state. This method should only be called if setChanged() has been called.

notifyListeners

public void **notifyListeners** (Object *data*)
Notifies all subscribed listeners of a state change and pushes the data.

Parameters

- **data** – The data object to push to the listeners.

recalculateLifetimes

public void **recalculateLifetimes** (double *laserPower*)
Recalculates the lifetimes of this emitter based on current laser power.

Parameters

- **laserPower** – current laser power

setChanged

public void **setChanged** ()
Indicates that the state of this Observable has been changed.

simulateBrightness

protected double **simulateBrightness** ()

toJson

public JsonElement **toJson** ()
Returns the fluorophore's properties as a JSON string.

Returns The properties of the fluorophore as a JSON string.

6.17.2 DefaultFluorophore.IlluminationListener

class **IlluminationListener** implements *Listener*
Listens to the irradiance profile and changes the fluorophore's state accordingly.

Methods

update

public void **update** (Object *data*)
This method is called by an Illumination profile when its state has changed.

Parameters

- **data** – The data object that is passed from the Observable, or null.

6.17.3 DefaultFluorophoreSerializer

class **DefaultFluorophoreSerializer** implements JsonSerializer<*DefaultFluorophore*>

Methods

serialize

public JsonElement **serialize** (*DefaultFluorophore src*, Type *typeOfSrc*, JsonSerializationContext *context*)

6.17.4 DefaultFluorophoreTest

public class **DefaultFluorophoreTest**

Author Kyle M. Douglass

Constructors

DefaultFluorophoreTest

public **DefaultFluorophoreTest** ()

Methods

setUp

public void **setUp** ()

testAddDeleteListeners

public void **testAddDeleteListeners** ()
Test of addListener and deleteListener methods, of class DefaultFluorophoreTest.

testDefaultFluorophoreToJson

public void **testDefaultFluorophoreToJson** ()
Test that the fluorophore serializes itself to JSON correctly.

testNotifyListeners

public void **testNotifyListeners** ()
Test of notifyListeners method, of class DefaultFluorophoreTest.

testNotifyListenersArg

public void **testNotifyListenersArg** ()
Test of notifyListeners method, of class DefaultFluorophoreTest.

6.17.5 DefaultFluorophoreTest.TestListener

class **TestListener** implements *Listener*

A test class that implements basic Listener capabilities.

Fields

currentState

public int **currentState**

id

public int **id**

isNull

public boolean **isNull**

nextState

public int **nextState**

timeElapsed

public double **timeElapsed**

Methods

update

public void **update** (Object *data*)

6.17.6 PhysicalFluorophore

public class **PhysicalFluorophore** extends *AbstractEmitter* implements *Fluorophore*

A general fluorescent molecule which emits light. This class directly implements the methods of Observables, rather than extending AbstractObservable, because Java does not support multiple inheritance. TODO: IMPLEMENT TESTS FOR THIS CLASS.

Author Marcel Stefko, Kyle M. Douglass

Constructors

PhysicalFluorophore

public **PhysicalFluorophore**(*PSFBuilder psfBuilder*, *Illumination illumination*, double *quantumYield*, double *extinctionCoefficient*, double *secondsPerFrame*, *StateSystem stateSystem*, int *startState*, double *x*, double *y*, double *z*)

Initialize fluorophore and calculate its pattern on camera

Parameters

- **psfBuilder** – The Builder for calculating microscope PSFs.
- **illumination** – The illumination at the fluorophore.
- **quantumYield** – The fluorophore's quantumYield.
- **extinctionCoefficient** – The fluorophore's extinction coefficient.
- **secondsPerFrame** – The length of a frame's exposure time in seconds.
- **stateSystem** – Internal state system for this fluorophore
- **startState** – Initial state number
- **x** – x-position in pixels
- **y** – y-position in pixels
- **z** – z-position in pixels

Methods

addListener

public void **addListener**(*Listener listener*)

Adds a new listener to the list of subscribed listeners.

deleteListener

public void **deleteListener**(*Listener listener*)

Deletes a listener from the list of subscribed listeners.

getCurrentState

public int **getCurrentState**()

Returns the id of the fluorophore state system's current state.

Returns The id of the current state of the fluorophore's state system.

getExtinctionCoefficient

public double **getExtinctionCoefficient**()

Returns the fluorophore extinction coefficient.

Returns The fluorophore's extinction coefficient.

getIllumination

public *Illumination* **getIllumination**()
Returns the fluorophore's illumination profile.

Returns The fluorophore's illumination profile.

getIlluminationListener

public *Listener* **getIlluminationListener**()
Returns the Listener that is attached to the illumination profile.

Returns The illumination Listener.

getOnTimeThisFrame

public double **getOnTimeThisFrame**()
Returns the time spent in the emitting state during the previous frame. This time is the proportion of the frame's duration; 1 corresponds to having spent the entirety of the frame in the emitting state.

Returns The time spent in the emitting state.

getPhotonsThisFrame

public double **getPhotonsThisFrame**()
Returns the number of photons emitted during the previous frame.

Returns The number of photons emitted during the previous frame.

getQuantumYield

public double **getQuantumYield**()
Returns the fluorophore's quantum yield.

Returns The fluorophore's quantum yield.

getSecondsPerFrame

public double **getSecondsPerFrame**()
Returns the length of a camera exposure in seconds.

Returns The length of a camera exposure in seconds.

getSignal

public double **getSignal**()
Returns the fluorophore's number of photons per frame.

Returns The number of photons per frame emitted by the fluorophore.

getX

public double **getX**()

Return the x-position of the fluorophore.

Returns The fluorophore's x-position.

getY

public double **getY**()

Return the y-position of the fluorophore.

Returns The fluorophore's y-position.

getZ

public double **getZ**()

Return the z-position of the fluorophore.

Returns The fluorophore's z-position.

isBleached

public boolean **isBleached**()

Informs if this emitter switched into the irreversible bleached state.

Returns boolean, true if emitter is bleached

isOn

public boolean **isOn**()

Returns the current state of the emitter (on or off), but does not inform if this emitter is also bleached!

Returns true-emitter is on, false-emitter is off

nextExponential

protected final double **nextExponential**(double *mean*)

Sample an random number from an exponential distribution

Parameters

- **mean** – mean of the distribution

Returns random number from this distribution

notifyListeners

public void **notifyListeners**()

Notifies all subscribed listeners to a change in the Observable's state. This method should only be called if setChanged() has been called.

notifyListeners

public void **notifyListeners** (Object *data*)

Notifies all subscribed listeners of a state change and pushes the data.

Parameters

- **data** – The data object to push to the listeners.

recalculateLifetimes

public void **recalculateLifetimes** (double *laserPower*)

Recalculates the lifetimes of this emitter based on current laser power.

Parameters

- **laserPower** – current laser power

setChanged

public void **setChanged** ()

Indicates that the state of this Observable has been changed.

simulateBrightness

protected double **simulateBrightness** ()

toJson

public JsonElement **toJson** ()

Returns the fluorophore's properties as a JSON string.

Returns The properties of the fluorophore as a JSON string.

6.17.7 PhysicalFluorophore.IlluminationListener

class **IlluminationListener** implements *Listener*

Methods

update

public void **update** (Object *data*)

This method is called by an Illumination profile when its state has changed.

Parameters

- **data** – The data object that is passed from the Observable, or null.

6.17.8 PhysicalFluorophoreSerializer

class **PhysicalFluorophoreSerializer** implements JsonSerializer<*PhysicalFluorophore*>

Methods

serialize

public JsonElement **serialize**(*PhysicalFluorophore src*, Type *typeOfSrc*, JsonSerializationContext *context*)

6.18 ch.epfl.leb.sass.models.illuminations

6.18.1 ElectricField

public interface **ElectricField**

Common methods for ElectricField instances of an illumination profile.

Author Kyle M. Douglass

Methods

getEx

public Complex **getEx**(double *x*, double *y*, double *z*)

Returns the x-component of the time-independent electric field at the position (x, y, z).

Parameters

- **x** – The x-position within the sample.
- **y** – The y-position within the sample.
- **z** – The z-position within the sample.

Returns The x-component of the electric field at the position (x, y, z).

getEy

public Complex **getEy**(double *x*, double *y*, double *z*)

Returns the y-component of the time-independent electric field at the position (x, y, z).

Parameters

- **x** – The x-position within the sample.
- **y** – The y-position within the sample.
- **z** – The z-position within the sample.

Returns The y-component of the electric field at the position (x, y, z).

getEz

public Complex **getEz** (double *x*, double *y*, double *z*)

Returns the z-component of the time-independent electric field at the position (x, y, z).

Parameters

- **x** – The x-position within the sample.
- **y** – The y-position within the sample.
- **z** – The z-position within the sample.

Returns The z-component of the electric field at the position (x, y, z).

getRefractiveIndex

public *RefractiveIndex* **getRefractiveIndex** ()

Returns the sample's refractive index that produced this field.

Returns The refractive index distribution of the sample.

getWavelength

public double **getWavelength** ()

Returns the radiation's wavelength.

Returns The wavelength of the radiation.

6.18.2 ElectricFieldBuilder

public interface **ElectricFieldBuilder**

Builds a new ElectricField instance.

Author Kyle M. Douglass

Methods

build

public *ElectricField* **build** ()

Builds a new ElectricField instance.

Returns A new instance of an electric field.

refractiveIndex

public *ElectricFieldBuilder* **refractiveIndex** (*RefractiveIndex n*)

Sets the refractive index of the sample.

Parameters

- **n** – The sample's refractive index.

Returns A new ElectricFieldBuilder with the refractive index set.

wavelength

public *ElectricFieldBuilder* **wavelength**(double *wavelength*)

Sets the free space wavelength of the radiation.

Parameters

- **wavelength** – The free space wavelength.

Returns A new ElectricFieldBuilder with the wavelength set.

6.18.3 Illumination

public interface **Illumination** extends *Listener*, *Observable*

Common methods for the microscope's illumination profile.

Author Kyle M. Douglass

Methods

getElectricField

public *ElectricField* **getElectricField**()

Retrieves the complex electric field.

Returns The complex electric field.

getIrradiance

public double **getIrradiance**(double *x*, double *y*, double *z*)

Returns the illumination irradiance at the point (x, y, z).

Parameters

- **x** – The x-position within the sample.
- **y** – The y-position within the sample.
- **z** – The z-position within the sample.

Returns The irradiance at the point (x, y, z).

getPower

public double **getPower**()

Returns the power carried by the illumination profile. This quantity is the irradiance integrated over the illuminated area and is controlled by the laser power.

Returns The power carried by the illumination profile.

See also: `.setPower(double)`

setPower

public void **setPower** (double *power*)
Sets the power carried by the illumination.

Parameters

- **power** – The power carried by the illumination.

6.18.4 IlluminationBuilder

public interface **IlluminationBuilder**
Builds a new Illumination instance.

Author Kyle M. Douglass

Methods

build

public *Illumination* **build** ()
Creates a new Illumination instance from the builder's properties.

Returns A new Illumination instance.

power

public *IlluminationBuilder* **power** (double *power*)
Sets the power delivered by the illumination.

Parameters

- **power** – The illumination's power (energy per time).

Returns A new IlluminationBuilder with the power set.

refractiveIndex

public *IlluminationBuilder* **refractiveIndex** (*RefractiveIndex* *refractiveIndex*)
Sets the refractive index of the sample.

Parameters

- **refractiveIndex** – The sample's refractive index.

Returns A new IlluminationBuilder with the refractive index set.

wavelength

public *IlluminationBuilder* **wavelength** (double *wavelength*)
Sets the wavelength of the illumination.

Parameters

- **wavelength** – The illumination's wavelength.

Returns A new IlluminationBuilder with the refractive wavelength set.

6.19 ch.epfl.leb.sass.models.illuminations.commands

6.19.1 ElectricFieldCommand

public interface **`ElectricFieldCommand`**

Executes a command that generates a new ElectricField instance.

Author Kyle M. Douglass

Methods

generateElectricField

public *ElectricField* **`generateElectricField`**()

Creates a new ElectricField instance within the sample.

Returns A new ElectricField instance.

6.19.2 ElectricFieldCommandBuilder

public interface **`ElectricFieldCommandBuilder`**

Common methods for commands that build ElectricField generators. A command is like a method wrapped inside an object.

Author Kyle M. Douglass

Methods

build

public *ElectricFieldCommand* **`build`**()

Creates a new command for generating an ElectricField instance.

Returns A command for generating new ElectricField instances.

refractiveIndex

public *ElectricFieldCommandBuilder* **`refractiveIndex`**(*RefractiveIndex refractiveIndex*)

Assigns a value to the refractive index field of the command builder.

Parameters

- **`refractiveIndex`** –

Returns A copy of this builder with the new refractive index.

wavelength

public *ElectricFieldCommandBuilder* **wavelength** (double *wavelength*)

Assigns a value to the wavelength field of the command builder.

Parameters

- **wavelength** –

Returns A copy of this builder with the new wavelength.

6.20 ch.epfl.leb.sass.models.illuminations.commands.internal

6.20.1 ElectricFieldReceiver

public class **ElectricFieldReceiver**

Methods for creating new and different types of ElectricField instances.

Author Kyle M. Douglass

Methods

generateUniformSquareElectricField

public static *ElectricField* **generateUniformSquareElectricField** (double *width*, double *height*, Vector3D *orientation*, double *wavelength*, *RefractiveIndex* *refractiveIndex*)

Creates a uniform electric field of square shape. This electric field is uniformly polarized and of the same magnitude within the area (x, x+width), (y, y+height). It extends from z = 0 to z = infinity. Obviously, it is not physical but is a good approximation of a plane wave with finite energy.

Parameters

- **width** – The extent of the field in x from 0 to width.
- **height** – The extend of the field in y from 0 to height.
- **orientation** – The orientation of the electric field vector.
- **wavelength** – The wavelength of the radiation.
- **refractiveIndex** – The sample's refractive index distribution.

Returns A new instance of an ElectricField that is uniform and square.

6.20.2 GenerateSquareUniformElectricField

public class **GenerateSquareUniformElectricField** implements *ElectricFieldCommand*

Creates a uniform electric field of square shape. This electric field is uniformly polarized and of the same magnitude within the area (0, width), (0, height). It extends from z = 0 to z = infinity. Obviously, it is not physical but is a good approximation of a plane wave with finite energy.

Author Kyle M. Douglass

Fields

height

double **height**
The extend of the illumination from y = 0 to y = height.

orientation

Vector3D **orientation**
The orientation of the electric field vector. This vector should be normalized to 1.

refractiveIndex

RefractiveIndex **refractiveIndex**
The refractive index of the medium.

wavelength

double **wavelength**
The wavelength of the radiation in free space.

width

double **width**
The extend of the illumination from x = 0 to x = width.

Methods

generateElectricField

public *ElectricField* **generateElectricField**()
Executes the command that returns a uniform electric field of square shape. This electric field is uniformly polarized and of the same magnitude within the area (0, width), (0, height). It extends from z = 0 to z = infinity. Obviously, it is not physical but is a good approximation of a plane wave with finite energy.

6.20.3 GenerateSquareUniformElectricField.Builder

public static class **Builder** implements *ElectricFieldCommandBuilder*

Methods

build

public *ElectricFieldCommand* **build**()
Creates a new command for generating a SquareUniformElectricField.

Returns A new instance of this command.

height

public *Builder* **height** (double *height*)

orientation

public *Builder* **orientation** (Vector3D *orientation*)

refractiveIndex

public *Builder* **refractiveIndex** (*RefractiveIndex* *refractiveIndex*)

wavelength

public *Builder* **wavelength** (double *wavelength*)

width

public *Builder* **width** (double *width*)

6.20.4 GenerateSquareUniformElectricFieldIT

public class **GenerateSquareUniformElectricFieldIT**

Integration tests for GenerateSquareUniformElectricField and the ElectricFieldReceiver.

Author Kyle M. Douglass

Methods

setUp

public void **setUp** ()

testGenerateElectricField

public void **testGenerateElectricField** ()

Test of generateElectricField method, of class GenerateSquareUniformElectricField.

6.21 ch.epfl.leb.sass.models.illuminations.internal

6.21.1 SquareUniformElectricField

public class **SquareUniformElectricField** implements *ElectricField*

Creates a uniform electric field of square shape propagating in the +z-direction. This electric field is uniformly polarized and of the same magnitude within the area (0, width), (0, height). It extends from z = 0 to z = infinity.

Obviously, it is not physical but is a good approximation of a plane wave with finite energy. The field propagates in the z-direction.

Author Kyle M. Douglass

Methods

getEx

public Complex **getEx**(double *x*, double *y*, double *z*)

Returns the x-component of the time-independent electric field at the position (x, y, z).

Parameters

- **x** – The x-position within the sample.
- **y** – The y-position within the sample.
- **z** – The z-position within the sample.

Returns The x-component of the electric field at the position (x, y, z).

getEy

public Complex **getEy**(double *x*, double *y*, double *z*)

Returns the y-component of the time-independent electric field at the position (x, y, z).

Parameters

- **x** – The x-position within the sample.
- **y** – The y-position within the sample.
- **z** – The z-position within the sample.

Returns The y-component of the electric field at the position (x, y, z).

getEz

public Complex **getEz**(double *x*, double *y*, double *z*)

Returns the z-component of the time-independent electric field at the position (x, y, z).

Parameters

- **x** – The x-position within the sample.
- **y** – The y-position within the sample.
- **z** – The z-position within the sample.

Returns The z-component of the electric field at the position (x, y, z).

getRefractiveIndex

public *RefractiveIndex* **getRefractiveIndex**()

Returns the sample's refractive index that produced this field.

Returns The refractive index distribution of the sample.

getWavelength

public double **getWavelength**()
Returns the radiation's wavelength.

Returns The wavelength of the radiation.

6.21.2 SquareUniformElectricField.Builder

public static class **Builder** implements *ElectricFieldBuilder*
The builder for constructing Gaussian2D instances.

Methods

build

public *SquareUniformElectricField* **build**()

height

public *Builder* **height**(double *height*)

orientation

public *Builder* **orientation**(Vector3D *orientation*)

refractiveIndex

public *Builder* **refractiveIndex**(*RefractiveIndex n*)

wavelength

public *Builder* **wavelength**(double *wavelength*)

width

public *Builder* **width**(double *width*)

6.21.3 SquareUniformElectricFieldTest

public class **SquareUniformElectricFieldTest**
Unit tests for the SquareUniformElectricField.

Author Kyle M. Douglass

Methods

setUp

public void **setUp**()

testGetEx

public void **testGetEx**()
Test of getEx method, of class SquareUniformElectricField.

testGetExAbsorption

public void **testGetExAbsorption**()
Test of getEx method, of class SquareUniformElectricField.

testGetEy

public void **testGetEy**()
Test of getEy method, of class SquareUniformElectricField.

testGetEz

public void **testGetEz**()
Test of getEz method, of class SquareUniformElectricField.

testGetRefractiveIndex

public void **testGetRefractiveIndex**()
Test of getRefractiveIndexMethod, of class SquareUniformElectricField.

testGetWavelength

public void **testGetWavelength**()
Test of getWavelength, of class SquareUniformElectricField.

6.21.4 SquareUniformIllumination

public class **SquareUniformIllumination** extends *AbstractObservable* implements *Illumination*
Implements a linearly polarized uniform illumination profile of square shape.

Author Kyle M. Douglass

Methods

getElectricField

public *ElectricField* **getElectricField**()
Retrieves the complex electric field.

Returns The complex electric field.

getIrradiance

public double **getIrradiance**(double *x*, double *y*, double *z*)
Returns the irradiance in the sample at the point (x, y, z).

Parameters

- **x** – The x-position in the sample.
- **y** – The y-position in the sample.
- **z** – The z-position in the sample.

getPower

public double **getPower**()
Returns the power carried by the illumination profile. This quantity is the irradiance integrated over the illuminated area and is controlled by the laser power.

Returns The power carried by the illumination profile.

See also: `.setPower(double)`

setPower

public void **setPower**(double *power*)
Sets the power carried by the illumination.

Parameters

- **power** – The power carried by the illumination.

update

public void **update**(Object *data*)
This method is called by an illumination source when its state has changed.

Parameters

- **data** – The message that is passed from the illumination source.

6.21.5 SquareUniformIllumination.Builder

public static class **Builder** implements *IlluminationBuilder*
A Builder for creating new SquareUniformIllumination instances.

Methods

build

public *SquareUniformIllumination* **build**()

height

public *Builder* **height**(double *height*)

orientation

public *Builder* **orientation**(Vector3D *orientation*)

power

public *Builder* **power**(double *power*)

refractiveIndex

public *Builder* **refractiveIndex**(*RefractiveIndex* *refractiveIndex*)

wavelength

public *Builder* **wavelength**(double *wavelength*)

width

public *Builder* **width**(double *width*)

6.21.6 SquareUniformIlluminationIT

public class **SquareUniformIlluminationIT**

IntegrationTests for the SquareUniformIllumination class.

Author Kyle M. Douglass

Methods

setUp

public void **setUp**()

testGetElectricField

public void **testGetElectricField**()
Test of getElectricField method, of class SquareUniformIllumination.

testGetIrradiance

public void **testGetIrradiance**()
Test of getIrradiance method, of class SquareUniformIllumination.

testGetPower

public void **testGetPower**()
Test of getPower method, of class SquareUniformIllumination.

testSetPower

public void **testSetPower**()
Test of setPower method, of class SquareUniformIllumination.

testUpdate

public void **testUpdate**()
Test of update method, of class SquareUniformIllumination.

testUpdateWrongMessageType

public void **testUpdateWrongMessageType**()
Test of update method, of class SquareUniformIllumination.

6.22 ch.epfl.leb.sass.models.obstructors

6.22.1 Obstructor

public interface **Obstructor** extends Serializable
This object is a constant obstruction of the field of view (for example gold bead, foreign object in field of view, dirt, etc.)

Author Marcel Stefko

Methods

applyTo

public void **applyTo**(float[][] *pixels*)
Draws the obstruction onto the given float array representing an image.

Parameters

- **pixels** – image to be drawn on

6.23 ch.epfl.leb.sass.models.obstructors.internal

6.23.1 Fiducial

public class **Fiducial** extends *AbstractEmitter* implements *Obstructor*

Constructors

Fiducial

public **Fiducial** (*PSFBuilder psfBuilder*, double *brightness*, double *x*, double *y*, double *z*)

Methods

simulateBrightness

protected double **simulateBrightness** ()

6.24 ch.epfl.leb.sass.models.obstructors.internal.commands

6.24.1 GenerateFiducialsRandom2D

public final class **GenerateFiducialsRandom2D** implements *ObstructorCommand*

Author Kyle M. Douglass

Methods

generateObstructors

public List<*Obstructor*> **generateObstructors** ()

Executes the command that generates the fluorophores.

Returns The list of fluorophores.

6.24.2 GenerateFiducialsRandom2D.Builder

public static class **Builder** implements *ObstructorCommandBuilder*

A builder for creating this command for obstructor generation.

Methods

brightness

public *Builder* **brightness** (double *brightness*)

build

public *ObstructorCommand* **build** ()

camera

public *Builder* **camera** (*Camera camera*)

numFiducials

public *Builder* **numFiducials** (int *numFiducials*)

psfBuilder

public *Builder* **psfBuilder** (*PSFBuilder psfBuilder*)

stage

public *Builder* **stage** (*Stage stage*)

6.24.3 ObstructorCommand

public interface **ObstructorCommand**

Author Kyle M. Douglass

Methods

generateObstructors

public List<*Obstructor*> **generateObstructors** ()

6.24.4 ObstructorCommandBuilder

public interface **ObstructorCommandBuilder**

Interface for populating the field with obstructors, i.e. gold beads.

Author Kyle M. Douglass

Methods

brightness

public *ObstructorCommandBuilder* **brightness** (double *brightness*)

build

public *ObstructorCommand* **build** ()

camera

public *ObstructorCommandBuilder* **camera** (*Camera* *camera*)

psfBuilder

public *ObstructorCommandBuilder* **psfBuilder** (*PSFBuilder* *psfBuilder*)

stage

public *ObstructorCommandBuilder* **stage** (*Stage* *stage*)

6.24.5 ObstructorReceiver

public class **ObstructorReceiver**

Creates obstructors after receiving commands.

Author Kyle M. Douglass

Methods

generateGoldBeadsRandom2D

public static ArrayList<*Obstructor*> **generateGoldBeadsRandom2D** (int *numBeads*, double *brightness*, *Camera* *camera*, *Stage* *stage*, *PSFBuilder* *psfBuilder*)

6.25 ch.epfl.leb.sass.models.photophysics

6.25.1 FluorophoreDynamics

public abstract class **FluorophoreDynamics** implements Serializable

A fluorophore state system.

Fields

stateSystem

protected final *StateSystem* **stateSystem**
The state system describing the fluorescence dynamics.

Constructors

FluorophoreDynamics

protected **FluorophoreDynamics** (double *signal*, double *wavelength*, *StateSystem* *stateSystem*, int *startingState*, double[][][] *Mk*)
Initializes the state system with the transition rates and starting state.

Parameters

- **stateSystem** –
- **startingState** –
- **Mk** –

Methods

getMk

public double[][][] **getMk** ()

getSignal

public double **getSignal** ()

getStartingState

public int **getStartingState** ()

getStateSystem

public *StateSystem* **getStateSystem** ()

getWavelength

public double **getWavelength** ()

6.25.2 FluorophoreDynamicsBuilder

public interface **FluorophoreDynamicsBuilder**
Interface for creating fluorophore dynamics.

Methods

build

public *FluorophoreDynamics* **build**()

6.25.3 StateSystem

public class **StateSystem** implements Serializable

Class which describes a Markovian fluorophore state model. This class provides transition rates and mean lifetimes for Markovian models based on current laser illumination intensity.

Author stefko

Fields

current_laser_power

protected double **current_laser_power**

Laser power value for which the currently stored lifetime values are calculated.

Constructors

StateSystem

public **StateSystem**(int *N_states*, double[][][] *M_scaling*)

Initialize the state system.

Parameters

- **N_states** – number of states
- **M_scaling** – double[][][] matrix of dimensions N x N x A. A can be different for each position in the matrix. This matrix can be interpreted as follows: double[] P = M_scaling[i][j]; k_ij(I) = P[0] + P[1]*I + P[2]*I^2 + … P[n]*I^n; k_ij(I) is transition rate between i-th and j-th state under laser illumination intensity I. The first row of this matrix is considered the active state, the last row is considered the bleached state.

Methods

getMeanTransitionLifetime

public final double **getMeanTransitionLifetime**(int *from*, int *to*)

Parameters

- **from** – index of initial state
- **to** – index of final state

Returns mean transition lifetime from one state to another

getNStates

public int **getNStates** ( )

Returns number of states of this model

getTransitionRate

public final double **getTransitionRate** (int *from*, int *to*)

Parameters

- **from** – index of initial state
- **to** – index of final state

Returns transition rate from one state to another

isBleachedState

public boolean **isBleachedState** (int *state*)

Returns true if the state is the bleached state (the last state of the model)

Parameters

- **state** – id of current state

Returns state == (N_states - 1)

isOnState

public boolean **isOnState** (int *state*)

Returns true if the state is the active state (the 0-th state)

Parameters

- **state** – id of current state

Returns (state==0)

recalculate_lifetimes

public final void **recalculate_lifetimes** (double *laser_power*)

Recalculates each element of the transition matrix, based on the scaling matrix provided at initialization. double[] P = M_scaling[i][j]; k_ij(I) = P[0] + P[1]*I + P[2]*I^2 + … P[n]*I^n; k_ij(I) is transition rate between i-th and j-th state under laser illumination intensity I.

Parameters

- **laser_power** – illumination intensity I to recalculate for

6.26 ch.epfl.leb.sass.models.photophysics.internal

6.26.1 PalmDynamics

public class **PalmDynamics** extends *FluorophoreDynamics*

A dynamical system for modeling PALM-like fluorescence dynamics.

Author Marcel Stefko, Kyle M. Douglass

Fields

STARTINGSTATE

public static final int **STARTINGSTATE**

Fluorophores start in the dark state.

6.26.2 PalmDynamics.Builder

public static class **Builder** implements *FluorophoreDynamicsBuilder*

Builder for creating PALM dynamical systems.

Methods

build

public *PalmDynamics* **build**()

Initialize a PALM-like dynamical system for fluorescence dynamics.

Returns The PALM dynamical system.

kA

public *Builder* **kA**(double *kA*)

The activation rate

Parameters

- **kA** –

kB

public *Builder* **kB**(double *kB*)

The bleaching rate

kD1

public *Builder* **kD1**(double *kD1*)

The rate of entering the first dark state

kD2

public *Builder* **kD2** (double *kD2*)
The rate of entering the second dark state

kR1

public *Builder* **kR1** (double *kR1*)
The return rate from the first dark state

kR2

public *Builder* **kR2** (double *kR2*)
The return rate from the second dark state

signal

public *Builder* **signal** (double *signal*)
The average number of photons per fluorophore per frame

Parameters

- **signal** –

Returns PalmDynamics builder

wavelength

public *Builder* **wavelength** (double *wavelength*)
The center wavelength of the fluorescence emission

Parameters

- **wavelength** –

Returns PalmDynamics builder

6.26.3 SimpleDynamics

public class **SimpleDynamics** extends *FluorophoreDynamics*
Dynamics for a simple three-state system (emitting, non-emitting, and bleached).

Author Marcel Stefko, Kyle M. Douglass

Fields

STARTINGSTATE

public static final int **STARTINGSTATE**
Fluorophores start in the dark state.

6.26.4 SimpleDynamics.Builder

public static class **Builder** implements *FluorophoreDynamicsBuilder*

Builder for creating Simple dynamical systems.

Methods

build

public *SimpleDynamics* **build**()

Creates a Simple dynamical system.

signal

public *Builder* **signal**(double *signal*)

The average number of photons per fluorophore per frame

Parameters

- **signal** –

Returns SimpleDynamics builder

tBl

public *Builder* **tBl**(double *tBl*)

The average bleaching time

Parameters

- **tBl** –

Returns SimpleDynamics builder

tOff

public *Builder* **tOff**(double *tOff*)

The average off time

Parameters

- **tOff** –

Returns SimpleDynamics builder

tOn

public *Builder* **tOn**(double *tOn*)

The average on time

Parameters

- **tOn** –

Returns SimpleDynamics builder

wavelength

public *Builder* **wavelength** (double *wavelength*)
The center wavelength of the fluorescence emission

Parameters

- **wavelength** –

Returns SimpleDynamics builder

6.26.5 StormDynamics

public class **StormDynamics** extends *FluorophoreDynamics*
A dynamical system for modeling STORM-like fluorescence dynamics.

Author Marcel Stefko, Kyle M. Douglass

Fields

STARTINGSTATE

public static final int **STARTINGSTATE**
Fluorophores start in the dark state.

6.26.6 StormDynamics.Builder

public static class **Builder** implements *FluorophoreDynamicsBuilder*

Methods

build

public *StormDynamics* **build** ()

kBl

public *Builder* **kBl** (double *kBl*)
The bleaching rate

Returns StormDynamics Builder

kDark

public *Builder* **kDark** (double *kDark*)
The transition to the dark state

Parameters

- **kDark** –

Returns StormDynamics builder

kDarkRecovery

public *Builder* **kDarkRecovery** (double *kDarkRecovery*)

The recovery from the dark state

Parameters

- **kDarkRecovery** –

Returns StormDynamics builder

kDarkRecoveryConstant

public *Builder* **kDarkRecoveryConstant** (double *kDarkRecoveryConstant*)

The constant recovery rate from the dark state

Parameters

- **kDarkRecoveryConstant** –

Returns StormDynamics builder

kTriplet

public *Builder* **kTriplet** (double *kTriplet*)

The transition to the triplet state

Parameters

- **kTriplet** –

Returns StormDynamics builder

kTripletRecovery

public *Builder* **kTripletRecovery** (double *kTripletRecovery*)

The recovery rate from the triplet state

Parameters

- **kTripletRecovery** –

Returns StormDynamics builder

signal

public *Builder* **signal** (double *signal*)

The average number of photons per fluorophore per frame

Parameters

- **signal** –

Returns StormDynamics builder

wavelength

public *Builder* **wavelength** (double *wavelength*)

The center wavelength of the fluorescence emission

Parameters

- **wavelength** –

Returns StormDynamics builder

6.27 ch.epfl.leb.sass.models.psfs

6.27.1 PSF

public interface **PSF**

Interface that defines the behavior of a microscope point spread function.

Author Kyle M. Douglass

Methods

generatePixelSignature

public double **generatePixelSignature** (int *pixelX*, int *pixelY*)

Computes the expected value for the PSF integrated over a pixel.

Parameters

- **pixelX** – The pixel's x-position.
- **pixelY** – The pixel's y-position.

Throws

- **org.apache.commons.math.MathException** –

Returns The relative probability of a photon hitting this pixel.

generateSignature

public void **generateSignature** (ArrayList<*Pixel*> *pixels*)

Computes the digitized PSF across all pixels within the emitter's vicinity.

Parameters

- **pixels** – The list of pixels spanned by the emitter's image.

getRadius

public double **getRadius** ()

Returns the radius of the circle that fully encloses the PSF. This value is used to determine how many pixels within the vicinity of the emitter contribute to the PSF. It is necessary because many PSF models extend to infinity in one or more directions.

Returns The radius of the PSF in pixels.

6.27.2 PSFBuilder

public interface **PSFBuilder**

Defines the Builder interface for constructing PSFs. Passing Builders instances, rather than PSF instances, to the simulation allows the PSF to be constructed at different times during the simulation. For example, one might set basic parameters like the wavelength in the beginning of the simulation and set the emitter's z-position immediately before a frame is computed. This means the simulation can dynamically create new PSF instances in response to changing simulation parameters.

Author Kyle M. Douglass

Methods

FWHM

public *PSFBuilder* **FWHM**(double *FWHM*)

The Gaussian approximation's FWHM for this PSF.

NA

public *PSFBuilder* **NA**(double *NA*)

The numerical aperture of the objective.

build

public *PSF* **build**()

Builds a new instance of the PSF model.

Returns The PSF model.

eX

public *PSFBuilder* **eX**(double *eX*)

Sets the emitter's x-position.

Parameters

- **eX** – The emitter's x-position. [pixels]

eY

public *PSFBuilder* **eY**(double *eY*)

Sets the emitter's y-position.

Parameters

- **eY** – The emitter's y-position. [pixels]

eZ

public *PSFBuilder* **eZ** (double *eZ*)
Sets the emitter's z-position.

Parameters

- **eZ** – The emitter's z-position. [pixels]

resLateral

public *PSFBuilder* **resLateral** (double *resLateral*)
Object space pixel size

stageDisplacement

public *PSFBuilder* **stageDisplacement** (double *stageDisplacement*)
Sets the stage displacement for axially-dependent PSFs.

wavelength

public *PSFBuilder* **wavelength** (double *wavelength*)
Wavelength of the light.

6.28 ch.epfl.leb.sass.models.psfs.internal

6.28.1 Gaussian2D

public final class **Gaussian2D** implements *PSF*
Generates a digital representation of a two-dimensional Gaussian PSF.

Author Kyle M. Douglass

Methods

generatePixelSignature

public double **generatePixelSignature** (int *pixelX*, int *pixelY*)
Computes the relative probability of receiving a photon at the pixel. (emitterX, emitterY). The z-position of the emitter is ignored.

Parameters

- **pixelX** – The pixel's x-position.
- **pixelY** – The pixel's y-position.

Throws

- **org.apache.commons.math.MathException** –

Returns The probability of a photon hitting this pixel.

generateSignature

public void **generateSignature** (ArrayList<*Pixel*> *pixels*)

Generates the digital signature of the emitter on its nearby pixels.

Parameters

- **pixels** – The list of pixels spanned by the emitter's image.

getFWHM

public double **getFWHM** ()

getRadius

public double **getRadius** ()

Computes the half-width of the PSF for determining which pixels contribute to the emitter signal. For a 2D Gaussian, the effective width used here is three times the standard deviation.

Returns The width of the PSF.

setFWHM

public void **setFWHM** (double *fwhm*)

6.28.2 Gaussian2D.Builder

public static class **Builder** implements *PSFBuilder*

The builder for constructing Gaussian2D instances.

Methods

FWHM

public *Builder* **FWHM** (double *fwhm*)

NA

public *Builder* **NA** (double *NA*)

build

public *Gaussian2D* **build** ()

eX

public *Builder* **eX** (double *eX*)

eY

public *Builder* **eY** (double *eY*)

eZ

public *Builder* **eZ** (double *eZ*)

resLateral

public *Builder* **resLateral** (double *resLateral*)

stageDisplacement

public *Builder* **stageDisplacement** (double *stageDisplacement*)

wavelength

public *Builder* **wavelength** (double *wavelength*)

6.28.3 Gaussian2DTest

public class **Gaussian2DTest**

Author Kyle M. Douglass

Methods

setUp

public void **setUp** ()

testGeneratePixelSignature

public void **testGeneratePixelSignature** ()
Test of generatePixelSignature method, of class Gaussian2D.

testGetRadius

public void **testGetRadius** ()
Test of getRadius method, of class Gaussian2D.

testGetSignature

public void **testGetSignature** ()
Test of getSignature method, of class Gaussian2D.

6.28.4 Gaussian3D

public final class **Gaussian3D** implements *PSF*

Generates a digital representation of a three-dimensional Gaussian PSF. In this simple but unphysical model, the variance of the Gaussian PSF from an emitter at a distance z from the focal plane scales linearly with the amount of defocus.

Author Kyle M. Douglass

Methods

generatePixelSignature

public double **generatePixelSignature** (int *pixelX*, int *pixelY*)

Computes the relative probability of receiving a photon at the pixel.

Parameters

- **pixelX** – The pixel's x-position.
- **pixelY** – The pixel's y-position.

Throws

- **org.apache.commons.math.MathException** –

Returns The probability of a photon hitting this pixel.

generateSignature

public void **generateSignature** (ArrayList<*Pixel*> *pixels*)

Generates the digital signature of the emitter on its nearby pixels.

Parameters

- **pixels** – The list of pixels spanned by the emitter's image.

getFWHM

public double **getFWHM** ()

getNumericalAperture

public double **getNumericalAperture** ()

getRadius

public double **getRadius** ()

Computes the half-width of the PSF for determining which pixels contribute to the emitter signal. The effective width used here is five times the standard deviation when the emitter is exactly in focus. The larger factor of five accounts for the larger lateral PSF size when it is out of focus.

Returns The width of the PSF.

setFWHM

public void **setFWHM**(double *fwhm*)

setNumericalAperture

public void **setNumericalAperture**(double *numericalAperture*)

6.28.5 Gaussian3D.Builder

public static class **Builder** implements *PSFBuilder*
The builder for constructing Gaussian2D instances.

Methods

FWHM

public *Builder* **FWHM**(double *fwhm*)

NA

public *Builder* **NA**(double *NA*)

build

public *Gaussian3D* **build**()

eX

public *Builder* **eX**(double *eX*)

eY

public *Builder* **eY**(double *eY*)

eZ

public *Builder* **eZ**(double *eZ*)

resLateral

public *Builder* **resLateral**(double *resLateral*)

stageDisplacement

public *Builder* **stageDisplacement** (double *stageDisplacement*)

wavelength

public *Builder* **wavelength** (double *wavelength*)

6.28.6 Gaussian3DTest

public class **Gaussian3DTest**

Author douglass

Methods

setUp

public void **setUp** ()

testGeneratePixelSignatureInFocus

public void **testGeneratePixelSignatureInFocus** ()
Test of generatePixelSignature method, of class Gaussian3D.

testGeneratePixelSignatureOutOfFocus

public void **testGeneratePixelSignatureOutOfFocus** ()
Test of generatePixelSignature method, of class Gaussian3D.

testGetRadius

public void **testGetRadius** ()
Test of getRadius method, of class Gaussian2D.

testGetSignatureInFocus

public void **testGetSignatureInFocus** ()
Test of getSignature method, of class Gaussian2D.

6.28.7 GibsonLanniPSF

public final class **GibsonLanniPSF** implements *PSF*

Computes an emitter PSF based on the Gibson-Lanni model. This algorithm was first described in Li, J., Xue, F., and Blu, T. (2017). Fast and accurate three-dimensional point spread function computation for fluorescence microscopy. JOSA A, 34(6), 1029-1034. The code is adapted from MicroscPSF-ImageJ by Jizhou Li: https://github.com/hijizhou/MicroscPSF-ImageJ

Author Kyle M. Douglass

Methods

generatePixelSignature

public double **generatePixelSignature** (int *pixelX*, int *pixelY*)

Computes the relative probability of receiving a photon at pixel (pixelX, pixelY) from an emitter at (emitterX, emitterY, emitterZ).

Parameters

- **pixelX** – The pixel's x-position.
- **pixelY** – The pixel's y-position.

Returns The probability of a photon hitting this pixel.

generateSignature

public void **generateSignature** (ArrayList<*Pixel*> *pixels*)

Generates the digital signature (the PSF) of the emitter on its nearby pixels.

Parameters

- **pixels** – The list of pixels spanned by the emitter's image.

getRadius

public double **getRadius** ()

Computes the half-width of the PSF for determining which pixels contribute to the emitter signal. This number is based on the greatest horizontal or vertical extent of the grid that the PSF is computed on. If maxRadius is smaller than that determined by the PSF's computational grid, then maxRadius is returned.

Returns The width of the PSF.

6.28.8 GibsonLanniPSF.Builder

public static class **Builder** implements *PSFBuilder*

Constructors

Builder

public **Builder** ()

Methods

FWHM

public *Builder* **FWHM** (double *FWHM*)

NA

public *Builder* **NA** (double *NA*)

build

public *GibsonLanniPSF* **build** ()

eX

public *Builder* **eX** (double *eX*)

eY

public *Builder* **eY** (double *eY*)

eZ

public *Builder* **eZ** (double *eZ*)

maxRadius

public *Builder* **maxRadius** (double *maxRadius*)

ng

public *Builder* **ng** (double *ng*)

ng0

public *Builder* **ng0** (double *ng0*)

ni

public *Builder* **ni** (double *ni*)

ni0

public *Builder* **ni0** (double *ni0*)

ns

public *Builder* **ns** (double *ns*)

numBasis

public *Builder* **numBasis** (int *numBasis*)

numSamples

public *Builder* **numSamples** (int *numSamples*)

oversampling

public *Builder* **oversampling** (int *oversampling*)

resLateral

public *Builder* **resLateral** (double *resLateral*)

resPSF

public *Builder* **resPSF** (double *resPSF*)

resPSFAxial

public *Builder* **resPSFAxial** (double *resPSFAxial*)

sizeX

public *Builder* **sizeX** (int *sizeX*)

sizeY

public *Builder* **sizeY** (int *sizeY*)

solver

public *Builder* **solver** (String *solver*)

stageDisplacement

public *Builder* **stageDisplacement** (double *stageDisplacement*)

tg

public *Builder* **tg** (double *tg*)

tg0

public *Builder* **tg0** (double *tg0*)

ti0

public *Builder* **ti0** (double *ti0*)

wavelength

public *Builder* **wavelength** (double *wavelength*)

6.28.9 GibsonLanniPSFTest

public class **GibsonLanniPSFTest**

Tests for the GibsonLanniPSF class.

Author Kyle M. Douglass

Constructors

GibsonLanniPSFTest

public **GibsonLanniPSFTest** ()

Methods

setUp

public void **setUp** ()

testGeneratePixelSignature

public void **testGeneratePixelSignature** ()

Test of generatePixelSignature method, of class GibsonLanniPSF.

testGenerateSignature

public void **testGenerateSignature** ()

Test of generateSignature method, of class GibsonLanniPSF.

testGetRadius

public void **testGetRadius** ()

Test of getRadius method, of class GibsonLanniPSF.

testGetRadiusSmallMaxRadius

public void **testGetRadiusSmallMaxRadius** ()
Test of getRadius method, of class GibsonLanniPSF, with maxRadius small.

6.28.10 ProfileGibsonLanniPSF

public class **ProfileGibsonLanniPSF**
Demonstrates how to create a Gibson-Lanni PSF.

Author Kyle M. Douglass

Methods

main

public static void **main** (String[] *args*)

6.29 ch.epfl.leb.sass.models.samples

6.29.1 RefractiveIndex

public interface **RefractiveIndex**
Common methods for the sample's refractive index distribution.

Author Kyle M. Douglass

Methods

getN

public Complex **getN** (double *x*, double *y*, double *z*)
Returns the (complex) refractive index at the position (x, y, z). z = 0 corresponds to the plane of the coverslip.

Parameters

- **x** – The x-position within the sample.
- **y** – The y-position within the sample.
- **z** – The z-position within the sample.

Returns The complex refractive index at the position (x, y, z).

6.30 ch.epfl.leb.sass.models.samples.internal

6.30.1 UniformRefractiveIndex

public class **UniformRefractiveIndex** implements *RefractiveIndex*
A sample with uniform and isotropic refractive index throughout all of space.

Author Kyle M. Douglass

Constructors

UniformRefractiveIndex

public **`UniformRefractiveIndex`**(Complex *refractiveIndex*)

Constructs a new UniformRefractiveIndex instance. The index of refraction is the same and isotropic everywhere in space.

Parameters

- **`refractiveIndex`** – The complex index of refraction.

Methods

getN

public Complex **`getN`**(double *x*, double *y*, double *z*)

Returns the (complex) refractive index at the position (x, y, z). z = 0 corresponds to the plane of the coverslip.

Parameters

- **`x`** – The x-position within the sample.
- **`y`** – The y-position within the sample.
- **`z`** – The z-position within the sample.

Returns The complex refractive index at the position (x, y, z).

6.30.2 UniformRefractiveIndexTest

public class **`UniformRefractiveIndexTest`**

Unit tests for the UniformRefractiveIndex object.

Author Kyle M. Douglass

Methods

setUp

public void **`setUp`**()

testGetN

public void **`testGetN`**()

Test of getN method, of class UniformRefractiveIndex.

6.31 ch.epfl.leb.sass.server

6.31.1 ImageGenerationException

public class **ImageGenerationException** extends org.apache.thrift.TException implements org.apache.thrift.TBase<*ImageGene*

Fields

metaDataMap

public static final java.util.Map<*_Fields*, org.apache.thrift.meta_data.FieldMetaData> **metaDataMap**

Constructors

ImageGenerationException

public **ImageGenerationException**()

ImageGenerationException

public **ImageGenerationException**(*ImageGenerationException other*)
Performs a deep copy on *other*.

Methods

clear

public void **clear**()

compareTo

public int **compareTo**(*ImageGenerationException other*)

deepCopy

public *ImageGenerationException* **deepCopy**()

equals

public boolean **equals**(java.lang.Object *that*)

equals

public boolean **equals**(*ImageGenerationException that*)

fieldForId

public *_Fields* **fieldForId**(int *fieldId*)

getFieldValue

public java.lang.Object **getFieldValue**(*_Fields field*)

hashCode

public int **hashCode**()

isSet

public boolean **isSet**(*_Fields field*)

Returns true if field corresponding to fieldID is set (has been assigned a value) and false otherwise

read

public void **read**(org.apache.thrift.protocol.TProtocol *iprot*)

setFieldValue

public void **setFieldValue**(*_Fields field*, java.lang.Object *value*)

toString

public java.lang.String **toString**()

validate

public void **validate**()

write

public void **write**(org.apache.thrift.protocol.TProtocol *oprot*)

6.31.2 ImageGenerationException._Fields

public enum **_Fields** implements org.apache.thrift.TFieldIdEnum

The set of fields this struct contains, along with convenience methods for finding and manipulating them.

Enum Constants

6.31.3 RPCServer

public class **RPCServer**

An RPC server for remote control of the simulation over a network socket.

Author Kyle M. Douglass

Fields

handler

public static *RemoteSimulationServiceHandler* **handler**

processor

public static *RemoteSimulationService.Processor* **processor**

Constructors

RPCServer

public **RPCServer** (*IJPluginModel model*, int *port*)

Creates a new RPCServer and initializes–but does not start–it.

Parameters

- **model** – A model of a microscope to simulate.
- **port** – The port number for server communications.

RPCServer

public **RPCServer** (*Microscope microscope*, int *port*)

Creates a new RPCServer and initializes–but does not start–it.

Parameters

- **microscope** – An instance of a microscope to simulate.
- **port** – The port number for server communications.

RPCServer

public **RPCServer** (*SimulationManager manager*, int *port*)

Creates a new RPCServer and initializes–but does not start–it.

Parameters

- **manager** – An instance of a simulation manager.
- **port** – The port number for server communications.

Methods

isServing

public boolean **isServing**()
Checks the status of the server.

Returns Is the server running? (true or false)

main

public static void **main**(String[] *args*)
Main function used for testing the RPC server.

Parameters

- **args** –

serve

public void **serve**()
Starts the server.

simple

public static void **simple**(*RemoteSimulationService.Processor processor*)

stop

public void **stop**()
Stops the server.

6.31.4 RPCServerIT

public class **RPCServerIT**
Integration tests for the RPCServer.

Author Kyle M. Douglass

Fields

rpcClient

RPCClient **rpcClient**
A RPCClient that will be used to test server communications.

rpcServer

RPCServer **rpcServer**
A RPCServer that will be used to test client/server communications.

Methods

setUp

public void **setUp**()
Sets up two different Microscopes for acquisition simulations.

tearDown

public void **tearDown**()
Closes the server communications.

Throws

- `java.lang.InterruptedException` –

testCreateAndDeleteSimulation

public void **testCreateAndDeleteSimulation**()
Test of createSimulation and deleteSimulation methods, of class RemoteSimulationServiceHandler.

testGetControlSignal

public void **testGetControlSignal**()
Test of getControlSignal method, of class RemoteSimulationServiceHandler.

testGetFovSize

public void **testGetFovSize**()
Test of getFovSize method, of class RemoteSimulationServiceHandler.

testGetNextImageAndImageCount

public void **testGetNextImageAndImageCount**()
Test of getNextImage and getImageCount methods, of class RemoteSimulationServiceHandler.

testGetObjectSpacePixelSize

public void **testGetObjectSpacePixelSize**()
Test of getObjectSpacePixelSize method, of class RemoteSimulationServiceHandler.

testGetServerStatus

public void **testGetServerStatus**()
Test of getGetServerStatus method, of class RemoteSimulationServiceHandler.

testIncrementTimeStep

public void **testIncrementTimeStep**()
Test of incrementTimeStep method, of class RemoteSimulationServiceHandler.

testIsServing

public void **testIsServing**()
Test of isServing method, of class RPCServer.

testSetControlSignal

public void **testSetControlSignal**()
Test of setControlSignal method, of class RemoteSimulationServiceHandler.

testToJsonMessages

public void **testToJsonMessages**()
Test of toJsonMessages method, of class RemoteSimulationServiceHandler.

testToJsonStateCamera

public void **testToJsonStateCamera**()
Test of toJsonState and getCameraJsonName methods, of class RemoteSimulationServiceHandler.

testToJsonStateFluorescence

public void **testToJsonStateFluorescence**()
Test of toJsonState and getFluorescenceJsonName methods, of class RemoteSimulationServiceHandler.

testToJsonStateLaser

public void **testToJsonStateLaser**()
Test of toJsonState and getLaserJsonName methods, of class RemoteSimulationServiceHandler.

testToJsonStateObjective

public void **testToJsonStateObjective**()
Test of toJsonState and getObjectiveJsonName methods, of class RemoteSimulationServiceHandler.

testToJsonStateStage

public void **testToJsonStateStage**()
Test of toJsonState and getLaserJsonName methods, of class RemoteSimulationServiceHandler.

testTrueSignal

public void **testTrueSignal**()
Test of getShortTrueSignalDescription and getTrueSignal methods, of class RemoteSimulationServiceHandler.

6.31.5 RemoteSimulationService

public class **RemoteSimulationService**

6.31.6 RemoteSimulationService.AsyncClient

public static class **AsyncClient** extends org.apache.thrift.async.TAsyncClient implements *AsyncIface*

Constructors

AsyncClient

public **AsyncClient**(org.apache.thrift.protocol.TProtocolFactory *protocolFactory*, org.apache.thrift.async.TAsyncClientManager *clientManager*, org.apache.thrift.transport.TNonblockingTransport *transport*)

Methods

createSimulation

public void **createSimulation**(org.apache.thrift.async.AsyncMethodCallback<java.lang.Integer> *resultHandler*)

deleteSimulation

public void **deleteSimulation**(int *id*, org.apache.thrift.async.AsyncMethodCallback<Void> *resultHandler*)

getCameraJsonName

public void **getCameraJsonName**(int *id*, org.apache.thrift.async.AsyncMethodCallback<java.lang.String> *resultHandler*)

getControlSignal

public void **getControlSignal**(int *id*, org.apache.thrift.async.AsyncMethodCallback<java.lang.Double> *resultHandler*)

getFluorescenceJsonName

public void **getFluorescenceJsonName** (int *id*, org.apache.thrift.async.AsyncMethodCallback<java.lang.String> *resultHandler*)

getFovSize

public void **getFovSize** (int *id*, org.apache.thrift.async.AsyncMethodCallback<java.lang.Double> *resultHandler*)

getImageCount

public void **getImageCount** (int *id*, org.apache.thrift.async.AsyncMethodCallback<java.lang.Integer> *resultHandler*)

getLaserJsonName

public void **getLaserJsonName** (int *id*, org.apache.thrift.async.AsyncMethodCallback<java.lang.String> *resultHandler*)

getNextImage

public void **getNextImage** (int *id*, org.apache.thrift.async.AsyncMethodCallback<java.nio.ByteBuffer> *resultHandler*)

getObjectSpacePixelSize

public void **getObjectSpacePixelSize** (int *id*, org.apache.thrift.async.AsyncMethodCallback<java.lang.Double> *resultHandler*)

getObjectiveJsonName

public void **getObjectiveJsonName** (int *id*, org.apache.thrift.async.AsyncMethodCallback<java.lang.String> *resultHandler*)

getServerStatus

public void **getServerStatus** (org.apache.thrift.async.AsyncMethodCallback<java.lang.String> *resultHandler*)

getShortTrueSignalDescription

public void **getShortTrueSignalDescription** (int *id*, org.apache.thrift.async.AsyncMethodCallback<java.lang.String> *resultHandler*)

getStageJsonName

public void **getStageJsonName** (int *id*, org.apache.thrift.async.AsyncMethodCallback<java.lang.String> *resultHandler*)

getTrueSignal

public void **getTrueSignal** (int *id*, int *imageNum*, org.apache.thrift.async.AsyncMethodCallback<java.lang.Double> *resultHandler*)

incrementTimeStep

public void **incrementTimeStep** (int *id*, org.apache.thrift.async.AsyncMethodCallback<Void> *resultHandler*)

setControlSignal

public void **setControlSignal** (int *id*, double *power*, org.apache.thrift.async.AsyncMethodCallback<Void> *resultHandler*)

toJsonMessages

public void **toJsonMessages** (int *id*, org.apache.thrift.async.AsyncMethodCallback<java.lang.String> *resultHandler*)

toJsonState

public void **toJsonState** (int *id*, org.apache.thrift.async.AsyncMethodCallback<java.lang.String> *resultHandler*)

6.31.7 RemoteSimulationService.AsyncClient.Factory

public static class **Factory** implements org.apache.thrift.async.TAsyncClientFactory<*AsyncClient*>

Constructors

Factory

public **Factory** (org.apache.thrift.async.TAsyncClientManager *clientManager*, org.apache.thrift.protocol.TProtocolFactory *protocolFactory*)

Methods

getAsyncClient

public *AsyncClient* **getAsyncClient** (org.apache.thrift.transport.TNonblockingTransport *transport*)

6.31.8 RemoteSimulationService.AsyncClient.createSimulation_call

public static class **`createSimulation_call`** extends org.apache.thrift.async.TAsyncMethodCall<java.lang.Integer>

Constructors

createSimulation_call

public **`createSimulation_call`** (org.apache.thrift.async.AsyncMethodCallback<java.lang.Integer> *resultHandler*, org.apache.thrift.async.TAsyncClient *client*, org.apache.thrift.protocol.TProtocolFactory *protocolFactory*, org.apache.thrift.transport.TNonblockingTransport *transport*)

Methods

getResult

public java.lang.Integer **`getResult`** ()

write_args

public void **`write_args`** (org.apache.thrift.protocol.TProtocol *prot*)

6.31.9 RemoteSimulationService.AsyncClient.deleteSimulation_call

public static class **`deleteSimulation_call`** extends org.apache.thrift.async.TAsyncMethodCall<Void>

Constructors

deleteSimulation_call

public **`deleteSimulation_call`** (int *id*, org.apache.thrift.async.AsyncMethodCallback<Void> *resultHandler*, org.apache.thrift.async.TAsyncClient *client*, org.apache.thrift.protocol.TProtocolFactory *protocolFactory*, org.apache.thrift.transport.TNonblockingTransport *transport*)

Methods

getResult

public Void **`getResult`** ()

write_args

public void **`write_args`** (org.apache.thrift.protocol.TProtocol *prot*)

6.31.10 RemoteSimulationService.AsyncClient.getCameraJsonName_call

public static class **getCameraJsonName_call** extends org.apache.thrift.async.TAsyncMethodCall<java.lang.String>

Constructors

getCameraJsonName_call

public **getCameraJsonName_call** (int *id*, org.apache.thrift.async.AsyncMethodCallback<java.lang.String> *resultHandler*, org.apache.thrift.async.TAsyncClient *client*, org.apache.thrift.protocol.TProtocolFactory *protocolFactory*, org.apache.thrift.transport.TNonblockingTransport *transport*)

Methods

getResult

public java.lang.String **getResult** ()

write_args

public void **write_args** (org.apache.thrift.protocol.TProtocol *prot*)

6.31.11 RemoteSimulationService.AsyncClient.getControlSignal_call

public static class **getControlSignal_call** extends org.apache.thrift.async.TAsyncMethodCall<java.lang.Double>

Constructors

getControlSignal_call

public **getControlSignal_call** (int *id*, org.apache.thrift.async.AsyncMethodCallback<java.lang.Double> *resultHandler*, org.apache.thrift.async.TAsyncClient *client*, org.apache.thrift.protocol.TProtocolFactory *protocolFactory*, org.apache.thrift.transport.TNonblockingTransport *transport*)

Methods

getResult

public java.lang.Double **getResult** ()

write_args

public void **write_args** (org.apache.thrift.protocol.TProtocol *prot*)

6.31.12 RemoteSimulationService.AsyncClient.getFluorescenceJsonName_call

public static class **getFluorescenceJsonName_call** extends org.apache.thrift.async.TAsyncMethodCall<java.lang.String>

Constructors

getFluorescenceJsonName_call

public **getFluorescenceJsonName_call** (int *id*, org.apache.thrift.async.AsyncMethodCallback<java.lang.String> *resultHandler*, org.apache.thrift.async.TAsyncClient *client*, org.apache.thrift.protocol.TProtocolFactory *protocolFactory*, org.apache.thrift.transport.TNonblockingTransport *transport*)

Methods

getResult

public java.lang.String **getResult** ()

write_args

public void **write_args** (org.apache.thrift.protocol.TProtocol *prot*)

6.31.13 RemoteSimulationService.AsyncClient.getFovSize_call

public static class **getFovSize_call** extends org.apache.thrift.async.TAsyncMethodCall<java.lang.Double>

Constructors

getFovSize_call

public **getFovSize_call** (int *id*, org.apache.thrift.async.AsyncMethodCallback<java.lang.Double> *resultHandler*, org.apache.thrift.async.TAsyncClient *client*, org.apache.thrift.protocol.TProtocolFactory *protocolFactory*, org.apache.thrift.transport.TNonblockingTransport *transport*)

Methods

getResult

public java.lang.Double **getResult** ()

write_args

public void **write_args** (org.apache.thrift.protocol.TProtocol *prot*)

6.31.14 RemoteSimulationService.AsyncClient.getImageCount_call

public static class **getImageCount_call** extends org.apache.thrift.async.TAsyncMethodCall<java.lang.Integer>

Constructors

getImageCount_call

public **getImageCount_call**(int *id*, org.apache.thrift.async.AsyncMethodCallback<java.lang.Integer> *resultHandler*, org.apache.thrift.async.TAsyncClient *client*, org.apache.thrift.protocol.TProtocolFactory *protocolFactory*, org.apache.thrift.transport.TNonblockingTransport *transport*)

Methods

getResult

public java.lang.Integer **getResult**()

write_args

public void **write_args**(org.apache.thrift.protocol.TProtocol *prot*)

6.31.15 RemoteSimulationService.AsyncClient.getLaserJsonName_call

public static class **getLaserJsonName_call** extends org.apache.thrift.async.TAsyncMethodCall<java.lang.String>

Constructors

getLaserJsonName_call

public **getLaserJsonName_call**(int *id*, org.apache.thrift.async.AsyncMethodCallback<java.lang.String> *resultHandler*, org.apache.thrift.async.TAsyncClient *client*, org.apache.thrift.protocol.TProtocolFactory *protocolFactory*, org.apache.thrift.transport.TNonblockingTransport *transport*)

Methods

getResult

public java.lang.String **getResult**()

write_args

public void **write_args**(org.apache.thrift.protocol.TProtocol *prot*)

6.31.16 RemoteSimulationService.AsyncClient.getNextImage_call

public static class **getNextImage_call** extends org.apache.thrift.async.TAsyncMethodCall<java.nio.ByteBuffer>

Constructors

getNextImage_call

public **getNextImage_call** (int *id*, org.apache.thrift.async.AsyncMethodCallback<java.nio.ByteBuffer> *resultHandler*, org.apache.thrift.async.TAsyncClient *client*, org.apache.thrift.protocol.TProtocolFactory *protocolFactory*, org.apache.thrift.transport.TNonblockingTransport *transport*)

Methods

getResult

public java.nio.ByteBuffer **getResult** ()

write_args

public void **write_args** (org.apache.thrift.protocol.TProtocol *prot*)

6.31.17 RemoteSimulationService.AsyncClient.getObjectSpacePixelSize_call

public static class **getObjectSpacePixelSize_call** extends org.apache.thrift.async.TAsyncMethodCall<java.lang.Double>

Constructors

getObjectSpacePixelSize_call

public **getObjectSpacePixelSize_call** (int *id*, org.apache.thrift.async.AsyncMethodCallback<java.lang.Double> *resultHandler*, org.apache.thrift.async.TAsyncClient *client*, org.apache.thrift.protocol.TProtocolFactory *protocolFactory*, org.apache.thrift.transport.TNonblockingTransport *transport*)

Methods

getResult

public java.lang.Double **getResult** ()

write_args

public void **write_args** (org.apache.thrift.protocol.TProtocol *prot*)

6.31.18 RemoteSimulationService.AsyncClient.getObjectiveJsonName_call

public static class **getObjectiveJsonName_call** extends org.apache.thrift.async.TAsyncMethodCall<java.lang.String>

Constructors

getObjectiveJsonName_call

public **getObjectiveJsonName_call** (int *id*, org.apache.thrift.async.AsyncMethodCallback<java.lang.String> *resultHandler*, org.apache.thrift.async.TAsyncClient *client*, org.apache.thrift.protocol.TProtocolFactory *protocolFactory*, org.apache.thrift.transport.TNonblockingTransport *transport*)

Methods

getResult

public java.lang.String **getResult** ()

write_args

public void **write_args** (org.apache.thrift.protocol.TProtocol *prot*)

6.31.19 RemoteSimulationService.AsyncClient.getServerStatus_call

public static class **getServerStatus_call** extends org.apache.thrift.async.TAsyncMethodCall<java.lang.String>

Constructors

getServerStatus_call

public **getServerStatus_call** (org.apache.thrift.async.AsyncMethodCallback<java.lang.String> *resultHandler*, org.apache.thrift.async.TAsyncClient *client*, org.apache.thrift.protocol.TProtocolFactory *protocolFactory*, org.apache.thrift.transport.TNonblockingTransport *transport*)

Methods

getResult

public java.lang.String **getResult** ()

write_args

public void **write_args** (org.apache.thrift.protocol.TProtocol *prot*)

6.31.20 RemoteSimulationService.AsyncClient.getShortTrueSignalDescription_call

public static class **getShortTrueSignalDescription_call** extends org.apache.thrift.async.TAsyncMethodCall<java.lang.Stri

Constructors

getShortTrueSignalDescription_call

public **getShortTrueSignalDescription_call** (int *id*, org.apache.thrift.async.AsyncMethodCallback<java.lang.String> *resultHandler*, org.apache.thrift.async.TAsyncClient *client*, org.apache.thrift.protocol.TProtocolFactory *protocolFactory*, org.apache.thrift.transport.TNonblockingTransport *transport*)

Methods

getResult

public java.lang.String **getResult** ()

write_args

public void **write_args** (org.apache.thrift.protocol.TProtocol *prot*)

6.31.21 RemoteSimulationService.AsyncClient.getStageJsonName_call

public static class **getStageJsonName_call** extends org.apache.thrift.async.TAsyncMethodCall<java.lang.String>

Constructors

getStageJsonName_call

public **getStageJsonName_call** (int *id*, org.apache.thrift.async.AsyncMethodCallback<java.lang.String> *resultHandler*, org.apache.thrift.async.TAsyncClient *client*, org.apache.thrift.protocol.TProtocolFactory *protocolFactory*, org.apache.thrift.transport.TNonblockingTransport *transport*)

Methods

getResult

public java.lang.String **getResult** ()

write_args

public void **write_args** (org.apache.thrift.protocol.TProtocol *prot*)

6.31.22 RemoteSimulationService.AsyncClient.getTrueSignal_call

public static class **getTrueSignal_call** extends org.apache.thrift.async.TAsyncMethodCall<java.lang.Double>

Constructors

getTrueSignal_call

public **getTrueSignal_call** (int *id*, int *imageNum*, org.apache.thrift.async.AsyncMethodCallback<java.lang.Double> *resultHandler*, org.apache.thrift.async.TAsyncClient *client*, org.apache.thrift.protocol.TProtocolFactory *protocolFactory*, org.apache.thrift.transport.TNonblockingTransport *transport*)

Methods

getResult

public java.lang.Double **getResult** ()

write_args

public void **write_args** (org.apache.thrift.protocol.TProtocol *prot*)

6.31.23 RemoteSimulationService.AsyncClient.incrementTimeStep_call

public static class **incrementTimeStep_call** extends org.apache.thrift.async.TAsyncMethodCall<Void>

Constructors

incrementTimeStep_call

public **incrementTimeStep_call** (int *id*, org.apache.thrift.async.AsyncMethodCallback<Void> *resultHandler*, org.apache.thrift.async.TAsyncClient *client*, org.apache.thrift.protocol.TProtocolFactory *protocolFactory*, org.apache.thrift.transport.TNonblockingTransport *transport*)

Methods

getResult

public Void **getResult** ()

write_args

public void **write_args** (org.apache.thrift.protocol.TProtocol *prot*)

6.31.24 RemoteSimulationService.AsyncClient.setControlSignal_call

public static class **setControlSignal_call** extends org.apache.thrift.async.TAsyncMethodCall<Void>

Constructors

setControlSignal_call

public **setControlSignal_call** (int *id*, double *power*, org.apache.thrift.async.AsyncMethodCallback<Void> *resultHandler*, org.apache.thrift.async.TAsyncClient *client*, org.apache.thrift.protocol.TProtocolFactory *protocolFactory*, org.apache.thrift.transport.TNonblockingTransport *transport*)

Methods

getResult

public Void **getResult** ()

write_args

public void **write_args** (org.apache.thrift.protocol.TProtocol *prot*)

6.31.25 RemoteSimulationService.AsyncClient.toJsonMessages_call

public static class **toJsonMessages_call** extends org.apache.thrift.async.TAsyncMethodCall<java.lang.String>

Constructors

toJsonMessages_call

public **toJsonMessages_call** (int *id*, org.apache.thrift.async.AsyncMethodCallback<java.lang.String> *resultHandler*, org.apache.thrift.async.TAsyncClient *client*, org.apache.thrift.protocol.TProtocolFactory *protocolFactory*, org.apache.thrift.transport.TNonblockingTransport *transport*)

Methods

getResult

public java.lang.String **getResult** ()

write_args

public void **write_args** (org.apache.thrift.protocol.TProtocol *prot*)

6.31.26 RemoteSimulationService.AsyncClient.toJsonState_call

public static class **toJsonState_call** extends org.apache.thrift.async.TAsyncMethodCall<java.lang.String>

Constructors

toJsonState_call

public **toJsonState_call** (int *id*, org.apache.thrift.async.AsyncMethodCallback<java.lang.String> *resultHandler*, org.apache.thrift.async.TAsyncClient *client*, org.apache.thrift.protocol.TProtocolFactory *protocolFactory*, org.apache.thrift.transport.TNonblockingTransport *transport*)

Methods

getResult

public java.lang.String **getResult** ()

write_args

public void **write_args** (org.apache.thrift.protocol.TProtocol *prot*)

6.31.27 RemoteSimulationService.AsyncIface

public interface **AsyncIface**

Methods

createSimulation

public void **createSimulation** (org.apache.thrift.async.AsyncMethodCallback<java.lang.Integer> *resultHandler*)

deleteSimulation

public void **deleteSimulation** (int *id*, org.apache.thrift.async.AsyncMethodCallback<Void> *resultHandler*)

getCameraJsonName

public void **getCameraJsonName** (int *id*, org.apache.thrift.async.AsyncMethodCallback<java.lang.String> *resultHandler*)

getControlSignal

public void **getControlSignal** (int *id*, org.apache.thrift.async.AsyncMethodCallback<java.lang.Double> *resultHandler*)

getFluorescenceJsonName

public void **getFluorescenceJsonName** (int *id*, org.apache.thrift.async.AsyncMethodCallback<java.lang.String> *resultHandler*)

getFovSize

public void **getFovSize** (int *id*, org.apache.thrift.async.AsyncMethodCallback<java.lang.Double> *resultHandler*)

getImageCount

public void **getImageCount** (int *id*, org.apache.thrift.async.AsyncMethodCallback<java.lang.Integer> *resultHandler*)

getLaserJsonName

public void **getLaserJsonName** (int *id*, org.apache.thrift.async.AsyncMethodCallback<java.lang.String> *resultHandler*)

getNextImage

public void **getNextImage** (int *id*, org.apache.thrift.async.AsyncMethodCallback<java.nio.ByteBuffer> *resultHandler*)

getObjectSpacePixelSize

public void **getObjectSpacePixelSize** (int *id*, org.apache.thrift.async.AsyncMethodCallback<java.lang.Double> *resultHandler*)

getObjectiveJsonName

public void **getObjectiveJsonName** (int *id*, org.apache.thrift.async.AsyncMethodCallback<java.lang.String> *resultHandler*)

getServerStatus

public void **getServerStatus** (org.apache.thrift.async.AsyncMethodCallback<java.lang.String> *resultHandler*)

getShortTrueSignalDescription

public void **getShortTrueSignalDescription** (int *id*, org.apache.thrift.async.AsyncMethodCallback<java.lang.String> *resultHandler*)

getStageJsonName

public void **getStageJsonName** (int *id*, org.apache.thrift.async.AsyncMethodCallback<java.lang.String> *resultHandler*)

getTrueSignal

public void **getTrueSignal** (int *id*, int *imageNum*, org.apache.thrift.async.AsyncMethodCallback<java.lang.Double> *resultHandler*)

incrementTimeStep

public void **incrementTimeStep** (int *id*, org.apache.thrift.async.AsyncMethodCallback<Void> *resultHandler*)

setControlSignal

public void **setControlSignal** (int *id*, double *power*, org.apache.thrift.async.AsyncMethodCallback<Void> *resultHandler*)

toJsonMessages

public void **toJsonMessages** (int *id*, org.apache.thrift.async.AsyncMethodCallback<java.lang.String> *resultHandler*)

toJsonState

public void **toJsonState** (int *id*, org.apache.thrift.async.AsyncMethodCallback<java.lang.String> *resultHandler*)

6.31.28 RemoteSimulationService.AsyncProcessor

public static class **AsyncProcessor**<I extends AsyncIface> extends org.apache.thrift.TBaseAsyncProcessor<I>

Constructors

AsyncProcessor

public **AsyncProcessor** (I *iface*)

AsyncProcessor

protected **AsyncProcessor** (I *iface*, java.util.Map<java.lang.String, org.apache.thrift.AsyncProcessFunction<I, ? extends org.apache.thrift.TBase, ?>> *processMap*)

6.31.29 RemoteSimulationService.AsyncProcessor.createSimulation

public static class **createSimulation**<I extends AsyncIface> extends org.apache.thrift.AsyncProcessFunction<I, *createSimulation*

Constructors

createSimulation

public **createSimulation** ()

Methods

getEmptyArgsInstance

public *createSimulation_args* **getEmptyArgsInstance** ()

getResultHandler

public org.apache.thrift.async.AsyncMethodCallback<java.lang.Integer> **getResultHandler** (org.apache.thrift.server.AbstractNon *fb*, int *seqid*)

isOneway

protected boolean **isOneway** ()

start

public void **start** (I *iface*, *createSimulation_args* *args*, org.apache.thrift.async.AsyncMethodCallback<java.lang.Integer> *resultHandler*)

6.31.30 RemoteSimulationService.AsyncProcessor.deleteSimulation

public static class **deleteSimulation**<I extends AsyncIface> extends org.apache.thrift.AsyncProcessFunction<I, *deleteSimulation*

Constructors

deleteSimulation

public **deleteSimulation** ()

Methods

getEmptyArgsInstance

public *deleteSimulation_args* **getEmptyArgsInstance**()

getResultHandler

public org.apache.thrift.async.AsyncMethodCallback<Void> **getResultHandler**(org.apache.thrift.server.AbstractNonblockingSer
fb, int *seqid*)

isOneway

protected boolean **isOneway**()

start

public void **start**(I *iface*, *deleteSimulation_args* *args*, org.apache.thrift.async.AsyncMethodCallback<Void>
resultHandler)

6.31.31 RemoteSimulationService.AsyncProcessor.getCameraJsonName

public static class **getCameraJsonName**<I extends AsyncIface> extends org.apache.thrift.AsyncProcessFunction<I, *getCameraJson*

Constructors

getCameraJsonName

public **getCameraJsonName**()

Methods

getEmptyArgsInstance

public *getCameraJsonName_args* **getEmptyArgsInstance**()

getResultHandler

public org.apache.thrift.async.AsyncMethodCallback<java.lang.String> **getResultHandler**(org.apache.thrift.server.AbstractNonb
fb, int *seqid*)

isOneway

protected boolean **isOneway**()

start

public void **start** (I *iface*, *getCameraJsonName_args* *args*, org.apache.thrift.async.AsyncMethodCallback<java.lang.String> *resultHandler*)

6.31.32 RemoteSimulationService.AsyncProcessor.getControlSignal

public static class **getControlSignal**<I extends AsyncIface> extends org.apache.thrift.AsyncProcessFunction<I, *getControlSignal*

Constructors

getControlSignal

public **getControlSignal** ()

Methods

getEmptyArgsInstance

public *getControlSignal_args* **getEmptyArgsInstance** ()

getResultHandler

public org.apache.thrift.async.AsyncMethodCallback<java.lang.Double> **getResultHandler** (org.apache.thrift.server.AbstractNon *fb*, int *seqid*)

isOneway

protected boolean **isOneway** ()

start

public void **start** (I *iface*, *getControlSignal_args* *args*, org.apache.thrift.async.AsyncMethodCallback<java.lang.Double> *resultHandler*)

6.31.33 RemoteSimulationService.AsyncProcessor.getFluorescenceJsonName

public static class **getFluorescenceJsonName**<I extends AsyncIface> extends org.apache.thrift.AsyncProcessFunction<I, *getFlu*

Constructors

getFluorescenceJsonName

public **getFluorescenceJsonName** ()

Methods

getEmptyArgsInstance

public *getFluorescenceJsonName_args* **getEmptyArgsInstance**()

getResultHandler

public org.apache.thrift.async.AsyncMethodCallback<java.lang.String> **getResultHandler**(org.apache.thrift.server.AbstractNonb
fb, int *seqid*)

isOneway

protected boolean **isOneway**()

start

public void **start**(I *iface*, *getFluorescenceJsonName_args* *args*, org.apache.thrift.async.AsyncMethodCallback<java.lang.String>
resultHandler)

6.31.34 RemoteSimulationService.AsyncProcessor.getFovSize

public static class **getFovSize**<I extends AsyncIface> extends org.apache.thrift.AsyncProcessFunction<I, *getFovSize_args*, java.lang

Constructors

getFovSize

public **getFovSize**()

Methods

getEmptyArgsInstance

public *getFovSize_args* **getEmptyArgsInstance**()

getResultHandler

public org.apache.thrift.async.AsyncMethodCallback<java.lang.Double> **getResultHandler**(org.apache.thrift.server.AbstractNon
fb, int *se-*
qid)

isOneway

protected boolean **isOneway**()

start

public void **start** (I *iface*, *getFovSize_args* *args*, org.apache.thrift.async.AsyncMethodCallback<java.lang.Double> *resultHandler*)

6.31.35 RemoteSimulationService.AsyncProcessor.getImageCount

public static class **getImageCount**<I extends AsyncIface> extends org.apache.thrift.AsyncProcessFunction<I, *getImageCount_args*,

Constructors

getImageCount

public **getImageCount** ()

Methods

getEmptyArgsInstance

public *getImageCount_args* **getEmptyArgsInstance** ()

getResultHandler

public org.apache.thrift.async.AsyncMethodCallback<java.lang.Integer> **getResultHandler** (org.apache.thrift.server.AbstractNon *fb*, int *seqid*)

isOneway

protected boolean **isOneway** ()

start

public void **start** (I *iface*, *getImageCount_args* *args*, org.apache.thrift.async.AsyncMethodCallback<java.lang.Integer> *resultHandler*)

6.31.36 RemoteSimulationService.AsyncProcessor.getLaserJsonName

public static class **getLaserJsonName**<I extends AsyncIface> extends org.apache.thrift.AsyncProcessFunction<I, *getLaserJsonNa*

Constructors

getLaserJsonName

public **getLaserJsonName** ()

Methods

getEmptyArgsInstance

public *getLaserJsonName_args* **getEmptyArgsInstance**()

getResultHandler

public org.apache.thrift.async.AsyncMethodCallback<java.lang.String> **getResultHandler**(org.apache.thrift.server.AbstractNonb
fb, int *seqid*)

isOneway

protected boolean **isOneway**()

start

public void **start**(I *iface*, *getLaserJsonName_args* *args*, org.apache.thrift.async.AsyncMethodCallback<java.lang.String>
resultHandler)

6.31.37 RemoteSimulationService.AsyncProcessor.getNextImage

public static class **getNextImage**<I extends AsyncIface> extends org.apache.thrift.AsyncProcessFunction<I, *getNextImage_args*, jav

Constructors

getNextImage

public **getNextImage**()

Methods

getEmptyArgsInstance

public *getNextImage_args* **getEmptyArgsInstance**()

getResultHandler

public org.apache.thrift.async.AsyncMethodCallback<java.nio.ByteBuffer> **getResultHandler**(org.apache.thrift.server.AbstractN
fb, int
seqid)

isOneway

protected boolean **isOneway**()

start

public void **start** (I *iface*, *getNextImage_args* *args*, org.apache.thrift.async.AsyncMethodCallback<java.nio.ByteBuffer> *resultHandler*)

6.31.38 RemoteSimulationService.AsyncProcessor.getObjectSpacePixelSize

public static class **getObjectSpacePixelSize**<I extends AsyncIface> extends org.apache.thrift.AsyncProcessFunction<I, *getObj*

Constructors

getObjectSpacePixelSize

public **getObjectSpacePixelSize** ()

Methods

getEmptyArgsInstance

public *getObjectSpacePixelSize_args* **getEmptyArgsInstance** ()

getResultHandler

public org.apache.thrift.async.AsyncMethodCallback<java.lang.Double> **getResultHandler** (org.apache.thrift.server.AbstractNon *fb*, int *seqid*)

isOneway

protected boolean **isOneway** ()

start

public void **start** (I *iface*, *getObjectSpacePixelSize_args* *args*, org.apache.thrift.async.AsyncMethodCallback<java.lang.Double> *resultHandler*)

6.31.39 RemoteSimulationService.AsyncProcessor.getObjectiveJsonName

public static class **getObjectiveJsonName**<I extends AsyncIface> extends org.apache.thrift.AsyncProcessFunction<I, *getObjectiv*

Constructors

getObjectiveJsonName

public **getObjectiveJsonName** ()

Methods

getEmptyArgsInstance

public *getObjectiveJsonName_args* **getEmptyArgsInstance**()

getResultHandler

public org.apache.thrift.async.AsyncMethodCallback<java.lang.String> **getResultHandler**(org.apache.thrift.server.AbstractNonb
fb, int *seqid*)

isOneway

protected boolean **isOneway**()

start

public void **start**(I *iface*, *getObjectiveJsonName_args* *args*, org.apache.thrift.async.AsyncMethodCallback<java.lang.String>
resultHandler)

6.31.40 RemoteSimulationService.AsyncProcessor.getServerStatus

public static class **getServerStatus**<I extends AsyncIface> extends org.apache.thrift.AsyncProcessFunction<I, *getServerStatus_a*

Constructors

getServerStatus

public **getServerStatus**()

Methods

getEmptyArgsInstance

public *getServerStatus_args* **getEmptyArgsInstance**()

getResultHandler

public org.apache.thrift.async.AsyncMethodCallback<java.lang.String> **getResultHandler**(org.apache.thrift.server.AbstractNonb
fb, int *seqid*)

isOneway

protected boolean **isOneway**()

start

public void **start** (I *iface*, *getServerStatus_args* *args*, org.apache.thrift.async.AsyncMethodCallback<java.lang.String> *resultHandler*)

6.31.41 RemoteSimulationService.AsyncProcessor.getShortTrueSignalDescription

public static class **getShortTrueSignalDescription**<I extends AsyncIface> extends org.apache.thrift.AsyncProcessFunction<

Constructors

getShortTrueSignalDescription

public **getShortTrueSignalDescription** ()

Methods

getEmptyArgsInstance

public *getShortTrueSignalDescription_args* **getEmptyArgsInstance** ()

getResultHandler

public org.apache.thrift.async.AsyncMethodCallback<java.lang.String> **getResultHandler** (org.apache.thrift.server.AbstractNonb *fb*, int *seqid*)

isOneway

protected boolean **isOneway** ()

start

public void **start** (I *iface*, *getShortTrueSignalDescription_args* *args*, org.apache.thrift.async.AsyncMethodCallback<java.lang.String> *resultHandler*)

6.31.42 RemoteSimulationService.AsyncProcessor.getStageJsonName

public static class **getStageJsonName**<I extends AsyncIface> extends org.apache.thrift.AsyncProcessFunction<I, *getStageJsonNam*

Constructors

getStageJsonName

public **getStageJsonName** ()

Methods

getEmptyArgsInstance

public *getStageJsonName_args* **getEmptyArgsInstance**()

getResultHandler

public org.apache.thrift.async.AsyncMethodCallback<java.lang.String> **getResultHandler**(org.apache.thrift.server.AbstractNonb
fb, int *seqid*)

isOneway

protected boolean **isOneway**()

start

public void **start**(I *iface*, *getStageJsonName_args* *args*, org.apache.thrift.async.AsyncMethodCallback<java.lang.String>
resultHandler)

6.31.43 RemoteSimulationService.AsyncProcessor.getTrueSignal

public static class **getTrueSignal**<I extends AsyncIface> extends org.apache.thrift.AsyncProcessFunction<I, *getTrueSignal_args*, j

Constructors

getTrueSignal

public **getTrueSignal**()

Methods

getEmptyArgsInstance

public *getTrueSignal_args* **getEmptyArgsInstance**()

getResultHandler

public org.apache.thrift.async.AsyncMethodCallback<java.lang.Double> **getResultHandler**(org.apache.thrift.server.AbstractNon
fb, int *se-*
qid)

isOneway

protected boolean **isOneway**()

start

public void **start** (I *iface*, *getTrueSignal_args* *args*, org.apache.thrift.async.AsyncMethodCallback<java.lang.Double> *resultHandler*)

6.31.44 RemoteSimulationService.AsyncProcessor.incrementTimeStep

public static class **incrementTimeStep**<I extends AsyncIface> extends org.apache.thrift.AsyncProcessFunction<I, *incrementTimeS*

Constructors

incrementTimeStep

public **incrementTimeStep** ()

Methods

getEmptyArgsInstance

public *incrementTimeStep_args* **getEmptyArgsInstance** ()

getResultHandler

public org.apache.thrift.async.AsyncMethodCallback<Void> **getResultHandler** (org.apache.thrift.server.AbstractNonblockingSer *fb*, int *seqid*)

isOneway

protected boolean **isOneway** ()

start

public void **start** (I *iface*, *incrementTimeStep_args* *args*, org.apache.thrift.async.AsyncMethodCallback<Void> *resultHandler*)

6.31.45 RemoteSimulationService.AsyncProcessor.setControlSignal

public static class **setControlSignal**<I extends AsyncIface> extends org.apache.thrift.AsyncProcessFunction<I, *setControlSignal*

Constructors

setControlSignal

public **setControlSignal** ()

Methods

getEmptyArgsInstance

public *setControlSignal_args* **getEmptyArgsInstance**()

getResultHandler

public org.apache.thrift.async.AsyncMethodCallback<Void> **getResultHandler**(org.apache.thrift.server.AbstractNonblockingSer
fb, int *seqid*)

isOneway

protected boolean **isOneway**()

start

public void **start**(I *iface*, *setControlSignal_args* *args*, org.apache.thrift.async.AsyncMethodCallback<Void>
resultHandler)

6.31.46 RemoteSimulationService.AsyncProcessor.toJsonMessages

public static class **toJsonMessages**<I extends AsyncIface> extends org.apache.thrift.AsyncProcessFunction<I, *toJsonMessages_arg*

Constructors

toJsonMessages

public **toJsonMessages**()

Methods

getEmptyArgsInstance

public *toJsonMessages_args* **getEmptyArgsInstance**()

getResultHandler

public org.apache.thrift.async.AsyncMethodCallback<java.lang.String> **getResultHandler**(org.apache.thrift.server.AbstractNonb
fb, int *seqid*)

isOneway

protected boolean **isOneway**()

start

public void **start** (I *iface*, *toJsonMessages_args* *args*, org.apache.thrift.async.AsyncMethodCallback<java.lang.String> *resultHandler*)

6.31.47 RemoteSimulationService.AsyncProcessor.toJsonState

public static class **toJsonState**<I extends AsyncIface> extends org.apache.thrift.AsyncProcessFunction<I, *toJsonState_args*, java.la

Constructors

toJsonState

public **toJsonState** ()

Methods

getEmptyArgsInstance

public *toJsonState_args* **getEmptyArgsInstance** ()

getResultHandler

public org.apache.thrift.async.AsyncMethodCallback<java.lang.String> **getResultHandler** (org.apache.thrift.server.AbstractNonb *fb*, int *seqid*)

isOneway

protected boolean **isOneway** ()

start

public void **start** (I *iface*, *toJsonState_args* *args*, org.apache.thrift.async.AsyncMethodCallback<java.lang.String> *resultHandler*)

6.31.48 RemoteSimulationService.Client

public static class **Client** extends org.apache.thrift.TServiceClient implements *Iface*

Constructors

Client

public **Client** (org.apache.thrift.protocol.TProtocol *prot*)

Client

public **Client** (org.apache.thrift.protocol.TProtocol *iprot*, org.apache.thrift.protocol.TProtocol *oprot*)

Methods

createSimulation

public int **createSimulation** ()

deleteSimulation

public void **deleteSimulation** (int *id*)

getCameraJsonName

public java.lang.String **getCameraJsonName** (int *id*)

getControlSignal

public double **getControlSignal** (int *id*)

getFluorescenceJsonName

public java.lang.String **getFluorescenceJsonName** (int *id*)

getFovSize

public double **getFovSize** (int *id*)

getImageCount

public int **getImageCount** (int *id*)

getLaserJsonName

public java.lang.String **getLaserJsonName** (int *id*)

getNextImage

public java.nio.ByteBuffer **getNextImage** (int *id*)

getObjectSpacePixelSize

public double **getObjectSpacePixelSize** (int *id*)

getObjectiveJsonName

public java.lang.String **getObjectiveJsonName** (int *id*)

getServerStatus

public java.lang.String **getServerStatus** ()

getShortTrueSignalDescription

public java.lang.String **getShortTrueSignalDescription** (int *id*)

getStageJsonName

public java.lang.String **getStageJsonName** (int *id*)

getTrueSignal

public double **getTrueSignal** (int *id*, int *imageNum*)

incrementTimeStep

public void **incrementTimeStep** (int *id*)

recv_createSimulation

public int **recv_createSimulation** ()

recv_deleteSimulation

public void **recv_deleteSimulation** ()

recv_getCameraJsonName

public java.lang.String **recv_getCameraJsonName** ()

recv_getControlSignal

public double **recv_getControlSignal** ()

recv_getFluorescenceJsonName

public java.lang.String **recv_getFluorescenceJsonName**()

recv_getFovSize

public double **recv_getFovSize**()

recv_getImageCount

public int **recv_getImageCount**()

recv_getLaserJsonName

public java.lang.String **recv_getLaserJsonName**()

recv_getNextImage

public java.nio.ByteBuffer **recv_getNextImage**()

recv_getObjectSpacePixelSize

public double **recv_getObjectSpacePixelSize**()

recv_getObjectiveJsonName

public java.lang.String **recv_getObjectiveJsonName**()

recv_getServerStatus

public java.lang.String **recv_getServerStatus**()

recv_getShortTrueSignalDescription

public java.lang.String **recv_getShortTrueSignalDescription**()

recv_getStageJsonName

public java.lang.String **recv_getStageJsonName**()

recv_getTrueSignal

public double **recv_getTrueSignal**()

recv_incrementTimeStep

public void **recv_incrementTimeStep** ( )

recv_setControlSignal

public void **recv_setControlSignal** ( )

recv_toJsonMessages

public java.lang.String **recv_toJsonMessages** ( )

recv_toJsonState

public java.lang.String **recv_toJsonState** ( )

send_createSimulation

public void **send_createSimulation** ( )

send_deleteSimulation

public void **send_deleteSimulation** (int *id*)

send_getCameraJsonName

public void **send_getCameraJsonName** (int *id*)

send_getControlSignal

public void **send_getControlSignal** (int *id*)

send_getFluorescenceJsonName

public void **send_getFluorescenceJsonName** (int *id*)

send_getFovSize

public void **send_getFovSize** (int *id*)

send_getImageCount

public void **send_getImageCount** (int *id*)

send_getLaserJsonName

public void **send_getLaserJsonName** (int *id*)

send_getNextImage

public void **send_getNextImage** (int *id*)

send_getObjectSpacePixelSize

public void **send_getObjectSpacePixelSize** (int *id*)

send_getObjectiveJsonName

public void **send_getObjectiveJsonName** (int *id*)

send_getServerStatus

public void **send_getServerStatus** ()

send_getShortTrueSignalDescription

public void **send_getShortTrueSignalDescription** (int *id*)

send_getStageJsonName

public void **send_getStageJsonName** (int *id*)

send_getTrueSignal

public void **send_getTrueSignal** (int *id*, int *imageNum*)

send_incrementTimeStep

public void **send_incrementTimeStep** (int *id*)

send_setControlSignal

public void **send_setControlSignal** (int *id*, double *power*)

send_toJsonMessages

public void **send_toJsonMessages** (int *id*)

send_toJsonState

public void **send_toJsonState** (int *id*)

setControlSignal

public void **setControlSignal** (int *id*, double *power*)

toJsonMessages

public java.lang.String **toJsonMessages** (int *id*)

toJsonState

public java.lang.String **toJsonState** (int *id*)

6.31.49 RemoteSimulationService.Client.Factory

public static class **Factory** implements org.apache.thrift.TServiceClientFactory<*Client*>

Constructors

Factory

public **Factory** ()

Methods

getClient

public *Client* **getClient** (org.apache.thrift.protocol.TProtocol *prot*)

getClient

public *Client* **getClient** (org.apache.thrift.protocol.TProtocol *iprot*, org.apache.thrift.protocol.TProtocol *oprot*)

6.31.50 RemoteSimulationService.Iface

public interface **Iface**

RPC wrapper around the Simulator class.

Methods

createSimulation

public int **`createSimulation`**()

Creates a new simulation. The ID of the simulation is returned.

deleteSimulation

public void **`deleteSimulation`**(int *id*)

Deletes a simulation with the given ID.

Parameters

- **`id`** –

getCameraJsonName

public java.lang.String **`getCameraJsonName`**(int *id*)

Gets the name of the JSON key for the camera information.

Parameters

- **`id`** –

getControlSignal

public double **`getControlSignal`**(int *id*)

Returns the current value for the control signal.

Parameters

- **`id`** –

getFluorescenceJsonName

public java.lang.String **`getFluorescenceJsonName`**(int *id*)

Gets the name of the JSON key for the fluoresence information.

Parameters

- **`id`** –

getFovSize

public double **`getFovSize`**(int *id*)

Returns the size of the field-of-view in object space units.

Parameters

- **`id`** –

getImageCount

public int **getImageCount** (int *id*)

Returns the number of images that have been simulated.

Parameters

- **id** –

getLaserJsonName

public java.lang.String **getLaserJsonName** (int *id*)

Gets the name of the JSON key for the laser information.

Parameters

- **id** –

getNextImage

public java.nio.ByteBuffer **getNextImage** (int *id*)

Increments the simulation by one time step and returns an image.

Parameters

- **id** –

getObjectSpacePixelSize

public double **getObjectSpacePixelSize** (int *id*)

Returns the object space pixel size. Units are the same as those of the camera's pixel size.

Parameters

- **id** –

getObjectiveJsonName

public java.lang.String **getObjectiveJsonName** (int *id*)

Gets the name of the JSON key for the objective information.

Parameters

- **id** –

getServerStatus

public java.lang.String **getServerStatus** ()

Returns the simulation server's current status.

getShortTrueSignalDescription

public java.lang.String **getShortTrueSignalDescription** (int *id*)

Returns a brief description of the ground truth signal.

Parameters

- **id** –

getStageJsonName

public java.lang.String **getStageJsonName** (int *id*)

Gets the name of the JSON key for the stage information.

Parameters

- **id** –

getTrueSignal

public double **getTrueSignal** (int *id*, int *imageNum*)

Returns the true simulation signal at the given image.

Parameters

- **id** –
- **imageNum** –

incrementTimeStep

public void **incrementTimeStep** (int *id*)

Advances the simulation without creating a new image.

Parameters

- **id** –

setControlSignal

public void **setControlSignal** (int *id*, double *power*)

Changes the simulation's control signal.

Parameters

- **id** –
- **power** –

toJsonMessages

public java.lang.String **toJsonMessages** (int *id*)

Returns information about changes in the state of the simulation as a JSON string.

Parameters

- **id** –

toJsonState

public java.lang.String **toJsonState** (int *id*)

Returns information about the state of the simulation as a JSON string.

Parameters

- **id** –

6.31.51 RemoteSimulationService.Processor

public static class **Processor**<I extends Iface> extends org.apache.thrift.TBaseProcessor<I> implements org.apache.thrift.TProcesso

Constructors

Processor

public **Processor** (I *iface*)

Processor

protected **Processor** (I *iface*, java.util.Map<java.lang.String, org.apache.thrift.ProcessFunction<I, ? extends org.apache.thrift.TBase>> *processMap*)

6.31.52 RemoteSimulationService.Processor.createSimulation

public static class **createSimulation**<I extends Iface> extends org.apache.thrift.ProcessFunction<I, *createSimulation_args*>

Constructors

createSimulation

public **createSimulation** ()

Methods

getEmptyArgsInstance

public *createSimulation_args* **getEmptyArgsInstance** ()

getResult

public *createSimulation_result* **getResult** (I *iface*, *createSimulation_args* *args*)

handleRuntimeExceptions

protected boolean **handleRuntimeExceptions** ( )

isOneway

protected boolean **isOneway** ( )

6.31.53 RemoteSimulationService.Processor.deleteSimulation

public static class **deleteSimulation**<I extends Iface> extends org.apache.thrift.ProcessFunction<I, *deleteSimulation_args*>

Constructors

deleteSimulation

public **deleteSimulation** ( )

Methods

getEmptyArgsInstance

public *deleteSimulation_args* **getEmptyArgsInstance** ( )

getResult

public *deleteSimulation_result* **getResult** (I *iface*, *deleteSimulation_args* *args*)

handleRuntimeExceptions

protected boolean **handleRuntimeExceptions** ( )

isOneway

protected boolean **isOneway** ( )

6.31.54 RemoteSimulationService.Processor.getCameraJsonName

public static class **getCameraJsonName**<I extends Iface> extends org.apache.thrift.ProcessFunction<I, *getCameraJsonName_args*>

Constructors

getCameraJsonName

public **getCameraJsonName** ( )

Methods

getEmptyArgsInstance

public *getCameraJsonName_args* **getEmptyArgsInstance**()

getResult

public *getCameraJsonName_result* **getResult**(I *iface*, *getCameraJsonName_args* *args*)

handleRuntimeExceptions

protected boolean **handleRuntimeExceptions**()

isOneway

protected boolean **isOneway**()

6.31.55 RemoteSimulationService.Processor.getControlSignal

public static class **getControlSignal**<I extends Iface> extends org.apache.thrift.ProcessFunction<I, *getControlSignal_args*>

Constructors

getControlSignal

public **getControlSignal**()

Methods

getEmptyArgsInstance

public *getControlSignal_args* **getEmptyArgsInstance**()

getResult

public *getControlSignal_result* **getResult**(I *iface*, *getControlSignal_args* *args*)

handleRuntimeExceptions

protected boolean **handleRuntimeExceptions**()

isOneway

protected boolean **isOneway**()

6.31.56 RemoteSimulationService.Processor.getFluorescenceJsonName

public static class **getFluorescenceJsonName**<I extends Iface> extends org.apache.thrift.ProcessFunction<I, *getFluorescenceJso*

Constructors

getFluorescenceJsonName

public **getFluorescenceJsonName** ( )

Methods

getEmptyArgsInstance

public *getFluorescenceJsonName_args* **getEmptyArgsInstance** ( )

getResult

public *getFluorescenceJsonName_result* **getResult** (I *iface*, *getFluorescenceJsonName_args* *args*)

handleRuntimeExceptions

protected boolean **handleRuntimeExceptions** ( )

isOneway

protected boolean **isOneway** ( )

6.31.57 RemoteSimulationService.Processor.getFovSize

public static class **getFovSize**<I extends Iface> extends org.apache.thrift.ProcessFunction<I, *getFovSize_args*>

Constructors

getFovSize

public **getFovSize** ( )

Methods

getEmptyArgsInstance

public *getFovSize_args* **getEmptyArgsInstance** ( )

getResult

public *getFovSize_result* **getResult** (I *iface*, *getFovSize_args* *args*)

handleRuntimeExceptions

protected boolean **handleRuntimeExceptions** ()

isOneway

protected boolean **isOneway** ()

6.31.58 RemoteSimulationService.Processor.getImageCount

public static class **getImageCount**<I extends Iface> extends org.apache.thrift.ProcessFunction<I, *getImageCount_args*>

Constructors

getImageCount

public **getImageCount** ()

Methods

getEmptyArgsInstance

public *getImageCount_args* **getEmptyArgsInstance** ()

getResult

public *getImageCount_result* **getResult** (I *iface*, *getImageCount_args* *args*)

handleRuntimeExceptions

protected boolean **handleRuntimeExceptions** ()

isOneway

protected boolean **isOneway** ()

6.31.59 RemoteSimulationService.Processor.getLaserJsonName

public static class **getLaserJsonName**<I extends Iface> extends org.apache.thrift.ProcessFunction<I, *getLaserJsonName_args*>

Constructors

getLaserJsonName

public **getLaserJsonName**()

Methods

getEmptyArgsInstance

public *getLaserJsonName_args* **getEmptyArgsInstance**()

getResult

public *getLaserJsonName_result* **getResult**(I *iface*, *getLaserJsonName_args* *args*)

handleRuntimeExceptions

protected boolean **handleRuntimeExceptions**()

isOneway

protected boolean **isOneway**()

6.31.60 RemoteSimulationService.Processor.getNextImage

public static class **getNextImage**<I extends Iface> extends org.apache.thrift.ProcessFunction<I, *getNextImage_args*>

Constructors

getNextImage

public **getNextImage**()

Methods

getEmptyArgsInstance

public *getNextImage_args* **getEmptyArgsInstance**()

getResult

public *getNextImage_result* **getResult**(I *iface*, *getNextImage_args* *args*)

handleRuntimeExceptions

protected boolean **handleRuntimeExceptions** ( )

isOneway

protected boolean **isOneway** ( )

6.31.61 RemoteSimulationService.Processor.getObjectSpacePixelSize

public static class **getObjectSpacePixelSize**<I extends Iface> extends org.apache.thrift.ProcessFunction<I, *getObjectSpacePixe*

Constructors

getObjectSpacePixelSize

public **getObjectSpacePixelSize** ( )

Methods

getEmptyArgsInstance

public *getObjectSpacePixelSize_args* **getEmptyArgsInstance** ( )

getResult

public *getObjectSpacePixelSize_result* **getResult** (I *iface*, *getObjectSpacePixelSize_args* *args*)

handleRuntimeExceptions

protected boolean **handleRuntimeExceptions** ( )

isOneway

protected boolean **isOneway** ( )

6.31.62 RemoteSimulationService.Processor.getObjectiveJsonName

public static class **getObjectiveJsonName**<I extends Iface> extends org.apache.thrift.ProcessFunction<I, *getObjectiveJsonName_*

Constructors

getObjectiveJsonName

public **getObjectiveJsonName** ( )

Methods

getEmptyArgsInstance

public *getObjectiveJsonName_args* **getEmptyArgsInstance**()

getResult

public *getObjectiveJsonName_result* **getResult**(I *iface*, *getObjectiveJsonName_args* *args*)

handleRuntimeExceptions

protected boolean **handleRuntimeExceptions**()

isOneway

protected boolean **isOneway**()

6.31.63 RemoteSimulationService.Processor.getServerStatus

public static class **getServerStatus**<I extends Iface> extends org.apache.thrift.ProcessFunction<I, *getServerStatus_args*>

Constructors

getServerStatus

public **getServerStatus**()

Methods

getEmptyArgsInstance

public *getServerStatus_args* **getEmptyArgsInstance**()

getResult

public *getServerStatus_result* **getResult**(I *iface*, *getServerStatus_args* *args*)

handleRuntimeExceptions

protected boolean **handleRuntimeExceptions**()

isOneway

protected boolean **isOneway**()

6.31.64 RemoteSimulationService.Processor.getShortTrueSignalDescription

public static class **getShortTrueSignalDescription**<I extends Iface> extends org.apache.thrift.ProcessFunction<I, *getShortTr*

Constructors

getShortTrueSignalDescription

public **getShortTrueSignalDescription**()

Methods

getEmptyArgsInstance

public *getShortTrueSignalDescription_args* **getEmptyArgsInstance**()

getResult

public *getShortTrueSignalDescription_result* **getResult**(I *iface*, *getShortTrueSignalDescription_args* *args*)

handleRuntimeExceptions

protected boolean **handleRuntimeExceptions**()

isOneway

protected boolean **isOneway**()

6.31.65 RemoteSimulationService.Processor.getStageJsonName

public static class **getStageJsonName**<I extends Iface> extends org.apache.thrift.ProcessFunction<I, *getStageJsonName_args*>

Constructors

getStageJsonName

public **getStageJsonName**()

Methods

getEmptyArgsInstance

public *getStageJsonName_args* **getEmptyArgsInstance**()

getResult

public *getStageJsonName_result* **getResult** (I *iface*, *getStageJsonName_args* *args*)

handleRuntimeExceptions

protected boolean **handleRuntimeExceptions** ()

isOneway

protected boolean **isOneway** ()

6.31.66 RemoteSimulationService.Processor.getTrueSignal

public static class **getTrueSignal**<I extends Iface> extends org.apache.thrift.ProcessFunction<I, *getTrueSignal_args*>

Constructors

getTrueSignal

public **getTrueSignal** ()

Methods

getEmptyArgsInstance

public *getTrueSignal_args* **getEmptyArgsInstance** ()

getResult

public *getTrueSignal_result* **getResult** (I *iface*, *getTrueSignal_args* *args*)

handleRuntimeExceptions

protected boolean **handleRuntimeExceptions** ()

isOneway

protected boolean **isOneway** ()

6.31.67 RemoteSimulationService.Processor.incrementTimeStep

public static class **incrementTimeStep**<I extends Iface> extends org.apache.thrift.ProcessFunction<I, *incrementTimeStep_args*>

Constructors

incrementTimeStep

public **incrementTimeStep**()

Methods

getEmptyArgsInstance

public *incrementTimeStep_args* **getEmptyArgsInstance**()

getResult

public *incrementTimeStep_result* **getResult**(I *iface*, *incrementTimeStep_args* *args*)

handleRuntimeExceptions

protected boolean **handleRuntimeExceptions**()

isOneway

protected boolean **isOneway**()

6.31.68 RemoteSimulationService.Processor.setControlSignal

public static class **setControlSignal**<I extends Iface> extends org.apache.thrift.ProcessFunction<I, *setControlSignal_args*>

Constructors

setControlSignal

public **setControlSignal**()

Methods

getEmptyArgsInstance

public *setControlSignal_args* **getEmptyArgsInstance**()

getResult

public *setControlSignal_result* **getResult**(I *iface*, *setControlSignal_args* *args*)

handleRuntimeExceptions

protected boolean **handleRuntimeExceptions** ( )

isOneway

protected boolean **isOneway** ( )

6.31.69 RemoteSimulationService.Processor.toJsonMessages

public static class **toJsonMessages**<I extends Iface> extends org.apache.thrift.ProcessFunction<I, *toJsonMessages_args*>

Constructors

toJsonMessages

public **toJsonMessages** ( )

Methods

getEmptyArgsInstance

public *toJsonMessages_args* **getEmptyArgsInstance** ( )

getResult

public *toJsonMessages_result* **getResult** (I *iface*, *toJsonMessages_args* *args*)

handleRuntimeExceptions

protected boolean **handleRuntimeExceptions** ( )

isOneway

protected boolean **isOneway** ( )

6.31.70 RemoteSimulationService.Processor.toJsonState

public static class **toJsonState**<I extends Iface> extends org.apache.thrift.ProcessFunction<I, *toJsonState_args*>

Constructors

toJsonState

public **toJsonState** ( )

Methods

getEmptyArgsInstance

public *toJsonState_args* **getEmptyArgsInstance**()

getResult

public *toJsonState_result* **getResult**(I *iface*, *toJsonState_args* *args*)

handleRuntimeExceptions

protected boolean **handleRuntimeExceptions**()

isOneway

protected boolean **isOneway**()

6.31.71 RemoteSimulationService.createSimulation_args

public static class **createSimulation_args** implements org.apache.thrift.TBase<*createSimulation_args*, *createSimulation_args*.

Fields

metaDataMap

public static final java.util.Map<*_Fields*, org.apache.thrift.meta_data.FieldMetaData> **metaDataMap**

Constructors

createSimulation_args

public **createSimulation_args**()

createSimulation_args

public **createSimulation_args**(*createSimulation_args* *other*)
Performs a deep copy on *other*.

Methods

clear

public void **clear**()

compareTo

public int **compareTo** (*createSimulation_args other*)

deepCopy

public *createSimulation_args* **deepCopy** ()

equals

public boolean **equals** (java.lang.Object *that*)

equals

public boolean **equals** (*createSimulation_args that*)

fieldForId

public *_Fields* **fieldForId** (int *fieldId*)

getFieldValue

public java.lang.Object **getFieldValue** (*_Fields field*)

hashCode

public int **hashCode** ()

isSet

public boolean **isSet** (*_Fields field*)

Returns true if field corresponding to fieldID is set (has been assigned a value) and false otherwise

read

public void **read** (org.apache.thrift.protocol.TProtocol *iprot*)

setFieldValue

public void **setFieldValue** (*_Fields field*, java.lang.Object *value*)

toString

public java.lang.String **toString** ()

validate

public void **validate** ( )

write

public void **write** (org.apache.thrift.protocol.TProtocol *oprot*)

6.31.72 RemoteSimulationService.createSimulation_args._Fields

public enum **_Fields** implements org.apache.thrift.TFieldIdEnum
The set of fields this struct contains, along with convenience methods for finding and manipulating them.

Enum Constants

6.31.73 RemoteSimulationService.createSimulation_result

public static class **createSimulation_result** implements org.apache.thrift.TBase<*createSimulation_result*, *createSimulation_re*

Fields

metaDataMap

public static final java.util.Map<*_Fields*, org.apache.thrift.meta_data.FieldMetaData> **metaDataMap**

success

public int **success**

Constructors

createSimulation_result

public **createSimulation_result** ( )

createSimulation_result

public **createSimulation_result** (int *success*)

createSimulation_result

public **createSimulation_result** (*createSimulation_result other*)
Performs a deep copy on *other*.

Methods

clear

public void **clear**()

compareTo

public int **compareTo**(*createSimulation_result other*)

deepCopy

public *createSimulation_result* **deepCopy**()

equals

public boolean **equals**(java.lang.Object *that*)

equals

public boolean **equals**(*createSimulation_result that*)

fieldForId

public *_Fields* **fieldForId**(int *fieldId*)

getFieldValue

public java.lang.Object **getFieldValue**(*_Fields field*)

getSuccess

public int **getSuccess**()

hashCode

public int **hashCode**()

isSet

public boolean **isSet**(*_Fields field*)

Returns true if field corresponding to fieldID is set (has been assigned a value) and false otherwise

isSetSuccess

public boolean **isSetSuccess**()
Returns true if field success is set (has been assigned a value) and false otherwise

read

public void **read**(org.apache.thrift.protocol.TProtocol *iprot*)

setFieldValue

public void **setFieldValue**(*_Fields field*, java.lang.Object *value*)

setSuccess

public *createSimulation_result* **setSuccess**(int *success*)

setSuccessIsSet

public void **setSuccessIsSet**(boolean *value*)

toString

public java.lang.String **toString**()

unsetSuccess

public void **unsetSuccess**()

validate

public void **validate**()

write

public void **write**(org.apache.thrift.protocol.TProtocol *oprot*)

6.31.74 RemoteSimulationService.createSimulation_result._Fields

public enum **_Fields** implements org.apache.thrift.TFieldIdEnum
The set of fields this struct contains, along with convenience methods for finding and manipulating them.

Enum Constants

SUCCESS

public static final *RemoteSimulationService.createSimulation_result._Fields* **SUCCESS**

6.31.75 RemoteSimulationService.deleteSimulation_args

public static class **deleteSimulation_args** implements org.apache.thrift.TBase<*deleteSimulation_args*, *deleteSimulation_args._*

Fields

id

public int **id**

metaDataMap

public static final java.util.Map<*_Fields*, org.apache.thrift.meta_data.FieldMetaData> **metaDataMap**

Constructors

deleteSimulation_args

public **deleteSimulation_args** ( )

deleteSimulation_args

public **deleteSimulation_args** (int *id*)

deleteSimulation_args

public **deleteSimulation_args** (*deleteSimulation_args other*)
Performs a deep copy on *other*.

Methods

clear

public void **clear** ( )

compareTo

public int **compareTo** (*deleteSimulation_args other*)

deepCopy

public *deleteSimulation_args* **deepCopy**()

equals

public boolean **equals**(java.lang.Object *that*)

equals

public boolean **equals**(*deleteSimulation_args* *that*)

fieldForId

public *_Fields* **fieldForId**(int *fieldId*)

getFieldValue

public java.lang.Object **getFieldValue**(*_Fields* *field*)

getId

public int **getId**()

hashCode

public int **hashCode**()

isSet

public boolean **isSet**(*_Fields* *field*)

Returns true if field corresponding to fieldID is set (has been assigned a value) and false otherwise

isSetId

public boolean **isSetId**()

Returns true if field id is set (has been assigned a value) and false otherwise

read

public void **read**(org.apache.thrift.protocol.TProtocol *iprot*)

setFieldValue

public void **setFieldValue** (*_Fields field*, java.lang.Object *value*)

setId

public *deleteSimulation_args* **setId** (int *id*)

setIdIsSet

public void **setIdIsSet** (boolean *value*)

toString

public java.lang.String **toString** ()

unsetId

public void **unsetId** ()

validate

public void **validate** ()

write

public void **write** (org.apache.thrift.protocol.TProtocol *oprot*)

6.31.76 RemoteSimulationService.deleteSimulation_args._Fields

public enum **_Fields** implements org.apache.thrift.TFieldIdEnum

The set of fields this struct contains, along with convenience methods for finding and manipulating them.

Enum Constants

ID

public static final *RemoteSimulationService.deleteSimulation_args._Fields* **ID**

6.31.77 RemoteSimulationService.deleteSimulation_result

public static class **deleteSimulation_result** implements org.apache.thrift.TBase<*deleteSimulation_result*, *deleteSimulation_re*

Fields

ex

public *UnknownSimulationIdException* **ex**

metaDataMap

public static final java.util.Map<*_Fields*, org.apache.thrift.meta_data.FieldMetaData> **metaDataMap**

Constructors

deleteSimulation_result

public **deleteSimulation_result** ()

deleteSimulation_result

public **deleteSimulation_result** (*UnknownSimulationIdException ex*)

deleteSimulation_result

public **deleteSimulation_result** (*deleteSimulation_result other*)
Performs a deep copy on *other*.

Methods

clear

public void **clear** ()

compareTo

public int **compareTo** (*deleteSimulation_result other*)

deepCopy

public *deleteSimulation_result* **deepCopy** ()

equals

public boolean **equals** (java.lang.Object *that*)

equals

public boolean **equals** (*deleteSimulation_result that*)

fieldForId

public *_Fields* **fieldForId** (int *fieldId*)

getEx

public *UnknownSimulationIdException* **getEx** ()

getFieldValue

public java.lang.Object **getFieldValue** (*_Fields field*)

hashCode

public int **hashCode** ()

isSet

public boolean **isSet** (*_Fields field*)

Returns true if field corresponding to fieldID is set (has been assigned a value) and false otherwise

isSetEx

public boolean **isSetEx** ()

Returns true if field ex is set (has been assigned a value) and false otherwise

read

public void **read** (org.apache.thrift.protocol.TProtocol *iprot*)

setEx

public *deleteSimulation_result* **setEx** (*UnknownSimulationIdException ex*)

setExIsSet

public void **setExIsSet** (boolean *value*)

setFieldValue

public void **setFieldValue** (*_Fields field*, java.lang.Object *value*)

toString

public java.lang.String **toString** ()

unsetEx

public void **unsetEx** ()

validate

public void **validate** ()

write

public void **write** (org.apache.thrift.protocol.TProtocol *oprot*)

6.31.78 RemoteSimulationService.deleteSimulation_result._Fields

public enum **_Fields** implements org.apache.thrift.TFieldIdEnum

The set of fields this struct contains, along with convenience methods for finding and manipulating them.

Enum Constants

EX

public static final *RemoteSimulationService.deleteSimulation_result._Fields* **EX**

6.31.79 RemoteSimulationService.getCameraJsonName_args

public static class **getCameraJsonName_args** implements org.apache.thrift.TBase<*getCameraJsonName_args*, *getCameraJsonNa*

Fields

id

public int **id**

metaDataMap

public static final java.util.Map<*_Fields*, org.apache.thrift.meta_data.FieldMetaData> **metaDataMap**

Constructors

getCameraJsonName_args

public **getCameraJsonName_args** ( )

getCameraJsonName_args

public **getCameraJsonName_args** (int *id*)

getCameraJsonName_args

public **getCameraJsonName_args** (*getCameraJsonName_args other*)
Performs a deep copy on *other*.

Methods

clear

public void **clear** ( )

compareTo

public int **compareTo** (*getCameraJsonName_args other*)

deepCopy

public *getCameraJsonName_args* **deepCopy** ( )

equals

public boolean **equals** (java.lang.Object *that*)

equals

public boolean **equals** (*getCameraJsonName_args that*)

fieldForId

public *_Fields* **fieldForId** (int *fieldId*)

getFieldValue

public java.lang.Object **getFieldValue** (*_Fields field*)

getId

public int **getId**()

hashCode

public int **hashCode**()

isSet

public boolean **isSet**(*_Fields field*)

Returns true if field corresponding to fieldID is set (has been assigned a value) and false otherwise

isSetId

public boolean **isSetId**()

Returns true if field id is set (has been assigned a value) and false otherwise

read

public void **read**(org.apache.thrift.protocol.TProtocol *iprot*)

setFieldValue

public void **setFieldValue**(*_Fields field*, java.lang.Object *value*)

setId

public *getCameraJsonName_args* **setId**(int *id*)

setIdIsSet

public void **setIdIsSet**(boolean *value*)

toString

public java.lang.String **toString**()

unsetId

public void **unsetId**()

validate

public void **validate** ( )

write

public void **write** (org.apache.thrift.protocol.TProtocol *oprot*)

6.31.80 RemoteSimulationService.getCameraJsonName_args._Fields

public enum **_Fields** implements org.apache.thrift.TFieldIdEnum
The set of fields this struct contains, along with convenience methods for finding and manipulating them.

Enum Constants

ID

public static final *RemoteSimulationService.getCameraJsonName_args._Fields* **ID**

6.31.81 RemoteSimulationService.getCameraJsonName_result

public static class **getCameraJsonName_result** implements org.apache.thrift.TBase<*getCameraJsonName_result*, *getCameraJso*

Fields

ex

public *UnknownSimulationIdException* **ex**

metaDataMap

public static final java.util.Map<*_Fields*, org.apache.thrift.meta_data.FieldMetaData> **metaDataMap**

success

public java.lang.String **success**

Constructors

getCameraJsonName_result

public **getCameraJsonName_result** ( )

getCameraJsonName_result

public **getCameraJsonName_result** (java.lang.String *success*, *UnknownSimulationIdException* *ex*)

getCameraJsonName_result

public **getCameraJsonName_result** (*getCameraJsonName_result* *other*)
Performs a deep copy on *other*.

Methods

clear

public void **clear** ()

compareTo

public int **compareTo** (*getCameraJsonName_result* *other*)

deepCopy

public *getCameraJsonName_result* **deepCopy** ()

equals

public boolean **equals** (java.lang.Object *that*)

equals

public boolean **equals** (*getCameraJsonName_result* *that*)

fieldForId

public *_Fields* **fieldForId** (int *fieldId*)

getEx

public *UnknownSimulationIdException* **getEx** ()

getFieldValue

public java.lang.Object **getFieldValue** (*_Fields* *field*)

getSuccess

public java.lang.String **getSuccess**()

hashCode

public int **hashCode**()

isSet

public boolean **isSet**(*_Fields field*)

Returns true if field corresponding to fieldID is set (has been assigned a value) and false otherwise

isSetEx

public boolean **isSetEx**()

Returns true if field ex is set (has been assigned a value) and false otherwise

isSetSuccess

public boolean **isSetSuccess**()

Returns true if field success is set (has been assigned a value) and false otherwise

read

public void **read**(org.apache.thrift.protocol.TProtocol *iprot*)

setEx

public *getCameraJsonName_result* **setEx**(*UnknownSimulationIdException ex*)

setExIsSet

public void **setExIsSet**(boolean *value*)

setFieldValue

public void **setFieldValue**(*_Fields field*, java.lang.Object *value*)

setSuccess

public *getCameraJsonName_result* **setSuccess**(java.lang.String *success*)

setSuccessIsSet

public void **setSuccessIsSet** (boolean *value*)

toString

public java.lang.String **toString** ()

unsetEx

public void **unsetEx** ()

unsetSuccess

public void **unsetSuccess** ()

validate

public void **validate** ()

write

public void **write** (org.apache.thrift.protocol.TProtocol *oprot*)

6.31.82 RemoteSimulationService.getCameraJsonName_result._Fields

public enum **_Fields** implements org.apache.thrift.TFieldIdEnum

The set of fields this struct contains, along with convenience methods for finding and manipulating them.

Enum Constants

EX

public static final *RemoteSimulationService.getCameraJsonName_result._Fields* **EX**

SUCCESS

public static final *RemoteSimulationService.getCameraJsonName_result._Fields* **SUCCESS**

6.31.83 RemoteSimulationService.getControlSignal_args

public static class **getControlSignal_args** implements org.apache.thrift.TBase<*getControlSignal_args*, *getControlSignal_args*.

Fields

id

public int **id**

metaDataMap

public static final java.util.Map<*_Fields*, org.apache.thrift.meta_data.FieldMetaData> **metaDataMap**

Constructors

getControlSignal_args

public **getControlSignal_args** ( )

getControlSignal_args

public **getControlSignal_args** (int *id*)

getControlSignal_args

public **getControlSignal_args** (*getControlSignal_args other*)
Performs a deep copy on *other*.

Methods

clear

public void **clear** ( )

compareTo

public int **compareTo** (*getControlSignal_args other*)

deepCopy

public *getControlSignal_args* **deepCopy** ( )

equals

public boolean **equals** (java.lang.Object *that*)

equals

public boolean **equals** (*getControlSignal_args that*)

fieldForId

public *_Fields* **fieldForId** (int *fieldId*)

getFieldValue

public java.lang.Object **getFieldValue** (*_Fields field*)

getId

public int **getId** ()

hashCode

public int **hashCode** ()

isSet

public boolean **isSet** (*_Fields field*)

Returns true if field corresponding to fieldID is set (has been assigned a value) and false otherwise

isSetId

public boolean **isSetId** ()

Returns true if field id is set (has been assigned a value) and false otherwise

read

public void **read** (org.apache.thrift.protocol.TProtocol *iprot*)

setFieldValue

public void **setFieldValue** (*_Fields field*, java.lang.Object *value*)

setId

public *getControlSignal_args* **setId** (int *id*)

setIdIsSet

public void **setIdIsSet** (boolean *value*)

toString

public java.lang.String **toString** ()

unsetId

public void **unsetId** ()

validate

public void **validate** ()

write

public void **write** (org.apache.thrift.protocol.TProtocol *oprot*)

6.31.84 RemoteSimulationService.getControlSignal_args._Fields

public enum **_Fields** implements org.apache.thrift.TFieldIdEnum

The set of fields this struct contains, along with convenience methods for finding and manipulating them.

Enum Constants

ID

public static final *RemoteSimulationService.getControlSignal_args._Fields* **ID**

6.31.85 RemoteSimulationService.getControlSignal_result

public static class **getControlSignal_result** implements org.apache.thrift.TBase<*getControlSignal_result*, *getControlSignal_re*

Fields

ex

public *UnknownSimulationIdException* **ex**

metaDataMap

public static final java.util.Map<*_Fields*, org.apache.thrift.meta_data.FieldMetaData> **metaDataMap**

success

public double **success**

Constructors

getControlSignal_result

public **getControlSignal_result** ()

getControlSignal_result

public **getControlSignal_result** (double *success*, *UnknownSimulationIdException* *ex*)

getControlSignal_result

public **getControlSignal_result** (*getControlSignal_result* *other*)
Performs a deep copy on *other*.

Methods

clear

public void **clear** ()

compareTo

public int **compareTo** (*getControlSignal_result* *other*)

deepCopy

public *getControlSignal_result* **deepCopy** ()

equals

public boolean **equals** (java.lang.Object *that*)

equals

public boolean **equals** (*getControlSignal_result* *that*)

fieldForId

public *_Fields* **fieldForId** (int *fieldId*)

getEx

public *UnknownSimulationIdException* **getEx**()

getFieldValue

public java.lang.Object **getFieldValue**(*_Fields field*)

getSuccess

public double **getSuccess**()

hashCode

public int **hashCode**()

isSet

public boolean **isSet**(*_Fields field*)

Returns true if field corresponding to fieldID is set (has been assigned a value) and false otherwise

isSetEx

public boolean **isSetEx**()

Returns true if field ex is set (has been assigned a value) and false otherwise

isSetSuccess

public boolean **isSetSuccess**()

Returns true if field success is set (has been assigned a value) and false otherwise

read

public void **read**(org.apache.thrift.protocol.TProtocol *iprot*)

setEx

public *getControlSignal_result* **setEx**(*UnknownSimulationIdException ex*)

setExIsSet

public void **setExIsSet**(boolean *value*)

setFieldValue

public void **setFieldValue** (*_Fields field*, java.lang.Object *value*)

setSuccess

public *getControlSignal_result* **setSuccess** (double *success*)

setSuccessIsSet

public void **setSuccessIsSet** (boolean *value*)

toString

public java.lang.String **toString** ()

unsetEx

public void **unsetEx** ()

unsetSuccess

public void **unsetSuccess** ()

validate

public void **validate** ()

write

public void **write** (org.apache.thrift.protocol.TProtocol *oprot*)

6.31.86 RemoteSimulationService.getControlSignal_result._Fields

public enum **_Fields** implements org.apache.thrift.TFieldIdEnum

The set of fields this struct contains, along with convenience methods for finding and manipulating them.

Enum Constants

EX

public static final *RemoteSimulationService.getControlSignal_result._Fields* **EX**

SUCCESS

public static final *RemoteSimulationService.getControlSignal_result._Fields* **SUCCESS**

6.31.87 RemoteSimulationService.getFluorescenceJsonName_args

public static class **getFluorescenceJsonName_args** implements org.apache.thrift.TBase<*getFluorescenceJsonName_args*, *getI*

Fields

id

public int **id**

metaDataMap

public static final java.util.Map<*_Fields*, org.apache.thrift.meta_data.FieldMetaData> **metaDataMap**

Constructors

getFluorescenceJsonName_args

public **getFluorescenceJsonName_args** ( )

getFluorescenceJsonName_args

public **getFluorescenceJsonName_args** (int *id*)

getFluorescenceJsonName_args

public **getFluorescenceJsonName_args** (*getFluorescenceJsonName_args* *other*)
Performs a deep copy on *other*.

Methods

clear

public void **clear** ( )

compareTo

public int **compareTo** (*getFluorescenceJsonName_args* *other*)

deepCopy

public *getFluorescenceJsonName_args* **deepCopy**()

equals

public boolean **equals**(java.lang.Object *that*)

equals

public boolean **equals**(*getFluorescenceJsonName_args* *that*)

fieldForId

public *_Fields* **fieldForId**(int *fieldId*)

getFieldValue

public java.lang.Object **getFieldValue**(*_Fields* *field*)

getId

public int **getId**()

hashCode

public int **hashCode**()

isSet

public boolean **isSet**(*_Fields* *field*)

Returns true if field corresponding to fieldID is set (has been assigned a value) and false otherwise

isSetId

public boolean **isSetId**()

Returns true if field id is set (has been assigned a value) and false otherwise

read

public void **read**(org.apache.thrift.protocol.TProtocol *iprot*)

setFieldValue

public void **setFieldValue** (*_Fields field*, java.lang.Object *value*)

setId

public *getFluorescenceJsonName_args* **setId** (int *id*)

setIdIsSet

public void **setIdIsSet** (boolean *value*)

toString

public java.lang.String **toString** ()

unsetId

public void **unsetId** ()

validate

public void **validate** ()

write

public void **write** (org.apache.thrift.protocol.TProtocol *oprot*)

6.31.88 RemoteSimulationService.getFluorescenceJsonName_args._Fields

public enum **_Fields** implements org.apache.thrift.TFieldIdEnum

The set of fields this struct contains, along with convenience methods for finding and manipulating them.

Enum Constants

ID

public static final *RemoteSimulationService.getFluorescenceJsonName_args._Fields* **ID**

6.31.89 RemoteSimulationService.getFluorescenceJsonName_result

public static class **getFluorescenceJsonName_result** implements org.apache.thrift.TBase<*getFluorescenceJsonName_result*,

Fields

ex

public *UnknownSimulationIdException* **ex**

metaDataMap

public static final java.util.Map<*_Fields*, org.apache.thrift.meta_data.FieldMetaData> **metaDataMap**

success

public java.lang.String **success**

Constructors

getFluorescenceJsonName_result

public **getFluorescenceJsonName_result** ()

getFluorescenceJsonName_result

public **getFluorescenceJsonName_result** (java.lang.String *success*, *UnknownSimulationIdException* *ex*)

getFluorescenceJsonName_result

public **getFluorescenceJsonName_result** (*getFluorescenceJsonName_result* *other*)

Performs a deep copy on *other*.

Methods

clear

public void **clear** ()

compareTo

public int **compareTo** (*getFluorescenceJsonName_result* *other*)

deepCopy

public *getFluorescenceJsonName_result* **deepCopy** ()

equals

public boolean **equals** (java.lang.Object *that*)

equals

public boolean **equals** (*getFluorescenceJsonName_result* *that*)

fieldForId

public *_Fields* **fieldForId** (int *fieldId*)

getEx

public *UnknownSimulationIdException* **getEx** ()

getFieldValue

public java.lang.Object **getFieldValue** (*_Fields* *field*)

getSuccess

public java.lang.String **getSuccess** ()

hashCode

public int **hashCode** ()

isSet

public boolean **isSet** (*_Fields* *field*)

Returns true if field corresponding to fieldID is set (has been assigned a value) and false otherwise

isSetEx

public boolean **isSetEx** ()

Returns true if field ex is set (has been assigned a value) and false otherwise

isSetSuccess

public boolean **isSetSuccess** ()

Returns true if field success is set (has been assigned a value) and false otherwise

read

public void **read** (org.apache.thrift.protocol.TProtocol *iprot*)

setEx

public *getFluorescenceJsonName_result* **setEx** (*UnknownSimulationIdException ex*)

setExIsSet

public void **setExIsSet** (boolean *value*)

setFieldValue

public void **setFieldValue** (*_Fields field*, java.lang.Object *value*)

setSuccess

public *getFluorescenceJsonName_result* **setSuccess** (java.lang.String *success*)

setSuccessIsSet

public void **setSuccessIsSet** (boolean *value*)

toString

public java.lang.String **toString** ()

unsetEx

public void **unsetEx** ()

unsetSuccess

public void **unsetSuccess** ()

validate

public void **validate** ()

write

public void **write** (org.apache.thrift.protocol.TProtocol *oprot*)

6.31.90 RemoteSimulationService.getFluorescenceJsonName_result._Fields

public enum **_Fields** implements org.apache.thrift.TFieldIdEnum

The set of fields this struct contains, along with convenience methods for finding and manipulating them.

Enum Constants

EX

public static final *RemoteSimulationService.getFluorescenceJsonName_result._Fields* **EX**

SUCCESS

public static final *RemoteSimulationService.getFluorescenceJsonName_result._Fields* **SUCCESS**

6.31.91 RemoteSimulationService.getFovSize_args

public static class **getFovSize_args** implements org.apache.thrift.TBase<*getFovSize_args*, *getFovSize_args._Fields*>, java.io.Serial

Fields

id

public int **id**

metaDataMap

public static final java.util.Map<*_Fields*, org.apache.thrift.meta_data.FieldMetaData> **metaDataMap**

Constructors

getFovSize_args

public **getFovSize_args**()

getFovSize_args

public **getFovSize_args**(int *id*)

getFovSize_args

public **getFovSize_args**(*getFovSize_args other*)

Performs a deep copy on *other*.

Methods

clear

public void **clear**()

compareTo

public int **compareTo**(*getFovSize_args other*)

deepCopy

public *getFovSize_args* **deepCopy**()

equals

public boolean **equals**(java.lang.Object *that*)

equals

public boolean **equals**(*getFovSize_args that*)

fieldForId

public *_Fields* **fieldForId**(int *fieldId*)

getFieldValue

public java.lang.Object **getFieldValue**(*_Fields field*)

getId

public int **getId**()

hashCode

public int **hashCode**()

isSet

public boolean **isSet**(*_Fields field*)

Returns true if field corresponding to fieldID is set (has been assigned a value) and false otherwise

isSetId

public boolean **isSetId**()
Returns true if field id is set (has been assigned a value) and false otherwise

read

public void **read**(org.apache.thrift.protocol.TProtocol *iprot*)

setFieldValue

public void **setFieldValue**(*_Fields field*, java.lang.Object *value*)

setId

public *getFovSize_args* **setId**(int *id*)

setIdIsSet

public void **setIdIsSet**(boolean *value*)

toString

public java.lang.String **toString**()

unsetId

public void **unsetId**()

validate

public void **validate**()

write

public void **write**(org.apache.thrift.protocol.TProtocol *oprot*)

6.31.92 RemoteSimulationService.getFovSize_args._Fields

public enum **_Fields** implements org.apache.thrift.TFieldIdEnum
The set of fields this struct contains, along with convenience methods for finding and manipulating them.

Enum Constants

ID

public static final *RemoteSimulationService.getFovSize_args._Fields* **ID**

6.31.93 RemoteSimulationService.getFovSize_result

public static class **getFovSize_result** implements org.apache.thrift.TBase<*getFovSize_result*, *getFovSize_result._Fields*>, java.io.

Fields

ex

public *UnknownSimulationIdException* **ex**

metaDataMap

public static final java.util.Map<*_Fields*, org.apache.thrift.meta_data.FieldMetaData> **metaDataMap**

success

public double **success**

Constructors

getFovSize_result

public **getFovSize_result** ()

getFovSize_result

public **getFovSize_result** (double *success*, *UnknownSimulationIdException ex*)

getFovSize_result

public **getFovSize_result** (*getFovSize_result other*)
Performs a deep copy on *other*.

Methods

clear

public void **clear** ()

compareTo

public int **compareTo** (*getFovSize_result other*)

deepCopy

public *getFovSize_result* **deepCopy** ()

equals

public boolean **equals** (java.lang.Object *that*)

equals

public boolean **equals** (*getFovSize_result that*)

fieldForId

public *_Fields* **fieldForId** (int *fieldId*)

getEx

public *UnknownSimulationIdException* **getEx** ()

getFieldValue

public java.lang.Object **getFieldValue** (*_Fields field*)

getSuccess

public double **getSuccess** ()

hashCode

public int **hashCode** ()

isSet

public boolean **isSet** (*_Fields field*)

Returns true if field corresponding to fieldID is set (has been assigned a value) and false otherwise

isSetEx

public boolean **isSetEx**()

Returns true if field ex is set (has been assigned a value) and false otherwise

isSetSuccess

public boolean **isSetSuccess**()

Returns true if field success is set (has been assigned a value) and false otherwise

read

public void **read**(org.apache.thrift.protocol.TProtocol *iprot*)

setEx

public *getFovSize_result* **setEx**(*UnknownSimulationIdException* *ex*)

setExIsSet

public void **setExIsSet**(boolean *value*)

setFieldValue

public void **setFieldValue**(*_Fields* *field*, java.lang.Object *value*)

setSuccess

public *getFovSize_result* **setSuccess**(double *success*)

setSuccessIsSet

public void **setSuccessIsSet**(boolean *value*)

toString

public java.lang.String **toString**()

unsetEx

public void **unsetEx**()

unsetSuccess

public void **unsetSuccess** ( )

validate

public void **validate** ( )

write

public void **write** (org.apache.thrift.protocol.TProtocol *oprot*)

6.31.94 RemoteSimulationService.getFovSize_result._Fields

public enum **_Fields** implements org.apache.thrift.TFieldIdEnum

The set of fields this struct contains, along with convenience methods for finding and manipulating them.

Enum Constants

EX

public static final *RemoteSimulationService.getFovSize_result._Fields* **EX**

SUCCESS

public static final *RemoteSimulationService.getFovSize_result._Fields* **SUCCESS**

6.31.95 RemoteSimulationService.getImageCount_args

public static class **getImageCount_args** implements org.apache.thrift.TBase<*getImageCount_args*, *getImageCount_args._Fields*>

Fields

id

public int **id**

metaDataMap

public static final java.util.Map<*_Fields*, org.apache.thrift.meta_data.FieldMetaData> **metaDataMap**

Constructors

getImageCount_args

public **getImageCount_args**()

getImageCount_args

public **getImageCount_args**(int *id*)

getImageCount_args

public **getImageCount_args**(*getImageCount_args other*)
Performs a deep copy on *other*.

Methods

clear

public void **clear**()

compareTo

public int **compareTo**(*getImageCount_args other*)

deepCopy

public *getImageCount_args* **deepCopy**()

equals

public boolean **equals**(java.lang.Object *that*)

equals

public boolean **equals**(*getImageCount_args that*)

fieldForId

public *_Fields* **fieldForId**(int *fieldId*)

getFieldValue

public java.lang.Object **getFieldValue**(*_Fields field*)

getId

public int **getId**()

hashCode

public int **hashCode**()

isSet

public boolean **isSet**(*_Fields field*)

Returns true if field corresponding to fieldID is set (has been assigned a value) and false otherwise

isSetId

public boolean **isSetId**()

Returns true if field id is set (has been assigned a value) and false otherwise

read

public void **read**(org.apache.thrift.protocol.TProtocol *iprot*)

setFieldValue

public void **setFieldValue**(*_Fields field*, java.lang.Object *value*)

setId

public *getImageCount_args* **setId**(int *id*)

setIdIsSet

public void **setIdIsSet**(boolean *value*)

toString

public java.lang.String **toString**()

unsetId

public void **unsetId**()

validate

public void **validate** ( )

write

public void **write** (org.apache.thrift.protocol.TProtocol *oprot*)

6.31.96 RemoteSimulationService.getImageCount_args._Fields

public enum **_Fields** implements org.apache.thrift.TFieldIdEnum
The set of fields this struct contains, along with convenience methods for finding and manipulating them.

Enum Constants

ID

public static final *RemoteSimulationService.getImageCount_args._Fields* **ID**

6.31.97 RemoteSimulationService.getImageCount_result

public static class **getImageCount_result** implements org.apache.thrift.TBase<*getImageCount_result*, *getImageCount_result._Fi*

Fields

ex

public *UnknownSimulationIdException* **ex**

metaDataMap

public static final java.util.Map<*_Fields*, org.apache.thrift.meta_data.FieldMetaData> **metaDataMap**

success

public int **success**

Constructors

getImageCount_result

public **getImageCount_result** ( )

getImageCount_result

public **getImageCount_result** (int *success*, *UnknownSimulationIdException* *ex*)

getImageCount_result

public **getImageCount_result** (*getImageCount_result* *other*)
Performs a deep copy on *other*.

Methods

clear

public void **clear** ()

compareTo

public int **compareTo** (*getImageCount_result* *other*)

deepCopy

public *getImageCount_result* **deepCopy** ()

equals

public boolean **equals** (java.lang.Object *that*)

equals

public boolean **equals** (*getImageCount_result* *that*)

fieldForId

public *_Fields* **fieldForId** (int *fieldId*)

getEx

public *UnknownSimulationIdException* **getEx** ()

getFieldValue

public java.lang.Object **getFieldValue** (*_Fields* *field*)

getSuccess

public int **getSuccess**()

hashCode

public int **hashCode**()

isSet

public boolean **isSet**(*_Fields field*)
Returns true if field corresponding to fieldID is set (has been assigned a value) and false otherwise

isSetEx

public boolean **isSetEx**()
Returns true if field ex is set (has been assigned a value) and false otherwise

isSetSuccess

public boolean **isSetSuccess**()
Returns true if field success is set (has been assigned a value) and false otherwise

read

public void **read**(org.apache.thrift.protocol.TProtocol *iprot*)

setEx

public *getImageCount_result* **setEx**(*UnknownSimulationIdException ex*)

setExIsSet

public void **setExIsSet**(boolean *value*)

setFieldValue

public void **setFieldValue**(*_Fields field*, java.lang.Object *value*)

setSuccess

public *getImageCount_result* **setSuccess**(int *success*)

setSuccessIsSet

public void **setSuccessIsSet** (boolean *value*)

toString

public java.lang.String **toString** ()

unsetEx

public void **unsetEx** ()

unsetSuccess

public void **unsetSuccess** ()

validate

public void **validate** ()

write

public void **write** (org.apache.thrift.protocol.TProtocol *oprot*)

6.31.98 RemoteSimulationService.getImageCount_result._Fields

public enum **_Fields** implements org.apache.thrift.TFieldIdEnum

The set of fields this struct contains, along with convenience methods for finding and manipulating them.

Enum Constants

EX

public static final *RemoteSimulationService.getImageCount_result._Fields* **EX**

SUCCESS

public static final *RemoteSimulationService.getImageCount_result._Fields* **SUCCESS**

6.31.99 RemoteSimulationService.getLaserJsonName_args

public static class **getLaserJsonName_args** implements org.apache.thrift.TBase<*getLaserJsonName_args*, *getLaserJsonName_a*

Fields

id

public int **id**

metaDataMap

public static final java.util.Map<*_Fields*, org.apache.thrift.meta_data.FieldMetaData> **metaDataMap**

Constructors

getLaserJsonName_args

public **getLaserJsonName_args** ( )

getLaserJsonName_args

public **getLaserJsonName_args** (int *id*)

getLaserJsonName_args

public **getLaserJsonName_args** (*getLaserJsonName_args* *other*)
Performs a deep copy on *other*.

Methods

clear

public void **clear** ( )

compareTo

public int **compareTo** (*getLaserJsonName_args* *other*)

deepCopy

public *getLaserJsonName_args* **deepCopy** ( )

equals

public boolean **equals** (java.lang.Object *that*)

equals

public boolean **equals** (*getLaserJsonName_args that*)

fieldForId

public *_Fields* **fieldForId** (int *fieldId*)

getFieldValue

public java.lang.Object **getFieldValue** (*_Fields field*)

getId

public int **getId** ()

hashCode

public int **hashCode** ()

isSet

public boolean **isSet** (*_Fields field*)

Returns true if field corresponding to fieldID is set (has been assigned a value) and false otherwise

isSetId

public boolean **isSetId** ()

Returns true if field id is set (has been assigned a value) and false otherwise

read

public void **read** (org.apache.thrift.protocol.TProtocol *iprot*)

setFieldValue

public void **setFieldValue** (*_Fields field*, java.lang.Object *value*)

setId

public *getLaserJsonName_args* **setId** (int *id*)

setIdIsSet

public void **setIdIsSet** (boolean *value*)

toString

public java.lang.String **toString** ()

unsetId

public void **unsetId** ()

validate

public void **validate** ()

write

public void **write** (org.apache.thrift.protocol.TProtocol *oprot*)

6.31.100 RemoteSimulationService.getLaserJsonName_args._Fields

public enum **_Fields** implements org.apache.thrift.TFieldIdEnum

The set of fields this struct contains, along with convenience methods for finding and manipulating them.

Enum Constants

ID

public static final *RemoteSimulationService.getLaserJsonName_args._Fields* **ID**

6.31.101 RemoteSimulationService.getLaserJsonName_result

public static class **getLaserJsonName_result** implements org.apache.thrift.TBase<*getLaserJsonName_result*, *getLaserJsonNam*

Fields

ex

public *UnknownSimulationIdException* **ex**

metaDataMap

public static final java.util.Map<*_Fields*, org.apache.thrift.meta_data.FieldMetaData> **metaDataMap**

success

public java.lang.String **success**

Constructors

getLaserJsonName_result

public **getLaserJsonName_result** ()

getLaserJsonName_result

public **getLaserJsonName_result** (java.lang.String *success*, *UnknownSimulationIdException* *ex*)

getLaserJsonName_result

public **getLaserJsonName_result** (*getLaserJsonName_result* *other*)
Performs a deep copy on *other*.

Methods

clear

public void **clear** ()

compareTo

public int **compareTo** (*getLaserJsonName_result* *other*)

deepCopy

public *getLaserJsonName_result* **deepCopy** ()

equals

public boolean **equals** (java.lang.Object *that*)

equals

public boolean **equals** (*getLaserJsonName_result* *that*)

fieldForId

public *_Fields* **fieldForId** (int *fieldId*)

getEx

public *UnknownSimulationIdException* **getEx**()

getFieldValue

public java.lang.Object **getFieldValue**(*_Fields field*)

getSuccess

public java.lang.String **getSuccess**()

hashCode

public int **hashCode**()

isSet

public boolean **isSet**(*_Fields field*)

Returns true if field corresponding to fieldID is set (has been assigned a value) and false otherwise

isSetEx

public boolean **isSetEx**()

Returns true if field ex is set (has been assigned a value) and false otherwise

isSetSuccess

public boolean **isSetSuccess**()

Returns true if field success is set (has been assigned a value) and false otherwise

read

public void **read**(org.apache.thrift.protocol.TProtocol *iprot*)

setEx

public *getLaserJsonName_result* **setEx**(*UnknownSimulationIdException ex*)

setExIsSet

public void **setExIsSet**(boolean *value*)

setFieldValue

public void **setFieldValue** (*_Fields field*, java.lang.Object *value*)

setSuccess

public *getLaserJsonName_result* **setSuccess** (java.lang.String *success*)

setSuccessIsSet

public void **setSuccessIsSet** (boolean *value*)

toString

public java.lang.String **toString** ()

unsetEx

public void **unsetEx** ()

unsetSuccess

public void **unsetSuccess** ()

validate

public void **validate** ()

write

public void **write** (org.apache.thrift.protocol.TProtocol *oprot*)

6.31.102 RemoteSimulationService.getLaserJsonName_result._Fields

public enum **_Fields** implements org.apache.thrift.TFieldIdEnum

The set of fields this struct contains, along with convenience methods for finding and manipulating them.

Enum Constants

EX

public static final *RemoteSimulationService.getLaserJsonName_result._Fields* **EX**

SUCCESS

public static final *RemoteSimulationService.getLaserJsonName_result._Fields* **SUCCESS**

6.31.103 RemoteSimulationService.getNextImage_args

public static class **getNextImage_args** implements org.apache.thrift.TBase<*getNextImage_args*, *getNextImage_args._Fields*>, java

Fields

id

public int **id**

metaDataMap

public static final java.util.Map<*_Fields*, org.apache.thrift.meta_data.FieldMetaData> **metaDataMap**

Constructors

getNextImage_args

public **getNextImage_args** ()

getNextImage_args

public **getNextImage_args** (int *id*)

getNextImage_args

public **getNextImage_args** (*getNextImage_args other*)
Performs a deep copy on *other*.

Methods

clear

public void **clear** ()

compareTo

public int **compareTo** (*getNextImage_args other*)

deepCopy

public *getNextImage_args* **deepCopy**()

equals

public boolean **equals**(java.lang.Object *that*)

equals

public boolean **equals**(*getNextImage_args* *that*)

fieldForId

public *_Fields* **fieldForId**(int *fieldId*)

getFieldValue

public java.lang.Object **getFieldValue**(*_Fields* *field*)

getId

public int **getId**()

hashCode

public int **hashCode**()

isSet

public boolean **isSet**(*_Fields* *field*)

Returns true if field corresponding to fieldID is set (has been assigned a value) and false otherwise

isSetId

public boolean **isSetId**()

Returns true if field id is set (has been assigned a value) and false otherwise

read

public void **read**(org.apache.thrift.protocol.TProtocol *iprot*)

setFieldValue

public void **setFieldValue** (*_Fields field*, java.lang.Object *value*)

setId

public *getNextImage_args* **setId** (int *id*)

setIdIsSet

public void **setIdIsSet** (boolean *value*)

toString

public java.lang.String **toString** ()

unsetId

public void **unsetId** ()

validate

public void **validate** ()

write

public void **write** (org.apache.thrift.protocol.TProtocol *oprot*)

6.31.104 RemoteSimulationService.getNextImage_args._Fields

public enum **_Fields** implements org.apache.thrift.TFieldIdEnum

The set of fields this struct contains, along with convenience methods for finding and manipulating them.

Enum Constants

ID

public static final *RemoteSimulationService.getNextImage_args._Fields* **ID**

6.31.105 RemoteSimulationService.getNextImage_result

public static class **getNextImage_result** implements org.apache.thrift.TBase<*getNextImage_result*, *getNextImage_result._Fields*>

Fields

ex

public *ImageGenerationException* **ex**

ex2

public *UnknownSimulationIdException* **ex2**

metaDataMap

public static final java.util.Map<*_Fields*, org.apache.thrift.meta_data.FieldMetaData> **metaDataMap**

success

public java.nio.ByteBuffer **success**

Constructors

getNextImage_result

public **getNextImage_result**()

getNextImage_result

public **getNextImage_result**(java.nio.ByteBuffer *success*, *ImageGenerationException* *ex*, *UnknownSimulationIdException* *ex2*)

getNextImage_result

public **getNextImage_result**(*getNextImage_result* *other*)

Performs a deep copy on *other*.

Methods

bufferForSuccess

public java.nio.ByteBuffer **bufferForSuccess**()

clear

public void **clear**()

compareTo

public int **compareTo** (*getNextImage_result other*)

deepCopy

public *getNextImage_result* **deepCopy** ()

equals

public boolean **equals** (java.lang.Object *that*)

equals

public boolean **equals** (*getNextImage_result that*)

fieldForId

public *_Fields* **fieldForId** (int *fieldId*)

getEx

public *ImageGenerationException* **getEx** ()

getEx2

public *UnknownSimulationIdException* **getEx2** ()

getFieldValue

public java.lang.Object **getFieldValue** (*_Fields field*)

getSuccess

public byte[] **getSuccess** ()

hashCode

public int **hashCode** ()

isSet

public boolean **isSet** (*_Fields field*)

Returns true if field corresponding to fieldID is set (has been assigned a value) and false otherwise

isSetEx

public boolean **isSetEx**()
Returns true if field ex is set (has been assigned a value) and false otherwise

isSetEx2

public boolean **isSetEx2**()
Returns true if field ex2 is set (has been assigned a value) and false otherwise

isSetSuccess

public boolean **isSetSuccess**()
Returns true if field success is set (has been assigned a value) and false otherwise

read

public void **read**(org.apache.thrift.protocol.TProtocol *iprot*)

setEx

public *getNextImage_result* **setEx**(*ImageGenerationException ex*)

setEx2

public *getNextImage_result* **setEx2**(*UnknownSimulationIdException ex2*)

setEx2IsSet

public void **setEx2IsSet**(boolean *value*)

setExIsSet

public void **setExIsSet**(boolean *value*)

setFieldValue

public void **setFieldValue**(*_Fields field*, java.lang.Object *value*)

setSuccess

public *getNextImage_result* **setSuccess**(byte[] *success*)

setSuccess

public *getNextImage_result* **setSuccess** (java.nio.ByteBuffer *success*)

setSuccessIsSet

public void **setSuccessIsSet** (boolean *value*)

toString

public java.lang.String **toString** ()

unsetEx

public void **unsetEx** ()

unsetEx2

public void **unsetEx2** ()

unsetSuccess

public void **unsetSuccess** ()

validate

public void **validate** ()

write

public void **write** (org.apache.thrift.protocol.TProtocol *oprot*)

6.31.106 RemoteSimulationService.getNextImage_result._Fields

public enum **_Fields** implements org.apache.thrift.TFieldIdEnum

The set of fields this struct contains, along with convenience methods for finding and manipulating them.

Enum Constants

EX

public static final *RemoteSimulationService.getNextImage_result._Fields* **EX**

EX2

public static final *RemoteSimulationService.getNextImage_result._Fields* **EX2**

SUCCESS

public static final *RemoteSimulationService.getNextImage_result._Fields* **SUCCESS**

6.31.107 RemoteSimulationService.getObjectSpacePixelSize_args

public static class **getObjectSpacePixelSize_args** implements org.apache.thrift.TBase<*getObjectSpacePixelSize_args*, *getOb*

Fields

id

public int **id**

metaDataMap

public static final java.util.Map<*_Fields*, org.apache.thrift.meta_data.FieldMetaData> **metaDataMap**

Constructors

getObjectSpacePixelSize_args

public **getObjectSpacePixelSize_args** ( )

getObjectSpacePixelSize_args

public **getObjectSpacePixelSize_args** (int *id*)

getObjectSpacePixelSize_args

public **getObjectSpacePixelSize_args** (*getObjectSpacePixelSize_args other*)
Performs a deep copy on *other*.

Methods

clear

public void **clear** ( )

compareTo

public int **compareTo** (*getObjectSpacePixelSize_args other*)

deepCopy

public *getObjectSpacePixelSize_args* **deepCopy** ()

equals

public boolean **equals** (java.lang.Object *that*)

equals

public boolean **equals** (*getObjectSpacePixelSize_args that*)

fieldForId

public *_Fields* **fieldForId** (int *fieldId*)

getFieldValue

public java.lang.Object **getFieldValue** (*_Fields field*)

getId

public int **getId** ()

hashCode

public int **hashCode** ()

isSet

public boolean **isSet** (*_Fields field*)

Returns true if field corresponding to fieldID is set (has been assigned a value) and false otherwise

isSetId

public boolean **isSetId** ()

Returns true if field id is set (has been assigned a value) and false otherwise

read

public void **read**(org.apache.thrift.protocol.TProtocol *iprot*)

setFieldValue

public void **setFieldValue**(*_Fields field*, java.lang.Object *value*)

setId

public *getObjectSpacePixelSize_args* **setId**(int *id*)

setIdIsSet

public void **setIdIsSet**(boolean *value*)

toString

public java.lang.String **toString**()

unsetId

public void **unsetId**()

validate

public void **validate**()

write

public void **write**(org.apache.thrift.protocol.TProtocol *oprot*)

6.31.108 RemoteSimulationService.getObjectSpacePixelSize_args._Fields

public enum **_Fields** implements org.apache.thrift.TFieldIdEnum

The set of fields this struct contains, along with convenience methods for finding and manipulating them.

Enum Constants

ID

public static final *RemoteSimulationService.getObjectSpacePixelSize_args._Fields* **ID**

6.31.109 RemoteSimulationService.getObjectSpacePixelSize_result

public static class **getObjectSpacePixelSize_result** implements org.apache.thrift.TBase<*getObjectSpacePixelSize_result*, *g*

Fields

ex

public *UnknownSimulationIdException* **ex**

metaDataMap

public static final java.util.Map<*_Fields*, org.apache.thrift.meta_data.FieldMetaData> **metaDataMap**

success

public double **success**

Constructors

getObjectSpacePixelSize_result

public **getObjectSpacePixelSize_result** ()

getObjectSpacePixelSize_result

public **getObjectSpacePixelSize_result** (double *success*, *UnknownSimulationIdException* *ex*)

getObjectSpacePixelSize_result

public **getObjectSpacePixelSize_result** (*getObjectSpacePixelSize_result* *other*)
Performs a deep copy on *other*.

Methods

clear

public void **clear** ()

compareTo

public int **compareTo** (*getObjectSpacePixelSize_result* *other*)

deepCopy

public *getObjectSpacePixelSize_result* **deepCopy**()

equals

public boolean **equals**(java.lang.Object *that*)

equals

public boolean **equals**(*getObjectSpacePixelSize_result* *that*)

fieldForId

public *_Fields* **fieldForId**(int *fieldId*)

getEx

public *UnknownSimulationIdException* **getEx**()

getFieldValue

public java.lang.Object **getFieldValue**(*_Fields* *field*)

getSuccess

public double **getSuccess**()

hashCode

public int **hashCode**()

isSet

public boolean **isSet**(*_Fields* *field*)

Returns true if field corresponding to fieldID is set (has been assigned a value) and false otherwise

isSetEx

public boolean **isSetEx**()

Returns true if field ex is set (has been assigned a value) and false otherwise

isSetSuccess

public boolean **isSetSuccess**()

Returns true if field success is set (has been assigned a value) and false otherwise

read

public void **read**(org.apache.thrift.protocol.TProtocol *iprot*)

setEx

public *getObjectSpacePixelSize_result* **setEx**(*UnknownSimulationIdException ex*)

setExIsSet

public void **setExIsSet**(boolean *value*)

setFieldValue

public void **setFieldValue**(*_Fields field*, java.lang.Object *value*)

setSuccess

public *getObjectSpacePixelSize_result* **setSuccess**(double *success*)

setSuccessIsSet

public void **setSuccessIsSet**(boolean *value*)

toString

public java.lang.String **toString**()

unsetEx

public void **unsetEx**()

unsetSuccess

public void **unsetSuccess**()

validate

public void **validate**()

write

public void **write** (org.apache.thrift.protocol.TProtocol *oprot*)

6.31.110 RemoteSimulationService.getObjectSpacePixelSize_result._Fields

public enum **_Fields** implements org.apache.thrift.TFieldIdEnum
The set of fields this struct contains, along with convenience methods for finding and manipulating them.

Enum Constants

EX

public static final *RemoteSimulationService.getObjectSpacePixelSize_result._Fields* **EX**

SUCCESS

public static final *RemoteSimulationService.getObjectSpacePixelSize_result._Fields* **SUCCESS**

6.31.111 RemoteSimulationService.getObjectiveJsonName_args

public static class **getObjectiveJsonName_args** implements org.apache.thrift.TBase<*getObjectiveJsonName_args*, *getObjective*

Fields

id

public int **id**

metaDataMap

public static final java.util.Map<*_Fields*, org.apache.thrift.meta_data.FieldMetaData> **metaDataMap**

Constructors

getObjectiveJsonName_args

public **getObjectiveJsonName_args** ()

getObjectiveJsonName_args

public **getObjectiveJsonName_args** (int *id*)

getObjectiveJsonName_args

public **getObjectiveJsonName_args** (*getObjectiveJsonName_args other*)
Performs a deep copy on *other*.

Methods

clear

public void **clear** ()

compareTo

public int **compareTo** (*getObjectiveJsonName_args other*)

deepCopy

public *getObjectiveJsonName_args* **deepCopy** ()

equals

public boolean **equals** (java.lang.Object *that*)

equals

public boolean **equals** (*getObjectiveJsonName_args that*)

fieldForId

public *_Fields* **fieldForId** (int *fieldId*)

getFieldValue

public java.lang.Object **getFieldValue** (*_Fields field*)

getId

public int **getId** ()

hashCode

public int **hashCode** ()

isSet

public boolean **isSet** (*_Fields field*)

Returns true if field corresponding to fieldID is set (has been assigned a value) and false otherwise

isSetId

public boolean **isSetId** ()

Returns true if field id is set (has been assigned a value) and false otherwise

read

public void **read** (org.apache.thrift.protocol.TProtocol *iprot*)

setFieldValue

public void **setFieldValue** (*_Fields field*, java.lang.Object *value*)

setId

public *getObjectiveJsonName_args* **setId** (int *id*)

setIdIsSet

public void **setIdIsSet** (boolean *value*)

toString

public java.lang.String **toString** ()

unsetId

public void **unsetId** ()

validate

public void **validate** ()

write

public void **write** (org.apache.thrift.protocol.TProtocol *oprot*)

6.31.112 RemoteSimulationService.getObjectiveJsonName_args._Fields

public enum **_Fields** implements org.apache.thrift.TFieldIdEnum

The set of fields this struct contains, along with convenience methods for finding and manipulating them.

Enum Constants

ID

public static final *RemoteSimulationService.getObjectiveJsonName_args._Fields* **ID**

6.31.113 RemoteSimulationService.getObjectiveJsonName_result

public static class **getObjectiveJsonName_result** implements org.apache.thrift.TBase<*getObjectiveJsonName_result*, *getObje*

Fields

ex

public *UnknownSimulationIdException* **ex**

metaDataMap

public static final java.util.Map<*_Fields*, org.apache.thrift.meta_data.FieldMetaData> **metaDataMap**

success

public java.lang.String **success**

Constructors

getObjectiveJsonName_result

public **getObjectiveJsonName_result**()

getObjectiveJsonName_result

public **getObjectiveJsonName_result**(java.lang.String *success*, *UnknownSimulationIdException* *ex*)

getObjectiveJsonName_result

public **getObjectiveJsonName_result**(*getObjectiveJsonName_result* *other*)

Performs a deep copy on *other*.

Methods

clear

public void **clear**()

compareTo

public int **compareTo**(*getObjectiveJsonName_result other*)

deepCopy

public *getObjectiveJsonName_result* **deepCopy**()

equals

public boolean **equals**(java.lang.Object *that*)

equals

public boolean **equals**(*getObjectiveJsonName_result that*)

fieldForId

public *_Fields* **fieldForId**(int *fieldId*)

getEx

public *UnknownSimulationIdException* **getEx**()

getFieldValue

public java.lang.Object **getFieldValue**(*_Fields field*)

getSuccess

public java.lang.String **getSuccess**()

hashCode

public int **hashCode**()

isSet

public boolean **isSet** (*_Fields field*)

Returns true if field corresponding to fieldID is set (has been assigned a value) and false otherwise

isSetEx

public boolean **isSetEx** ()

Returns true if field ex is set (has been assigned a value) and false otherwise

isSetSuccess

public boolean **isSetSuccess** ()

Returns true if field success is set (has been assigned a value) and false otherwise

read

public void **read** (org.apache.thrift.protocol.TProtocol *iprot*)

setEx

public *getObjectiveJsonName_result* **setEx** (*UnknownSimulationIdException ex*)

setExIsSet

public void **setExIsSet** (boolean *value*)

setFieldValue

public void **setFieldValue** (*_Fields field*, java.lang.Object *value*)

setSuccess

public *getObjectiveJsonName_result* **setSuccess** (java.lang.String *success*)

setSuccessIsSet

public void **setSuccessIsSet** (boolean *value*)

toString

public java.lang.String **toString** ()

unsetEx

public void **unsetEx**()

unsetSuccess

public void **unsetSuccess**()

validate

public void **validate**()

write

public void **write**(org.apache.thrift.protocol.TProtocol *oprot*)

6.31.114 RemoteSimulationService.getObjectiveJsonName_result._Fields

public enum **_Fields** implements org.apache.thrift.TFieldIdEnum

The set of fields this struct contains, along with convenience methods for finding and manipulating them.

Enum Constants

EX

public static final *RemoteSimulationService.getObjectiveJsonName_result._Fields* **EX**

SUCCESS

public static final *RemoteSimulationService.getObjectiveJsonName_result._Fields* **SUCCESS**

6.31.115 RemoteSimulationService.getServerStatus_args

public static class **getServerStatus_args** implements org.apache.thrift.TBase<*getServerStatus_args*, *getServerStatus_args._Fiel*

Fields

metaDataMap

public static final java.util.Map<*_Fields*, org.apache.thrift.meta_data.FieldMetaData> **metaDataMap**

Constructors

getServerStatus_args

public **getServerStatus_args**()

getServerStatus_args

public **getServerStatus_args**(*getServerStatus_args other*)
Performs a deep copy on *other*.

Methods

clear

public void **clear**()

compareTo

public int **compareTo**(*getServerStatus_args other*)

deepCopy

public *getServerStatus_args* **deepCopy**()

equals

public boolean **equals**(java.lang.Object *that*)

equals

public boolean **equals**(*getServerStatus_args that*)

fieldForId

public *_Fields* **fieldForId**(int *fieldId*)

getFieldValue

public java.lang.Object **getFieldValue**(*_Fields field*)

hashCode

public int **hashCode**()

isSet

public boolean **isSet** (*_Fields field*)
Returns true if field corresponding to fieldID is set (has been assigned a value) and false otherwise

read

public void **read** (org.apache.thrift.protocol.TProtocol *iprot*)

setFieldValue

public void **setFieldValue** (*_Fields field*, java.lang.Object *value*)

toString

public java.lang.String **toString** ()

validate

public void **validate** ()

write

public void **write** (org.apache.thrift.protocol.TProtocol *oprot*)

6.31.116 RemoteSimulationService.getServerStatus_args._Fields

public enum **_Fields** implements org.apache.thrift.TFieldIdEnum
The set of fields this struct contains, along with convenience methods for finding and manipulating them.

Enum Constants

6.31.117 RemoteSimulationService.getServerStatus_result

public static class **getServerStatus_result** implements org.apache.thrift.TBase<*getServerStatus_result*, *getServerStatus_result*

Fields

metaDataMap

public static final java.util.Map<*_Fields*, org.apache.thrift.meta_data.FieldMetaData> **metaDataMap**

success

public java.lang.String **success**

Constructors

getServerStatus_result

public **getServerStatus_result** ()

getServerStatus_result

public **getServerStatus_result** (java.lang.String *success*)

getServerStatus_result

public **getServerStatus_result** (*getServerStatus_result other*)
Performs a deep copy on *other*.

Methods

clear

public void **clear** ()

compareTo

public int **compareTo** (*getServerStatus_result other*)

deepCopy

public *getServerStatus_result* **deepCopy** ()

equals

public boolean **equals** (java.lang.Object *that*)

equals

public boolean **equals** (*getServerStatus_result that*)

fieldForId

public *_Fields* **fieldForId** (int *fieldId*)

getFieldValue

public java.lang.Object **getFieldValue** (*_Fields field*)

getSuccess

public java.lang.String **getSuccess**()

hashCode

public int **hashCode**()

isSet

public boolean **isSet**(*_Fields field*)

Returns true if field corresponding to fieldID is set (has been assigned a value) and false otherwise

isSetSuccess

public boolean **isSetSuccess**()

Returns true if field success is set (has been assigned a value) and false otherwise

read

public void **read**(org.apache.thrift.protocol.TProtocol *iprot*)

setFieldValue

public void **setFieldValue**(*_Fields field*, java.lang.Object *value*)

setSuccess

public *getServerStatus_result* **setSuccess**(java.lang.String *success*)

setSuccessIsSet

public void **setSuccessIsSet**(boolean *value*)

toString

public java.lang.String **toString**()

unsetSuccess

public void **unsetSuccess**()

validate

public void **validate** ( )

write

public void **write** (org.apache.thrift.protocol.TProtocol *oprot*)

6.31.118 RemoteSimulationService.getServerStatus_result._Fields

public enum **_Fields** implements org.apache.thrift.TFieldIdEnum
The set of fields this struct contains, along with convenience methods for finding and manipulating them.

Enum Constants

SUCCESS

public static final *RemoteSimulationService.getServerStatus_result._Fields* **SUCCESS**

6.31.119 RemoteSimulationService.getShortTrueSignalDescription_args

public static class **getShortTrueSignalDescription_args** implements org.apache.thrift.TBase<*getShortTrueSignalDescript*

Fields

id

public int **id**

metaDataMap

public static final java.util.Map<*_Fields*, org.apache.thrift.meta_data.FieldMetaData> **metaDataMap**

Constructors

getShortTrueSignalDescription_args

public **getShortTrueSignalDescription_args** ( )

getShortTrueSignalDescription_args

public **getShortTrueSignalDescription_args** (int *id*)

getShortTrueSignalDescription_args

public **getShortTrueSignalDescription_args** (*getShortTrueSignalDescription_args other*)
Performs a deep copy on *other*.

Methods

clear

public void **clear** ()

compareTo

public int **compareTo** (*getShortTrueSignalDescription_args other*)

deepCopy

public *getShortTrueSignalDescription_args* **deepCopy** ()

equals

public boolean **equals** (java.lang.Object *that*)

equals

public boolean **equals** (*getShortTrueSignalDescription_args that*)

fieldForId

public *_Fields* **fieldForId** (int *fieldId*)

getFieldValue

public java.lang.Object **getFieldValue** (*_Fields field*)

getId

public int **getId** ()

hashCode

public int **hashCode** ()

isSet

public boolean **isSet** (*_Fields field*)

Returns true if field corresponding to fieldID is set (has been assigned a value) and false otherwise

isSetId

public boolean **isSetId** ()

Returns true if field id is set (has been assigned a value) and false otherwise

read

public void **read** (org.apache.thrift.protocol.TProtocol *iprot*)

setFieldValue

public void **setFieldValue** (*_Fields field*, java.lang.Object *value*)

setId

public *getShortTrueSignalDescription_args* **setId** (int *id*)

setIdIsSet

public void **setIdIsSet** (boolean *value*)

toString

public java.lang.String **toString** ()

unsetId

public void **unsetId** ()

validate

public void **validate** ()

write

public void **write** (org.apache.thrift.protocol.TProtocol *oprot*)

6.31.120 RemoteSimulationService.getShortTrueSignalDescription_args._Fields

public enum **_Fields** implements org.apache.thrift.TFieldIdEnum

The set of fields this struct contains, along with convenience methods for finding and manipulating them.

Enum Constants

ID

public static final *RemoteSimulationService.getShortTrueSignalDescription_args._Fields* **ID**

6.31.121 RemoteSimulationService.getShortTrueSignalDescription_result

public static class **getShortTrueSignalDescription_result** implements org.apache.thrift.TBase<*getShortTrueSignalDescr*

Fields

ex

public *UnknownSimulationIdException* **ex**

metaDataMap

public static final java.util.Map<*_Fields*, org.apache.thrift.meta_data.FieldMetaData> **metaDataMap**

success

public java.lang.String **success**

Constructors

getShortTrueSignalDescription_result

public **getShortTrueSignalDescription_result** ()

getShortTrueSignalDescription_result

public **getShortTrueSignalDescription_result** (java.lang.String *success*, *UnknownSimulationIdException ex*)

getShortTrueSignalDescription_result

public **getShortTrueSignalDescription_result** (*getShortTrueSignalDescription_result other*)

Performs a deep copy on *other*.

Methods

clear

public void **clear**()

compareTo

public int **compareTo**(*getShortTrueSignalDescription_result other*)

deepCopy

public *getShortTrueSignalDescription_result* **deepCopy**()

equals

public boolean **equals**(java.lang.Object *that*)

equals

public boolean **equals**(*getShortTrueSignalDescription_result that*)

fieldForId

public *_Fields* **fieldForId**(int *fieldId*)

getEx

public *UnknownSimulationIdException* **getEx**()

getFieldValue

public java.lang.Object **getFieldValue**(*_Fields field*)

getSuccess

public java.lang.String **getSuccess**()

hashCode

public int **hashCode**()

isSet

public boolean **isSet** (*_Fields field*)

Returns true if field corresponding to fieldID is set (has been assigned a value) and false otherwise

isSetEx

public boolean **isSetEx** ()

Returns true if field ex is set (has been assigned a value) and false otherwise

isSetSuccess

public boolean **isSetSuccess** ()

Returns true if field success is set (has been assigned a value) and false otherwise

read

public void **read** (org.apache.thrift.protocol.TProtocol *iprot*)

setEx

public *getShortTrueSignalDescription_result* **setEx** (*UnknownSimulationIdException ex*)

setExIsSet

public void **setExIsSet** (boolean *value*)

setFieldValue

public void **setFieldValue** (*_Fields field*, java.lang.Object *value*)

setSuccess

public *getShortTrueSignalDescription_result* **setSuccess** (java.lang.String *success*)

setSuccessIsSet

public void **setSuccessIsSet** (boolean *value*)

toString

public java.lang.String **toString** ()

unsetEx

public void **unsetEx**()

unsetSuccess

public void **unsetSuccess**()

validate

public void **validate**()

write

public void **write**(org.apache.thrift.protocol.TProtocol *oprot*)

6.31.122 RemoteSimulationService.getShortTrueSignalDescription_result._Fields

public enum **_Fields** implements org.apache.thrift.TFieldIdEnum
The set of fields this struct contains, along with convenience methods for finding and manipulating them.

Enum Constants

EX

public static final *RemoteSimulationService.getShortTrueSignalDescription_result._Fields* **EX**

SUCCESS

public static final *RemoteSimulationService.getShortTrueSignalDescription_result._Fields* **SUCCESS**

6.31.123 RemoteSimulationService.getStageJsonName_args

public static class **getStageJsonName_args** implements org.apache.thrift.TBase<*getStageJsonName_args*, *getStageJsonName_ar*

Fields

id

public int **id**

metaDataMap

public static final java.util.Map<*_Fields*, org.apache.thrift.meta_data.FieldMetaData> **metaDataMap**

Constructors

getStageJsonName_args

public **getStageJsonName_args** ( )

getStageJsonName_args

public **getStageJsonName_args** (int *id*)

getStageJsonName_args

public **getStageJsonName_args** (*getStageJsonName_args other*)
Performs a deep copy on *other*.

Methods

clear

public void **clear** ( )

compareTo

public int **compareTo** (*getStageJsonName_args other*)

deepCopy

public *getStageJsonName_args* **deepCopy** ( )

equals

public boolean **equals** (java.lang.Object *that*)

equals

public boolean **equals** (*getStageJsonName_args that*)

fieldForId

public *_Fields* **fieldForId** (int *fieldId*)

getFieldValue

public java.lang.Object **getFieldValue** (*_Fields field*)

getId

public int **getId**()

hashCode

public int **hashCode**()

isSet

public boolean **isSet**(*_Fields field*)

Returns true if field corresponding to fieldID is set (has been assigned a value) and false otherwise

isSetId

public boolean **isSetId**()

Returns true if field id is set (has been assigned a value) and false otherwise

read

public void **read**(org.apache.thrift.protocol.TProtocol *iprot*)

setFieldValue

public void **setFieldValue**(*_Fields field*, java.lang.Object *value*)

setId

public *getStageJsonName_args* **setId**(int *id*)

setIdIsSet

public void **setIdIsSet**(boolean *value*)

toString

public java.lang.String **toString**()

unsetId

public void **unsetId**()

validate

public void **validate** ()

write

public void **write** (org.apache.thrift.protocol.TProtocol *oprot*)

6.31.124 RemoteSimulationService.getStageJsonName_args._Fields

public enum **_Fields** implements org.apache.thrift.TFieldIdEnum
The set of fields this struct contains, along with convenience methods for finding and manipulating them.

Enum Constants

ID

public static final *RemoteSimulationService.getStageJsonName_args._Fields* **ID**

6.31.125 RemoteSimulationService.getStageJsonName_result

public static class **getStageJsonName_result** implements org.apache.thrift.TBase<*getStageJsonName_result*, *getStageJsonNam*

Fields

ex

public *UnknownSimulationIdException* **ex**

metaDataMap

public static final java.util.Map<*_Fields*, org.apache.thrift.meta_data.FieldMetaData> **metaDataMap**

success

public java.lang.String **success**

Constructors

getStageJsonName_result

public **getStageJsonName_result** ()

getStageJsonName_result

public **getStageJsonName_result** (java.lang.String *success*, *UnknownSimulationIdException* *ex*)

getStageJsonName_result

public **getStageJsonName_result** (*getStageJsonName_result* *other*)
Performs a deep copy on *other*.

Methods

clear

public void **clear** ()

compareTo

public int **compareTo** (*getStageJsonName_result* *other*)

deepCopy

public *getStageJsonName_result* **deepCopy** ()

equals

public boolean **equals** (java.lang.Object *that*)

equals

public boolean **equals** (*getStageJsonName_result* *that*)

fieldForId

public *_Fields* **fieldForId** (int *fieldId*)

getEx

public *UnknownSimulationIdException* **getEx** ()

getFieldValue

public java.lang.Object **getFieldValue** (*_Fields* *field*)

getSuccess

public java.lang.String **getSuccess**()

hashCode

public int **hashCode**()

isSet

public boolean **isSet**(*_Fields field*)

Returns true if field corresponding to fieldID is set (has been assigned a value) and false otherwise

isSetEx

public boolean **isSetEx**()

Returns true if field ex is set (has been assigned a value) and false otherwise

isSetSuccess

public boolean **isSetSuccess**()

Returns true if field success is set (has been assigned a value) and false otherwise

read

public void **read**(org.apache.thrift.protocol.TProtocol *iprot*)

setEx

public *getStageJsonName_result* **setEx**(*UnknownSimulationIdException ex*)

setExIsSet

public void **setExIsSet**(boolean *value*)

setFieldValue

public void **setFieldValue**(*_Fields field*, java.lang.Object *value*)

setSuccess

public *getStageJsonName_result* **setSuccess**(java.lang.String *success*)

setSuccessIsSet

public void **setSuccessIsSet** (boolean *value*)

toString

public java.lang.String **toString** ()

unsetEx

public void **unsetEx** ()

unsetSuccess

public void **unsetSuccess** ()

validate

public void **validate** ()

write

public void **write** (org.apache.thrift.protocol.TProtocol *oprot*)

6.31.126 RemoteSimulationService.getStageJsonName_result._Fields

public enum **_Fields** implements org.apache.thrift.TFieldIdEnum

The set of fields this struct contains, along with convenience methods for finding and manipulating them.

Enum Constants

EX

public static final *RemoteSimulationService.getStageJsonName_result._Fields* **EX**

SUCCESS

public static final *RemoteSimulationService.getStageJsonName_result._Fields* **SUCCESS**

6.31.127 RemoteSimulationService.getTrueSignal_args

public static class **getTrueSignal_args** implements org.apache.thrift.TBase<*getTrueSignal_args*, *getTrueSignal_args._Fields*>, ja

Fields

id

public int **id**

imageNum

public int **imageNum**

metaDataMap

public static final java.util.Map<*_Fields*, org.apache.thrift.meta_data.FieldMetaData> **metaDataMap**

Constructors

getTrueSignal_args

public **getTrueSignal_args**()

getTrueSignal_args

public **getTrueSignal_args**(int *id*, int *imageNum*)

getTrueSignal_args

public **getTrueSignal_args**(*getTrueSignal_args* *other*)
Performs a deep copy on *other*.

Methods

clear

public void **clear**()

compareTo

public int **compareTo**(*getTrueSignal_args* *other*)

deepCopy

public *getTrueSignal_args* **deepCopy**()

equals

public boolean **equals** (java.lang.Object *that*)

equals

public boolean **equals** (*getTrueSignal_args* *that*)

fieldForId

public *_Fields* **fieldForId** (int *fieldId*)

getFieldValue

public java.lang.Object **getFieldValue** (*_Fields* *field*)

getId

public int **getId** ()

getImageNum

public int **getImageNum** ()

hashCode

public int **hashCode** ()

isSet

public boolean **isSet** (*_Fields* *field*)

Returns true if field corresponding to fieldID is set (has been assigned a value) and false otherwise

isSetId

public boolean **isSetId** ()

Returns true if field id is set (has been assigned a value) and false otherwise

isSetImageNum

public boolean **isSetImageNum** ()

Returns true if field imageNum is set (has been assigned a value) and false otherwise

read

public void **read** (org.apache.thrift.protocol.TProtocol *iprot*)

setFieldValue

public void **setFieldValue** (*_Fields field*, java.lang.Object *value*)

setId

public *getTrueSignal_args* **setId** (int *id*)

setIdIsSet

public void **setIdIsSet** (boolean *value*)

setImageNum

public *getTrueSignal_args* **setImageNum** (int *imageNum*)

setImageNumIsSet

public void **setImageNumIsSet** (boolean *value*)

toString

public java.lang.String **toString** ()

unsetId

public void **unsetId** ()

unsetImageNum

public void **unsetImageNum** ()

validate

public void **validate** ()

write

public void **write** (org.apache.thrift.protocol.TProtocol *oprot*)

6.31.128 RemoteSimulationService.getTrueSignal_args._Fields

public enum **_Fields** implements org.apache.thrift.TFieldIdEnum

The set of fields this struct contains, along with convenience methods for finding and manipulating them.

Enum Constants

ID

public static final *RemoteSimulationService.getTrueSignal_args._Fields* **ID**

IMAGE_NUM

public static final *RemoteSimulationService.getTrueSignal_args._Fields* **IMAGE_NUM**

6.31.129 RemoteSimulationService.getTrueSignal_result

public static class **getTrueSignal_result** implements org.apache.thrift.TBase<*getTrueSignal_result*, *getTrueSignal_result._Field*

Fields

ex

public *UnknownSimulationIdException* **ex**

metaDataMap

public static final java.util.Map<*_Fields*, org.apache.thrift.meta_data.FieldMetaData> **metaDataMap**

success

public double **success**

Constructors

getTrueSignal_result

public **getTrueSignal_result**()

getTrueSignal_result

public **getTrueSignal_result**(double *success*, *UnknownSimulationIdException* *ex*)

getTrueSignal_result

public **getTrueSignal_result** (*getTrueSignal_result other*)
Performs a deep copy on *other*.

Methods

clear

public void **clear** ()

compareTo

public int **compareTo** (*getTrueSignal_result other*)

deepCopy

public *getTrueSignal_result* **deepCopy** ()

equals

public boolean **equals** (java.lang.Object *that*)

equals

public boolean **equals** (*getTrueSignal_result that*)

fieldForId

public *_Fields* **fieldForId** (int *fieldId*)

getEx

public *UnknownSimulationIdException* **getEx** ()

getFieldValue

public java.lang.Object **getFieldValue** (*_Fields field*)

getSuccess

public double **getSuccess** ()

hashCode

public int **hashCode**()

isSet

public boolean **isSet**(*_Fields field*)
Returns true if field corresponding to fieldID is set (has been assigned a value) and false otherwise

isSetEx

public boolean **isSetEx**()
Returns true if field ex is set (has been assigned a value) and false otherwise

isSetSuccess

public boolean **isSetSuccess**()
Returns true if field success is set (has been assigned a value) and false otherwise

read

public void **read**(org.apache.thrift.protocol.TProtocol *iprot*)

setEx

public *getTrueSignal_result* **setEx**(*UnknownSimulationIdException ex*)

setExIsSet

public void **setExIsSet**(boolean *value*)

setFieldValue

public void **setFieldValue**(*_Fields field*, java.lang.Object *value*)

setSuccess

public *getTrueSignal_result* **setSuccess**(double *success*)

setSuccessIsSet

public void **setSuccessIsSet**(boolean *value*)

toString

public java.lang.String **toString**()

unsetEx

public void **unsetEx**()

unsetSuccess

public void **unsetSuccess**()

validate

public void **validate**()

write

public void **write**(org.apache.thrift.protocol.TProtocol *oprot*)

6.31.130 RemoteSimulationService.getTrueSignal_result._Fields

public enum **_Fields** implements org.apache.thrift.TFieldIdEnum

The set of fields this struct contains, along with convenience methods for finding and manipulating them.

Enum Constants

EX

public static final *RemoteSimulationService.getTrueSignal_result._Fields* **EX**

SUCCESS

public static final *RemoteSimulationService.getTrueSignal_result._Fields* **SUCCESS**

6.31.131 RemoteSimulationService.incrementTimeStep_args

public static class **incrementTimeStep_args** implements org.apache.thrift.TBase<*incrementTimeStep_args*, *incrementTimeStep_*

Fields

id

public int **id**

metaDataMap

public static final java.util.Map<*_Fields*, org.apache.thrift.meta_data.FieldMetaData> **metaDataMap**

Constructors

incrementTimeStep_args

public **incrementTimeStep_args** ( )

incrementTimeStep_args

public **incrementTimeStep_args** (int *id*)

incrementTimeStep_args

public **incrementTimeStep_args** (*incrementTimeStep_args other*)
Performs a deep copy on *other*.

Methods

clear

public void **clear** ( )

compareTo

public int **compareTo** (*incrementTimeStep_args other*)

deepCopy

public *incrementTimeStep_args* **deepCopy** ( )

equals

public boolean **equals** (java.lang.Object *that*)

equals

public boolean **equals** (*incrementTimeStep_args that*)

fieldForId

public *_Fields* **fieldForId** (int *fieldId*)

getFieldValue

public java.lang.Object **getFieldValue** (*_Fields field*)

getId

public int **getId** ()

hashCode

public int **hashCode** ()

isSet

public boolean **isSet** (*_Fields field*)

Returns true if field corresponding to fieldID is set (has been assigned a value) and false otherwise

isSetId

public boolean **isSetId** ()

Returns true if field id is set (has been assigned a value) and false otherwise

read

public void **read** (org.apache.thrift.protocol.TProtocol *iprot*)

setFieldValue

public void **setFieldValue** (*_Fields field*, java.lang.Object *value*)

setId

public *incrementTimeStep_args* **setId** (int *id*)

setIdIsSet

public void **setIdIsSet** (boolean *value*)

toString

public java.lang.String **toString** ()

unsetId

public void **unsetId**()

validate

public void **validate**()

write

public void **write**(org.apache.thrift.protocol.TProtocol *oprot*)

6.31.132 RemoteSimulationService.incrementTimeStep_args._Fields

public enum **_Fields** implements org.apache.thrift.TFieldIdEnum

The set of fields this struct contains, along with convenience methods for finding and manipulating them.

Enum Constants

ID

public static final *RemoteSimulationService.incrementTimeStep_args._Fields* **ID**

6.31.133 RemoteSimulationService.incrementTimeStep_result

public static class **incrementTimeStep_result** implements org.apache.thrift.TBase<*incrementTimeStep_result*, *incrementTimeS*

Fields

ex

public *UnknownSimulationIdException* **ex**

metaDataMap

public static final java.util.Map<*_Fields*, org.apache.thrift.meta_data.FieldMetaData> **metaDataMap**

Constructors

incrementTimeStep_result

public **incrementTimeStep_result**()

incrementTimeStep_result

public **incrementTimeStep_result** (*UnknownSimulationIdException ex*)

incrementTimeStep_result

public **incrementTimeStep_result** (*incrementTimeStep_result other*)
Performs a deep copy on *other*.

Methods

clear

public void **clear** ()

compareTo

public int **compareTo** (*incrementTimeStep_result other*)

deepCopy

public *incrementTimeStep_result* **deepCopy** ()

equals

public boolean **equals** (java.lang.Object *that*)

equals

public boolean **equals** (*incrementTimeStep_result that*)

fieldForId

public *_Fields* **fieldForId** (int *fieldId*)

getEx

public *UnknownSimulationIdException* **getEx** ()

getFieldValue

public java.lang.Object **getFieldValue** (*_Fields field*)

hashCode

public int **hashCode**()

isSet

public boolean **isSet**(*_Fields field*)
Returns true if field corresponding to fieldID is set (has been assigned a value) and false otherwise

isSetEx

public boolean **isSetEx**()
Returns true if field ex is set (has been assigned a value) and false otherwise

read

public void **read**(org.apache.thrift.protocol.TProtocol *iprot*)

setEx

public *incrementTimeStep_result* **setEx**(*UnknownSimulationIdException ex*)

setExIsSet

public void **setExIsSet**(boolean *value*)

setFieldValue

public void **setFieldValue**(*_Fields field*, java.lang.Object *value*)

toString

public java.lang.String **toString**()

unsetEx

public void **unsetEx**()

validate

public void **validate**()

write

public void **write** (org.apache.thrift.protocol.TProtocol *oprot*)

6.31.134 RemoteSimulationService.incrementTimeStep_result._Fields

public enum **_Fields** implements org.apache.thrift.TFieldIdEnum

The set of fields this struct contains, along with convenience methods for finding and manipulating them.

Enum Constants

EX

public static final *RemoteSimulationService.incrementTimeStep_result._Fields* **EX**

6.31.135 RemoteSimulationService.setControlSignal_args

public static class **setControlSignal_args** implements org.apache.thrift.TBase<*setControlSignal_args*, *setControlSignal_args._*

Fields

id

public int **id**

metaDataMap

public static final java.util.Map<*_Fields*, org.apache.thrift.meta_data.FieldMetaData> **metaDataMap**

power

public double **power**

Constructors

setControlSignal_args

public **setControlSignal_args** ()

setControlSignal_args

public **setControlSignal_args** (int *id*, double *power*)

setControlSignal_args

public **setControlSignal_args** (*setControlSignal_args other*)
Performs a deep copy on *other*.

Methods

clear

public void **clear** ()

compareTo

public int **compareTo** (*setControlSignal_args other*)

deepCopy

public *setControlSignal_args* **deepCopy** ()

equals

public boolean **equals** (java.lang.Object *that*)

equals

public boolean **equals** (*setControlSignal_args that*)

fieldForId

public *_Fields* **fieldForId** (int *fieldId*)

getFieldValue

public java.lang.Object **getFieldValue** (*_Fields field*)

getId

public int **getId** ()

getPower

public double **getPower** ()

hashCode

public int **hashCode**()

isSet

public boolean **isSet**(*_Fields field*)
Returns true if field corresponding to fieldID is set (has been assigned a value) and false otherwise

isSetId

public boolean **isSetId**()
Returns true if field id is set (has been assigned a value) and false otherwise

isSetPower

public boolean **isSetPower**()
Returns true if field power is set (has been assigned a value) and false otherwise

read

public void **read**(org.apache.thrift.protocol.TProtocol *iprot*)

setFieldValue

public void **setFieldValue**(*_Fields field*, java.lang.Object *value*)

setId

public *setControlSignal_args* **setId**(int *id*)

setIdIsSet

public void **setIdIsSet**(boolean *value*)

setPower

public *setControlSignal_args* **setPower**(double *power*)

setPowerIsSet

public void **setPowerIsSet**(boolean *value*)

toString

public java.lang.String **toString**()

unsetId

public void **unsetId**()

unsetPower

public void **unsetPower**()

validate

public void **validate**()

write

public void **write**(org.apache.thrift.protocol.TProtocol *oprot*)

6.31.136 RemoteSimulationService.setControlSignal_args._Fields

public enum **_Fields** implements org.apache.thrift.TFieldIdEnum

The set of fields this struct contains, along with convenience methods for finding and manipulating them.

Enum Constants

ID

public static final *RemoteSimulationService.setControlSignal_args._Fields* **ID**

POWER

public static final *RemoteSimulationService.setControlSignal_args._Fields* **POWER**

6.31.137 RemoteSimulationService.setControlSignal_result

public static class **setControlSignal_result** implements org.apache.thrift.TBase<*setControlSignal_result*, *setControlSignal_re*

Fields

ex

public *UnknownSimulationIdException* **ex**

metaDataMap

public static final java.util.Map<*_Fields*, org.apache.thrift.meta_data.FieldMetaData> **metaDataMap**

Constructors

setControlSignal_result

public **setControlSignal_result**()

setControlSignal_result

public **setControlSignal_result**(*UnknownSimulationIdException ex*)

setControlSignal_result

public **setControlSignal_result**(*setControlSignal_result other*)
Performs a deep copy on *other*.

Methods

clear

public void **clear**()

compareTo

public int **compareTo**(*setControlSignal_result other*)

deepCopy

public *setControlSignal_result* **deepCopy**()

equals

public boolean **equals**(java.lang.Object *that*)

equals

public boolean **equals**(*setControlSignal_result that*)

fieldForId

public *_Fields* **fieldForId**(int *fieldId*)

getEx

public *UnknownSimulationIdException* **getEx**()

getFieldValue

public java.lang.Object **getFieldValue**(*_Fields field*)

hashCode

public int **hashCode**()

isSet

public boolean **isSet**(*_Fields field*)

Returns true if field corresponding to fieldID is set (has been assigned a value) and false otherwise

isSetEx

public boolean **isSetEx**()

Returns true if field ex is set (has been assigned a value) and false otherwise

read

public void **read**(org.apache.thrift.protocol.TProtocol *iprot*)

setEx

public *setControlSignal_result* **setEx**(*UnknownSimulationIdException ex*)

setExIsSet

public void **setExIsSet**(boolean *value*)

setFieldValue

public void **setFieldValue**(*_Fields field*, java.lang.Object *value*)

toString

public java.lang.String **toString**()

unsetEx

public void **unsetEx**()

validate

public void **validate**()

write

public void **write**(org.apache.thrift.protocol.TProtocol *oprot*)

6.31.138 RemoteSimulationService.setControlSignal_result._Fields

public enum **_Fields** implements org.apache.thrift.TFieldIdEnum

The set of fields this struct contains, along with convenience methods for finding and manipulating them.

Enum Constants

EX

public static final *RemoteSimulationService.setControlSignal_result._Fields* **EX**

6.31.139 RemoteSimulationService.toJsonMessages_args

public static class **toJsonMessages_args** implements org.apache.thrift.TBase<*toJsonMessages_args*, *toJsonMessages_args._Fiel*

Fields

id

public int **id**

metaDataMap

public static final java.util.Map<*_Fields*, org.apache.thrift.meta_data.FieldMetaData> **metaDataMap**

Constructors

toJsonMessages_args

public **toJsonMessages_args**()

toJsonMessages_args

public **toJsonMessages_args** (int *id*)

toJsonMessages_args

public **toJsonMessages_args** (*toJsonMessages_args other*)
Performs a deep copy on *other*.

Methods

clear

public void **clear** ()

compareTo

public int **compareTo** (*toJsonMessages_args other*)

deepCopy

public *toJsonMessages_args* **deepCopy** ()

equals

public boolean **equals** (java.lang.Object *that*)

equals

public boolean **equals** (*toJsonMessages_args that*)

fieldForId

public *_Fields* **fieldForId** (int *fieldId*)

getFieldValue

public java.lang.Object **getFieldValue** (*_Fields field*)

getId

public int **getId** ()

hashCode

public int **hashCode**()

isSet

public boolean **isSet**(*_Fields field*)

Returns true if field corresponding to fieldID is set (has been assigned a value) and false otherwise

isSetId

public boolean **isSetId**()

Returns true if field id is set (has been assigned a value) and false otherwise

read

public void **read**(org.apache.thrift.protocol.TProtocol *iprot*)

setFieldValue

public void **setFieldValue**(*_Fields field*, java.lang.Object *value*)

setId

public *toJsonMessages_args* **setId**(int *id*)

setIdIsSet

public void **setIdIsSet**(boolean *value*)

toString

public java.lang.String **toString**()

unsetId

public void **unsetId**()

validate

public void **validate**()

write

public void **write** (org.apache.thrift.protocol.TProtocol *oprot*)

6.31.140 RemoteSimulationService.toJsonMessages_args._Fields

public enum **_Fields** implements org.apache.thrift.TFieldIdEnum
The set of fields this struct contains, along with convenience methods for finding and manipulating them.

Enum Constants

ID

public static final *RemoteSimulationService.toJsonMessages_args._Fields* **ID**

6.31.141 RemoteSimulationService.toJsonMessages_result

public static class **toJsonMessages_result** implements org.apache.thrift.TBase<*toJsonMessages_result*, *toJsonMessages_result*.

Fields

ex

public *UnknownSimulationIdException* **ex**

metaDataMap

public static final java.util.Map<*_Fields*, org.apache.thrift.meta_data.FieldMetaData> **metaDataMap**

success

public java.lang.String **success**

Constructors

toJsonMessages_result

public **toJsonMessages_result** ()

toJsonMessages_result

public **toJsonMessages_result** (java.lang.String *success*, *UnknownSimulationIdException* *ex*)

toJsonMessages_result

public **toJsonMessages_result** (*toJsonMessages_result other*)
Performs a deep copy on *other*.

Methods

clear

public void **clear** ()

compareTo

public int **compareTo** (*toJsonMessages_result other*)

deepCopy

public *toJsonMessages_result* **deepCopy** ()

equals

public boolean **equals** (java.lang.Object *that*)

equals

public boolean **equals** (*toJsonMessages_result that*)

fieldForId

public *_Fields* **fieldForId** (int *fieldId*)

getEx

public *UnknownSimulationIdException* **getEx** ()

getFieldValue

public java.lang.Object **getFieldValue** (*_Fields field*)

getSuccess

public java.lang.String **getSuccess** ()

hashCode

public int **hashCode** ()

isSet

public boolean **isSet** (*_Fields field*)
Returns true if field corresponding to fieldID is set (has been assigned a value) and false otherwise

isSetEx

public boolean **isSetEx** ()
Returns true if field ex is set (has been assigned a value) and false otherwise

isSetSuccess

public boolean **isSetSuccess** ()
Returns true if field success is set (has been assigned a value) and false otherwise

read

public void **read** (org.apache.thrift.protocol.TProtocol *iprot*)

setEx

public *toJsonMessages_result* **setEx** (*UnknownSimulationIdException ex*)

setExIsSet

public void **setExIsSet** (boolean *value*)

setFieldValue

public void **setFieldValue** (*_Fields field*, java.lang.Object *value*)

setSuccess

public *toJsonMessages_result* **setSuccess** (java.lang.String *success*)

setSuccessIsSet

public void **setSuccessIsSet** (boolean *value*)

toString

public java.lang.String **toString**()

unsetEx

public void **unsetEx**()

unsetSuccess

public void **unsetSuccess**()

validate

public void **validate**()

write

public void **write**(org.apache.thrift.protocol.TProtocol *oprot*)

6.31.142 RemoteSimulationService.toJsonMessages_result._Fields

public enum **_Fields** implements org.apache.thrift.TFieldIdEnum

The set of fields this struct contains, along with convenience methods for finding and manipulating them.

Enum Constants

EX

public static final *RemoteSimulationService.toJsonMessages_result._Fields* **EX**

SUCCESS

public static final *RemoteSimulationService.toJsonMessages_result._Fields* **SUCCESS**

6.31.143 RemoteSimulationService.toJsonState_args

public static class **toJsonState_args** implements org.apache.thrift.TBase<*toJsonState_args*, *toJsonState_args._Fields*>, java.io.Se

Fields

id

public int **id**

metaDataMap

public static final java.util.Map<*_Fields*, org.apache.thrift.meta_data.FieldMetaData> **metaDataMap**

Constructors

toJsonState_args

public **toJsonState_args** ()

toJsonState_args

public **toJsonState_args** (int *id*)

toJsonState_args

public **toJsonState_args** (*toJsonState_args other*)
Performs a deep copy on *other*.

Methods

clear

public void **clear** ()

compareTo

public int **compareTo** (*toJsonState_args other*)

deepCopy

public *toJsonState_args* **deepCopy** ()

equals

public boolean **equals** (java.lang.Object *that*)

equals

public boolean **equals** (*toJsonState_args that*)

fieldForId

public *_Fields* **fieldForId** (int *fieldId*)

getFieldValue

public java.lang.Object **getFieldValue** (*_Fields field*)

getId

public int **getId** ()

hashCode

public int **hashCode** ()

isSet

public boolean **isSet** (*_Fields field*)

Returns true if field corresponding to fieldID is set (has been assigned a value) and false otherwise

isSetId

public boolean **isSetId** ()

Returns true if field id is set (has been assigned a value) and false otherwise

read

public void **read** (org.apache.thrift.protocol.TProtocol *iprot*)

setFieldValue

public void **setFieldValue** (*_Fields field*, java.lang.Object *value*)

setId

public *toJsonState_args* **setId** (int *id*)

setIdIsSet

public void **setIdIsSet** (boolean *value*)

toString

public java.lang.String **toString** ()

unsetId

public void **unsetId**()

validate

public void **validate**()

write

public void **write**(org.apache.thrift.protocol.TProtocol *oprot*)

6.31.144 RemoteSimulationService.toJsonState_args._Fields

public enum **_Fields** implements org.apache.thrift.TFieldIdEnum

The set of fields this struct contains, along with convenience methods for finding and manipulating them.

Enum Constants

ID

public static final *RemoteSimulationService.toJsonState_args._Fields* **ID**

6.31.145 RemoteSimulationService.toJsonState_result

public static class **toJsonState_result** implements org.apache.thrift.TBase<*toJsonState_result*, *toJsonState_result._Fields*>, java

Fields

ex

public *UnknownSimulationIdException* **ex**

metaDataMap

public static final java.util.Map<*_Fields*, org.apache.thrift.meta_data.FieldMetaData> **metaDataMap**

success

public java.lang.String **success**

Constructors

toJsonState_result

public **toJsonState_result**()

toJsonState_result

public **toJsonState_result**(java.lang.String *success*, *UnknownSimulationIdException ex*)

toJsonState_result

public **toJsonState_result**(*toJsonState_result other*)
Performs a deep copy on *other*.

Methods

clear

public void **clear**()

compareTo

public int **compareTo**(*toJsonState_result other*)

deepCopy

public *toJsonState_result* **deepCopy**()

equals

public boolean **equals**(java.lang.Object *that*)

equals

public boolean **equals**(*toJsonState_result that*)

fieldForId

public *_Fields* **fieldForId**(int *fieldId*)

getEx

public *UnknownSimulationIdException* **getEx**()

getFieldValue

public java.lang.Object **getFieldValue** (*_Fields field*)

getSuccess

public java.lang.String **getSuccess** ()

hashCode

public int **hashCode** ()

isSet

public boolean **isSet** (*_Fields field*)

Returns true if field corresponding to fieldID is set (has been assigned a value) and false otherwise

isSetEx

public boolean **isSetEx** ()

Returns true if field ex is set (has been assigned a value) and false otherwise

isSetSuccess

public boolean **isSetSuccess** ()

Returns true if field success is set (has been assigned a value) and false otherwise

read

public void **read** (org.apache.thrift.protocol.TProtocol *iprot*)

setEx

public *toJsonState_result* **setEx** (*UnknownSimulationIdException ex*)

setExIsSet

public void **setExIsSet** (boolean *value*)

setFieldValue

public void **setFieldValue** (*_Fields field*, java.lang.Object *value*)

setSuccess

public *toJsonState_result* **setSuccess** (java.lang.String *success*)

setSuccessIsSet

public void **setSuccessIsSet** (boolean *value*)

toString

public java.lang.String **toString** ()

unsetEx

public void **unsetEx** ()

unsetSuccess

public void **unsetSuccess** ()

validate

public void **validate** ()

write

public void **write** (org.apache.thrift.protocol.TProtocol *oprot*)

6.31.146 RemoteSimulationService.toJsonState_result._Fields

public enum **_Fields** implements org.apache.thrift.TFieldIdEnum

The set of fields this struct contains, along with convenience methods for finding and manipulating them.

Enum Constants

EX

public static final *RemoteSimulationService.toJsonState_result._Fields* **EX**

SUCCESS

public static final *RemoteSimulationService.toJsonState_result._Fields* **SUCCESS**

6.31.147 RemoteSimulationServiceHandler

public class **RemoteSimulationServiceHandler** implements *RemoteSimulationService.Iface*

Implements the remote simulation service functions.

Author Kyle M. Douglass

Constructors

RemoteSimulationServiceHandler

public **RemoteSimulationServiceHandler** ( )

Initializes the remote handler.

RemoteSimulationServiceHandler

public **RemoteSimulationServiceHandler** (*SimulationManager inputManager*)

Initializes the remote handler with a pre-specified SimulationManager.

Parameters

- **inputManager** – SimulationManager that handles multiple simulations.

Methods

createSimulation

public int **createSimulation** ( )

Creates a new simulation and returns its ID. This creates a copy of one of the already created simulations in the SimulationManager. If you wish to create a simulation with all new parameters, then you will need to create a new SimulationManager.

Returns The new simulation's ID.

deleteSimulation

public void **deleteSimulation** (int *id*)

Deletes the simulation with the given ID.

Parameters

- **id** – The ID of the simulation to delete.

getCameraJsonName

public String **getCameraJsonName** (int *id*)

Returns the name of the JSON key for the camera info.

Parameters

- **id** – The simulation ID.

Throws

- *UnknownSimulationIdException* –

Returns The name of the key indicating the camera information.

getControlSignal

public double **getControlSignal** (int *id*)

Returns the control signal for the current simulation.

Parameters

- **id** – The simulation ID.

Throws

- *ch.epfl.leb.sass.server.UnknownSimulationIdException* –

Returns The value of the simulation's current control signal.

getFluorescenceJsonName

public String **getFluorescenceJsonName** (int *id*)

Returns the name of the JSON key for the fluorescence info.

Parameters

- **id** – The simulation ID.

Throws

- *UnknownSimulationIdException* –

Returns The name of the key indicating the fluorescence information.

getFovSize

public double **getFovSize** (int *id*)

Returns the field-of-view size in object space units.

Parameters

- **id** – The simulation ID.

Throws

- *ch.epfl.leb.sass.server.UnknownSimulationIdException* –

Returns The size of the simulation's FOV.

getImageCount

public int **getImageCount** (int *id*)

Returns the number of images already simulated.

Parameters

- **id** – The simulation ID.

Throws

- *UnknownSimulationIdException* –

Returns The number of images already simulated.

getLaserJsonName

public String **getLaserJsonName** (int *id*)

Returns the name of the JSON key for the laser info.

Parameters

- **id** – The simulation ID.

Throws

- *UnknownSimulationIdException* –

Returns The name of the key indicating the laser information.

getNextImage

public ByteBuffer **getNextImage** (int *id*)

Advances the simulator by one time step and returns the image.

Parameters

- **id** – The simulation ID.

Throws

- *ch.epfl.leb.sass.server.ImageGenerationException* –

Returns A buffer containing the TIFF-encoded byte string of the simulator's next image.

getObjectSpacePixelSize

public double **getObjectSpacePixelSize** (int *id*)

Returns the object space pixel size. Units are the same as those of the camera pixel size.

Parameters

- **id** – The simulation ID.

Throws

- *UnknownSimulationIdException* –

Returns The object space pixel size.

getObjectiveJsonName

public String **getObjectiveJsonName** (int *id*)

Returns the name of the JSON key for the objective info.

Parameters

- **id** – The simulation ID.

Throws

- *UnknownSimulationIdException* –

Returns The name of the key indicating the laser information.

getServerStatus

public String **getServerStatus** ()

This method is used to determine whether the server is running.

Returns Basic information concerning the status of the server.

getShortTrueSignalDescription

public String **getShortTrueSignalDescription** (int *id*)

Returns a brief description of the ground-truth signal.

Parameters

- **id** – The simulation ID.

Throws

- *UnknownSimulationIdException* –

Returns A brief description of the ground truth signal.

getStageJsonName

public String **getStageJsonName** (int *id*)

Returns the name of the JSON key for the stage info.

Parameters

- **id** – The simulation ID.

Throws

- *UnknownSimulationIdException* –

Returns The name of the key indicating the stage information.

getTrueSignal

public double **getTrueSignal** (int *id*, int *imageNum*)

Returns the ground-truth signal of the image at the given index.

Parameters

- **id** – The simulation ID.
- **imageNum** – The index of the image to get the true signal for.

Throws

- *UnknownSimulationIdException* –

Returns The ground truth signal.

incrementTimeStep

public void **incrementTimeStep** (int *id*)

Advances the simulation without creating an image.

Parameters

- **id** – The simulation ID.

Throws

- ***UnknownSimulationIdException*** –

setControlSignal

public void **setControlSignal** (int *id*, double *power*)

Sets the activation laser power in the simulation.

Parameters

- **id** – The simulation ID.
- **power** – The power of the laser.

Throws

- ***UnknownSimulationIdException*** –

toJsonMessages

public String **toJsonMessages** (int *id*)

Returns messages about changes in the simulation state as a JSON string. Unlike `toJsonState()`, which returns information about the *current* state of just the fluorophores, this method returns the messages from individual simulation components that contain information about changes in their state that have occurred since the last time this method was called.

Parameters

- **id** – The simulation ID.

Throws

- ***UnknownSimulationIdException*** –

Returns A JSON string containing the messages that were recorded.

toJsonState

public String **toJsonState** (int *id*)

Returns information on the simulation's current state as a JSON object. Unlike `toJsonMessages()`, which returns information about previous changes in the simulation's state, this method reports on the current state of the simulation.

Parameters

- **id** – The simulation ID.

Throws

- ***UnknownSimulationIdException*** –

Returns The state of the sample fluorescence as a JSON string.

6.31.148 RemoteSimulationServiceHandlerTest

public class **RemoteSimulationServiceHandlerTest**

Author kmdouglass

Constructors

RemoteSimulationServiceHandlerTest

public **RemoteSimulationServiceHandlerTest** ( )

Methods

testGetNextImage

public void **testGetNextImage** ( )
Test of getNextImage method, of class RemoteSimulationServiceHandler.

Throws

- *ch.epfl.leb.sass.server.UnknownSimulationIdException* –
- *ch.epfl.leb.sass.server.ImageGenerationException* –
- *ch.epfl.leb.sass.utils.images.ImageShapeException* –

testGetServerStatus

public void **testGetServerStatus** ( )
Test of getServerStatus method, of class RemoteSimulationServiceHandler.

6.31.149 UnknownSimulationIdException

public class **UnknownSimulationIdException** extends org.apache.thrift.TException implements org.apache.thrift.TBase<*Unkn*

Fields

metaDataMap

public static final java.util.Map<*_Fields*, org.apache.thrift.meta_data.FieldMetaData> **metaDataMap**

Constructors

UnknownSimulationIdException

public **UnknownSimulationIdException** ( )

UnknownSimulationIdException

public **UnknownSimulationIdException** (*UnknownSimulationIdException other*)
Performs a deep copy on *other*.

Methods

clear

public void **clear** ()

compareTo

public int **compareTo** (*UnknownSimulationIdException other*)

deepCopy

public *UnknownSimulationIdException* **deepCopy** ()

equals

public boolean **equals** (java.lang.Object *that*)

equals

public boolean **equals** (*UnknownSimulationIdException that*)

fieldForId

public *_Fields* **fieldForId** (int *fieldId*)

getFieldValue

public java.lang.Object **getFieldValue** (*_Fields field*)

hashCode

public int **hashCode** ()

isSet

public boolean **isSet** (*_Fields field*)
Returns true if field corresponding to fieldID is set (has been assigned a value) and false otherwise

read

public void **read** (org.apache.thrift.protocol.TProtocol *iprot*)

setFieldValue

public void **setFieldValue** (*_Fields field*, java.lang.Object *value*)

toString

public java.lang.String **toString** ()

validate

public void **validate** ()

write

public void **write** (org.apache.thrift.protocol.TProtocol *oprot*)

6.31.150 UnknownSimulationIdException._Fields

public enum **_Fields** implements org.apache.thrift.TFieldIdEnum

The set of fields this struct contains, along with convenience methods for finding and manipulating them.

Enum Constants

6.32 ch.epfl.leb.sass.simulator

6.32.1 SimulationManager

public interface **SimulationManager**

Management tool for handling multiple Simulators.

Author Kyle M. Douglass

Methods

addSimulator

public void **addSimulator** (*Simulator simulator*)

Adds a simulation to the manager.

Parameters

- **simulator** – The simulation to add to the manager.

getIds

public List<Integer> **getIds**()

Returns a list of simulation IDs currently managed by this Manager. A simulation manager is a Singleton; there is only one.

Returns A list of simulation ID numbers.

getMostRecentMicroscope

public *Microscope* **getMostRecentMicroscope**()

Returns the most recent microscope that was used to create a simulation. This method serves as a sort of cache for remembering the most recently created `Microscope` object. Its purpose is to allow for easy generation of new Simulators. This method will return null if the SimulatorManager has never managed a simulation.

Returns A copy of the Microscope object null.

getSimulator

public *Simulator* **getSimulator**(int *id*)

Returns a reference to the simulator corresponding to the ID.

Parameters

- **id** – The ID number of a simulation.

Returns A reference to the Simulator.

removeSimulator

public void **removeSimulator**(int *id*)

Removes a Simulator from the manager.

Parameters

- **id** – The ID number of a simulation to remove.

6.32.2 Simulator

public interface **Simulator**

The interface that defines everything that a Simulator should do.

Author Marcel Stefko, Kyle M. Douglass

Methods

getCameraJsonName

public String **getCameraJsonName**()

Returns the name of the JSON key for the camera info.

Returns The name of the key indicating the camera information.

See also: `.toJsonState()`

getControlSignal

public double **getControlSignal**()

Returns currently set control signal of the generator (e.g. laser power settings).

Returns control signal value

getCustomParameters

public HashMap<String, Double> **getCustomParameters**()

Returns custom parameters of the generator.

Returns map of custom parameters

getFOVSize

public double **getFOVSize**()

Returns the size of the field-of-view in object space units.

Returns size of current FOV in object space units.

getFluorescenceJsonName

public String **getFluorescenceJsonName**()

Returns the name of the JSON key for the fluorescence state info.

Returns The name of the key indicating the fluorescence information.

See also: `.toJsonState()`

getId

public int **getId**()

Returns the unique ID assigned to this simulator.

Returns The integer ID of this simulator.

getImageCount

public int **getImageCount**()

Returns the number of images simulated. Because the simulation can advance without generating an image, this value will be less than or equal to the number of simulation time steps. Use *incrementTimeStep()* to advance the simulation one time step without generating an image.

Returns The number of images that have been simulated.

getLaserJsonName

public String **getLaserJsonName**()

Returns the name of the JSON key for the laser state info.

Returns The name of the key indicating the laser information.

See also: `.toJsonState()`

getMessages

public List<*Message*> **getMessages**()

Returns messages about changes in the simulation state. Unlike `getSimulationState()`, which returns information about the *current* state of the simulation, this method returns the messages from individual components that contain information about changes in their state that have occurred since the last time this method was called.

Returns A list containing the state change messages.

getMicroscope

public *Microscope* **getMicroscope**()

Returns a copy of the Microscope that is controlled by this simulation. The copy that is returned is a deep copy of the `Microscope` that the simulation was initialized with.

Returns A copy of the Microscope object controlled by this simulation.

getNextImage

public *ImageS* **getNextImage**()

Generates a new image and adds it to the internal stack.

Returns newly generated image

getObjectSpacePixelSize

public double **getObjectSpacePixelSize**()

Returns the size of a pixel in object space units. The units of this quantity are determined by those of the camera's pixels. The value is the magnitude of the camera's pixel size divided by the objective's magnification.

Returns length of one pixel side in object space units.

getObjectiveJsonName

public String **getObjectiveJsonName**()

Returns the name of the JSON key for the objective state info.

Returns The name of the key indicating the objective information.

See also: `.toJsonState()`

getShortTrueSignalDescription

public String **getShortTrueSignalDescription**()

Returns a brief description of the ground truth signal.

Returns A short description of the truth signal, typically its units.

getStack

public *ImageS* **getStack**()

Returns internal stack with all generated images.

Returns internal stack

getStageJsonName

public String **getStageJsonName**()

Returns the name of the JSON key for the stage state info.

Returns The name of the key indicating the stage information.

See also: `.toJsonState()`

getTrueSignal

public double **getTrueSignal**(int *image_no*)

Returns the actual value of signal (if applicable) for given image.

Parameters

- **image_no** – 1-based image number in history

Returns value of signal (e.g. no. of active emitters)

incrementTimeStep

public void **incrementTimeStep**()

Increments the simulation by one time step without creating an image.

saveMessages

public void **saveMessages**(File *file*)

Saves the messages in the cache to a select file.

Parameters

- **file** – The file to save to.

saveStack

public void **saveStack**(File *file*)

Saves the .tif image stack to a select file.

Parameters

- **file** – file to save to

saveState

public void **saveState** (File *file*)

Saves the current state of the simulation.

Parameters

- **file** – The file to save to.

setControlSignal

public void **setControlSignal** (double *value*)

Sets control signal of the generator (e.g. laser power). This should be used by the controller.

Parameters

- **value** – new value of the control signal

setCustomParameters

public void **setCustomParameters** (HashMap<String, Double> *map*)

Sets custom parameters of the generator.

Parameters

- **map** – map of custom parameters

toJsonMessages

public JsonElement **toJsonMessages** ()

Returns messages about changes in the simulation state as a JSON object. Unlike `toJsonState()`, which returns information about the *current* state of the simulation, this method returns the messages from individual simulation components that contain information about changes in their state that have occurred since the last time this method was called.

Returns A JSON object containing the simulation messages.

toJsonState

public JsonElement **toJsonState** ()

Returns information on the simulation's current state as a JSON object. Unlike `toJsonMessages()`, which returns information about previous changes in the simulation's state, this method reports on the current state of the simulation.

Returns A JSON object containing information on the simulation state.

6.33 ch.epfl.leb.sass.simulator.internal

6.33.1 AbstractSimulator

public abstract class **AbstractSimulator** implements *Simulator*

Fields and methods that all Simulators should possess.

Author Marcel Stefko

Fields

id

protected int **id**
A unique ID assigned to this simulator.

parameters

protected HashMap<String, Double> **parameters**
Map of custom parameters for the generator.

simulatorIds

protected static int **simulatorIds**
Running total of the number of simulators that have been created.

stack

protected *ImageS* **stack**
Stack to which the generated images are appended.

Constructors

AbstractSimulator

public **AbstractSimulator**()
Initializes the simulator.

Methods

getId

public int **getId**()
Returns the integer ID of the simulator instance.

Returns A unique integer ID for this simulator.

getImageCount

public int **getImageCount**()

getStack

public *ImageS* **getStack**()

saveStack

public void **saveStack** (File *file*)

6.33.2 DefaultSimulationManager

public class **DefaultSimulationManager** implements *SimulationManager*

A default implementation of the SimulationManager class.

Author Kyle M. Douglass

Constructors

DefaultSimulationManager

public **DefaultSimulationManager** ()

Default constructor.

DefaultSimulationManager

public **DefaultSimulationManager** (ConcurrentHashMap *simulations*)

Adds a table of simulations to a new SimulationManager instance.

Parameters

- **simulations** – A table of simulations to add to the new instance.

Methods

addSimulator

public void **addSimulator** (*Simulator simulator*)

Adds a simulation to the manager.

Parameters

- **The** – simulation to add to the manager.

getIds

public List<Integer> **getIds** ()

Returns the list of simulation IDs managed by the manager.

Returns The list of simulation IDs managed by the manager.

getMostRecentMicroscope

public *Microscope* **getMostRecentMicroscope** ()

Returns the most recent microscope that was used to create a simulation. This method serves as a sort of cache for remembering the most recently created `Microscope` object. Its purpose is to allow for easy generation of new Simulators. This method will return null if the SimulatorManager has never managed a simulation.

Returns A copy of the Microscope object or null.

getSimulator

public *Simulator* **getSimulator** (int *id*)

Returns a reference to the simulator corresponding to the ID.

Parameters

- **id** – The ID number of a simulation.

Returns A reference to the Simulator.

removeSimulator

public void **removeSimulator** (int *id*)

Removes a Simulator from the manager.

Parameters

- **id** – The ID number of a simulation to remove.

6.33.3 DefaultSimulationManagerTest

public class **DefaultSimulationManagerTest**

Unit tests for the SimulationManager class.

Author Kyle M. Douglass

Fields

dummySim1

Simulator **dummySim1**

dummySim2

Simulator **dummySim2**

Methods

setUp

public void **setUp** ()

Creates a dummy table of simulations to be managed.

testAddSimulator

public void **testAddSimulator** ()

Test of addSimulator method, of class DefaultSimulationManager.

testGetIds

public void **testGetIds**()
Test of getIds method, of class DefaultSimulationManager.

testGetSimulator

public void **testGetSimulator**()
Test of getSimulator method, of class DefaultSimulationManager.

testRemoveSimulator

public void **testRemoveSimulator**()
Test of removeSimulator method, of class DefaultSimulationManager.

6.33.4 DefaultSimulator

public class **DefaultSimulator** extends *AbstractSimulator*
The basic simulation engine from which others may be derived.

Author Marcel Stefko, Kyle M. Douglass

Fields

LOGGER

public static final Logger **LOGGER**

Constructors

DefaultSimulator

public **DefaultSimulator**(*Microscope microscope*)
Initialize the generator.

Parameters

- **microscope** –

Methods

getCameraJsonName

public String **getCameraJsonName**()
Returns the JSON member name assigned to the camera.

Returns The JSON member name for the Camera field.

getControlSignal

public double **getControlSignal**()

getCustomParameters

public HashMap<String, Double> **getCustomParameters**()

getFOVSize

public double **getFOVSize**()

Returns The size of the FOV in square object-space units.

getFluorescenceJsonName

public String **getFluorescenceJsonName**()

Returns the name of the JSON key for the fluorescence info.

Returns The name of the key indicating the fluorescence information.

getLaserJsonName

public String **getLaserJsonName**()

Returns the name of the JSON key for the laser info.

Returns The name of the key indicating the laser information.

getMessages

public List<*Message*> **getMessages**()

Returns messages about changes in the simulation state. Unlike `getSimulationState()`, which returns information about the *current* state of the simulation, this method returns the messages from individual components that contain information about changes in their state that have occurred since the last time this method was called.

Returns A list containing the state change messages.

getMicroscope

public *Microscope* **getMicroscope**()

Returns a copy of the Microscope that is controlled by this simulation. The copy that is returned is a deep copy of the `Microscope` that the simulation was initialized with.

Returns A copy of the Microscope object controlled by this simulation.

getNextImage

public *ImageS* **getNextImage**()

Generates a new image and adds it to the internal stack.

Returns newly generated image

getObjectSpacePixelSize

public double **getObjectSpacePixelSize**()

Returns Length of one pixel side in object-space units.

getObjectiveJsonName

public String **getObjectiveJsonName**()

Returns the name of the JSON key for the objective state info.

Returns The name of the key indicating the objective information.

See also: `.toJsonState()`

getShortTrueSignalDescription

public String **getShortTrueSignalDescription**()

getStageJsonName

public String **getStageJsonName**()

Returns the name of the JSON key for the stage info.

Returns The name of the key indicating the stage information.

getStateListener

public *DefaultSimulator.StateListener* **getStateListener**()

Returns this instance's StateListener. This method is primarily for testing purposes and is not exposed in the Simulator interface.

Returns A reference to this instance's StateListener.

getTrueSignal

public double **getTrueSignal**(int *image_no*)

incrementTimeStep

public void **incrementTimeStep**()

Advance the simulation by one time step (i.e. one frame). Simulates a frame but does not create an image.

saveMessages

public void **saveMessages** (File *file*)

Saves the messages in the cache to a select file.

Parameters

- **file** – The file to save to.

saveState

public void **saveState** (File *file*)

Saves the current state of the simulation.

Parameters

- **file** – The file to save to.

setControlSignal

public void **setControlSignal** (double *value*)

setCustomParameters

public void **setCustomParameters** (HashMap<String, Double> *map*)

toJsonMessages

public JsonElement **toJsonMessages** ()

Returns messages about changes in the simulation state as a JSON object. Unlike `toJsonState()`, which returns information about the *current* state of the simulation, this method returns the messages from individual simulation components that contain information about changes in their state that have occurred since the last time this method was called.

Returns A JSON object containing the simulation messages.

toJsonState

public JsonElement **toJsonState** ()

Returns information on the simulation's current state as a JSON object. Unlike `toJsonMessages()`, which returns information about previous changes in the simulation's state, this method reports on the current state of the simulation.

Returns A JSON object containing information on the simulation state.

6.33.5 DefaultSimulator.StateListener

class **StateListener** implements *Listener*

The StateListener listens for changes in the simulation's state. These changes can occur at any time on a continuous interval between the simulation time steps.

Fields

transitions

ArrayList<*Message*> **transitions**
A cache containing the state transition messages.

Methods

dumpMessageCache

public List<*Message*> **dumpMessageCache** ( )
Dumps the contents of the cache to a JSON string. Calling this method will irreversibly clear the cache. This method will return null if the cache is empty.

Returns The contents of the cache as JSON string or null.

update

public void **update** (Object *data*)
This method is called by an Observable when its state has changed.

Parameters

- **data** – The data object that is passed from the Observable.

6.33.6 DefaultSimulatorTest

public class **DefaultSimulatorTest**
Unit tests for the DefaultSimulator class.

Author Kyle M. Douglass

Fields

tempDir

public TemporaryFolder **tempDir**

Constructors

DefaultSimulatorTest

public **DefaultSimulatorTest** ( )

Methods

setUp

public void **setUp** ( )

testSaveMessages

public void **testSaveMessages**()
Test of saveMessages method, of class DefaultSimulator.

testSaveState

public void **testSaveState**()
Test of saveState method, of class DefaultSimulator.

testStateListenerDumpMessageCache

public void **testStateListenerDumpMessageCache**()
Test of dumpMessageCache method, of class DefaultSimulator.StateListener.

testStateListenerUpdate

public void **testStateListenerUpdate**()
Test of update method, of class DefaultSimulator.StateListener.

6.33.7 ImageJSimulator

public class **ImageJSimulator** extends *DefaultSimulator*
The default simulator that is run as, for example, the ImageJ plugin.

Author Marcel Stefko

Fields

TIMEPERFRAME

protected final long **TIMEPERFRAME**
The time duration of each frame. This is here only for compatibility with ALICA's analyzers, which require a time argument.

analyzer

protected final Analyzer **analyzer**
Analyzer which analyzes generated images

controller

protected final Controller **controller**
Takes the output of a single analyzer, processes it, and outputs a signal to the generator, for feedback loop control.

history

protected HashMap<Integer, JSONObject> **history**
Records of values of output of analyzer, controller.

image_count

protected int **image_count**
Number of already-generated images.

Constructors

ImageJSimulator

public **ImageJSimulator**(*Microscope microscope*, Analyzer *analyzer*, Controller *controller*)
Initialize the simulator from user-specified components.

Parameters

- **microscope** – The microscope to be simulated.
- **analyzer** – An analyzer for processing images from the microscope.
- **controller** – A controller that adjusts the state of the microscope.

Methods

execute

public *ImageS* **execute**(int *no_of_images*, int *controller_refresh_rate*, String *csv_save_path*, String *tiff_save_path*)
An example simulation

Parameters

- **no_of_images** –
- **controller_refresh_rate** –
- **csv_save_path** –
- **tiff_save_path** –

getImageCount

public int **getImageCount**()
Returns the number of generated images since simulation start.

Returns number of generated images

incrementCounter

public void **incrementCounter**()
Increments image counter in case an image was generated outside of this class.

saveStack

public void **saveStack** (File *tiff_file*)
Save current ImageStack to TIFF file

Parameters

- **tiff_file** – file to save to

saveToCsv

public void **saveToCsv** (File *file*)
Saves the data for generator, analyzer and controller for each frame into a .csv file

Parameters

- **file** – destination csv file

6.33.8 RPCSimulator

public class **RPCSimulator** extends *DefaultSimulator*
A simulator that is specialized for control by remote procedure calls (RPCs).

Author Kyle M. Douglass

Constructors

RPCSimulator

public **RPCSimulator** (*Microscope microscope*)
Initializes the SimpleSimulator and connects it to the simulation engine.

Parameters

- **microscope** – The engine that runs the simulation.

6.34 ch.epfl.leb.sass.utils

6.34.1 Constants

public class **Constants**
Implements physical constants.

Author Kyle M. Douglass

Fields

C

public static final double **C**
The speed of light in meters / second.

EPSILON_0

public static final double **EPSILON_0**
The permittivity of free space in Farads / meter.

HC

public static final double **HC**
Planck's constant times the speed of light in Joules * meters.

6.34.2 DeepCopy

public class **DeepCopy**
Makes a deep copy of a serializable object.

Author Kyle M. Douglass

Methods

deepCopy

public static Object **deepCopy** (Object *object*)
Makes a deep copy of any Java object that is passed.

See also: https://www.journaldev.com/17129/java-deep-copy-object

6.34.3 DeepCopyTest

public class **DeepCopyTest**
Unit tests for the DeepCopy class.

Author Kyle M. Douglass

Constructors

DeepCopyTest

public **DeepCopyTest** ()

Methods

testDeepCopy

public void **testDeepCopy** ()

6.34.4 RNG

public final class **RNG**

Random number generator for STORMsim. Ensures repeatability.

Author stefko

Methods

getGammaGenerator

public static Gamma **getGammaGenerator** ( )

Returns Gamma distribution RNG

getGaussianGenerator

public static Normal **getGaussianGenerator** ( )

Returns Gaussian distribution RNG

getPoissonGenerator

public static Poisson **getPoissonGenerator** ( )

Returns Poisson RNG

getUniformGenerator

public static Random **getUniformGenerator** ( )

Returns uniform RNG

setSeed

public static void **setSeed** (int *seed*)

This resets the generators

Parameters

- **seed** –

6.34.5 TestObject

class **TestObject** implements Serializable

Test class for DeepCopy.

Fields

testField

public int **testField**

Constructors

TestObject

public **TestObject** (int *number*)

6.34.6 TiffParser

public class **TiffParser**

Parses the ImageStack into RAM out of a .tiff file.

Author Marcel Stefko

Methods

loadGeneralTiff

public final ImageStack **loadGeneralTiff** (File *file*)

Loads a tiff stack from a file on disk into RAM

Parameters

- **file** – tiff file to be loaded

Returns loaded image stack

6.35 ch.epfl.leb.sass.utils.images

6.35.1 ImageS

public interface **ImageS**

An abstraction layer for a 3-dimensional, 16-bit image stack in SASS. This interface allows developers to more easily substitute other backends for image data into SASS. For example, one could write an implementation for ImgLib2 datatypes to replace ImageJ's original ImageStack. This interface should be used everywhere image data is passed between SASS components.

Author Kyle M. Douglass

Methods

addImage

public void **addImage** (short[][] *image*)

Adds a single image to the dataset. This method accepts a 2D array of pixels and adds it to the end of the dataset.

The size of the image in X and Y must be the same as the existing images.

Parameters

- **image** – The image data to add to the dataset.

Throws

- *ch.epfl.leb.sass.utils.images.ImageShapeException* –

addImage

public void **addImage** (int[][] *image*)

Adds a single image to the dataset. This method accepts a 2D array of pixels and adds it to the end of the dataset. The size of the image in X and Y must be the same as the existing images. Integer data will be truncated into shorts.

Parameters

- **image** – The image data to add to the dataset.

Throws

- *ch.epfl.leb.sass.utils.images.ImageShapeException* –

addImage

public void **addImage** (float[][] *image*)

Adds a single image to the dataset. This method accepts a 2D array of pixels and adds it to the end of the dataset. The size of the image in X and Y must be the same as the existing images. Float data will be truncated into shorts.

Parameters

- **image** – The image data to add to the dataset.

Throws

- *ch.epfl.leb.sass.utils.images.ImageShapeException* –

concatenate

public void **concatenate** (*ImageS dataset*)

Appends another ImageS dataset to the end of this one.

Parameters

- **dataset** – The images to add to the dataset.

Throws

- *ch.epfl.leb.sass.utils.images.ImageShapeException* –

getBitDepth

public int **getBitDepth** ()

Returns the bit depth of the pixels.

Returns The bit depth of the pixels.

getHeight

public int **getHeight**()
Returns the height of the images in the dataset.

Returns The height of the images in the dataset.

getPixelData

public short[] **getPixelData**(int *index*)
Returns the image data at the slice corresponding to index.

Parameters

- **index** –

getSize

public int **getSize**()
Returns the number of images in the dataset.

Returns The number of images in the dataset.

getSlice

public int **getSlice**()
Gets the active slice of the dataset (0-indexed). This is the image that will be displayed in the viewer.

Returns The index of the current slice.

getTitle

public String **getTitle**()
Returns the title (or, equivalently, the name) of the image dataset.

Returns The title of the dataset.

getWidth

public int **getWidth**()
Returns the width of the images in the dataset.

Returns The width of the images in the dataset.

saveAsTiffStack

public void **saveAsTiffStack** (File *file*)

Saves the images to a TIFF file.

Parameters

- **file** – The TIFF file where the dataset will be saved.

serializeToArray

public byte[] **serializeToArray** ()

Serializes the dataset into a TIFF-encoded byte array.

Returns The image data encoded as a TIFF-file byte string.

serializeToBuffer

public ByteBuffer **serializeToBuffer** ()

Returns a buffer containing the dataset in a TIFF-encoded byte array.

Returns A ByteBuffer containing the TIFF-encoded dataset.

setSlice

public void **setSlice** (int *index*)

Sets the active slice of the dataset (0-indexed). * This is the image that will be displayed in the viewer.

Parameters

- **index** – The index of the slice to activate.

setTitle

public void **setTitle** (String *title*)

Sets the title (or, equivalently, the name) of the dataset.

Parameters

- **title** – The title to give to the image dataset.

updateView

public void **updateView** ()

Updates the dataset viewer to show the currently active slice.

view

public void **view** ()

Displays the images.

6.35.2 ImageShapeException

public class **ImageShapeException** extends Exception

Raised when trying to add data to ImageS datasets of the wrong XY shape.

Author Kyle M. Douglass

Constructors

ImageShapeException

public **ImageShapeException** ()

ImageShapeException

public **ImageShapeException** (String *message*)

6.36 ch.epfl.leb.sass.utils.images.internal

6.36.1 DefaultImageS

public class **DefaultImageS** implements *ImageS*

The default implementation of the ImageS interface. The default implementation currently wraps ImageJ1's ImageStack class. See https://imagej.nih.gov/ij/developer/api/ij/ImagePlus.html for more information.

Author Kyle M. Douglass

Constructors

DefaultImageS

public **DefaultImageS** (int *width*, int *height*)

Creates a new and empty DefaultImageS.

DefaultImageS

public **DefaultImageS** (int[][] *pixels*)

Creates a new DefaultImageS object from a 2D array of ints. The first index of the input array should correspond to x; the second corresponds to y.

Parameters

- **pixels** – The 2D array of pixel values.

DefaultImageS

public **DefaultImageS** (float[][] *pixels*)

Creates a new DefaultImageS object from a 2D array of floats. The first index of the input array should correspond to x; the second corresponds to y.

Parameters

- **pixels** – The 2D array of pixel values.

Methods

addImage

public void **addImage** (short[][] *image*)

Adds a 2D array of shorts to the dataset.

Parameters

- **image** – A 2D array of shorts.

addImage

public void **addImage** (int[][] *image*)

Converts a 2D array of ints to 16-bit shorts and adds it to the dataset.

Parameters

- **image** – A 2D array of ints indexed by xy.

Throws

- *ch.epfl.leb.sass.utils.images.ImageShapeException* –

addImage

public void **addImage** (float[][] *image*)

Converts a 2D array of floats to 16-bit shorts and adds it to the dataset.

Parameters

- **image** – A 2D array of floats indexed by xy.

Throws

- *ch.epfl.leb.sass.utils.images.ImageShapeException* –

concatenate

public void **concatenate** (*ImageS dataset*)

Appends another ImageS dataset to the end of this one.

Parameters

- **dataset** – The images to add to the dataset.

getBitDepth

public int **getBitDepth** ()

getHeight

public int **getHeight** ()

Returns the height of the images in the dataset.

Returns The height of the images in the dataset.

getPixelData

public short[] **getPixelData** (int *index*)

Returns the pixel data at the given index as a 1D array.

Parameters

- **index** – The index of the corresponding slice.

Returns The pixel data at the provided index.

getSize

public int **getSize** ()

Returns the number of images in the dataset.

Returns The number of images in the dataset.

getSlice

public int **getSlice** ()

Gets the active slice of the dataset (0-indexed). This is the image that will be displayed in the viewer.

Returns The index of the active slice.

getTitle

public String **getTitle** ()

Returns the title of the image stack.

Returns The title of the image stack.

getWidth

public int **getWidth** ()

Returns the width of the images in the dataset.

Returns The width of the images in the dataset.

saveAsTiffStack

public void **saveAsTiffStack** (File *file*)

Saves the images to a TIFF file.

serializeToArray

public byte[] **serializeToArray**()
Serializes the image stack to a TIFF-encoded byte array.

Returns A TIFF-encoded byte array.

serializeToBuffer

public ByteBuffer **serializeToBuffer**()
Returns a buffer containing the dataset in a TIFF-encoded byte array.

Returns A buffer containing the dataset in a TIFF-encoded byte array.

setSlice

public void **setSlice**(int *index*)
Sets the active slice of the dataset (0-indexed). This is the image that will be displayed in the viewer.

Parameters

- **index** – The index of the slice to activate.

setTitle

public void **setTitle**(String *title*)
Sets the title of the image stack.

Parameters

- **title** – The title of the image stack.

updateView

public void **updateView**()
Updates the dataset viewer to show the currently active slice.

view

public void **view**()
Displays the images in a ImagePlus window.

6.36.2 DefaultImageSTest

public class **DefaultImageSTest**
Test suite for DefaultImageS.

Author Kyle M. Douglass

Fields

instance

DefaultImageS **instance**

tempDir

public TemporaryFolder **tempDir**

Methods

setUp

public void **setUp**()

testAddImage_floatArrArr

public void **testAddImage_floatArrArr**()
Test of addImage method, of class DefaultImageS.

testAddImage_floatArrArr_wrongSize

public void **testAddImage_floatArrArr_wrongSize**()
Test of addImage method, of class DefaultImageS.

testAddImage_intArrArr

public void **testAddImage_intArrArr**()
Test of addImage method, of class DefaultImageS.

testAddImage_intArrArr_wrongSize

public void **testAddImage_intArrArr_wrongSize**()
Test of addImage method, of class DefaultImageS.

testAddImage_shortArrArr

public void **testAddImage_shortArrArr**()
Test of addImage method, of class DefaultImageS.

testAddImage_shortArrArr_wrongSize

public void **testAddImage_shortArrArr_wrongSize**()
Test of addImage method, of class DefaultImageS.

testConcatenate

public void **testConcatenate**()
Test of concatenate method, of class DefaultImageS.

testConcatenate_wrongSize

public void **testConcatenate_wrongSize**()
Test of concatenate method, of class DefaultImageS.

testGetBitDepth

public void **testGetBitDepth**()
Test of getBitDepth method, of class DefaultImageS.

testGetHeight

public void **testGetHeight**()
Test of getHeight method, of class DefaultImageS.

testGetPixelData

public void **testGetPixelData**()
Test of getPixelData method, of class DefaultImageS.

testGetSize

public void **testGetSize**()
Test of getSize method, of class DefaultImageS.

testGetSlice

public void **testGetSlice**()
Test of getSlice method, of class DefaultImageS.

testGetTitle

public void **testGetTitle**()
Test of getTitle method, of class DefaultImageS.

testGetWidth

public void **testGetWidth**()
Test of getWidth method, of class DefaultImageS.

testSaveAsTiffStack

public void **testSaveAsTiffStack**()
Test of saveAsTiffStack method, of class DefaultImageS.

testSaveAsTiffStackEmpty

public void **testSaveAsTiffStackEmpty**()
Test of saveAsTiffStack method, of class DefaultImageS.

testSerializeToArray

public void **testSerializeToArray**()
Test of serializeToArray method, of class DefaultImageS.

testSerializeToBuffer

public void **testSerializeToBuffer**()
Test of serializeToBuffer method, of class DefaultImageS.

testSetSlice

public void **testSetSlice**()
Test of setSlice method, of class DefaultImageS.

testSetTitle

public void **testSetTitle**()
Test of setTitle method, of class DefaultImageS.

CHAPTER 7

About

SASS is an open-source Fiji plugin for simulating super-resolution microscopy experiments and fluorophore photophysics.

CHAPTER 8

Acknowledgements

8.1 Authors

- Marcel Štefko
- Kyle M. Douglass
- Baptiste Ottino

CHAPTER 9

See Also

- ALICA - Automated Laser Illumination Control Algorithm

CHAPTER 10

Indices and tables

- genindex
- modindex
- search

Index

Symbols

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z