Collaborate

Version 1.0 July 6, 2019

ElectroScience Laboratory The Ohio State University

Sections of this document were assembled from comments in the source code using Doxygen documentation tools. All illustrations were created by the developers using COLLABORATE analysis software or associated software dependencies.

This research was sponsored by the NASA Advanced Information Systems Technology Program `NNH16ZDA001N-AIST` and took place at the ElectroScience Laboratory facility at The Ohio State University.

Contents

1 Python Script Execution

1.1 Animate Data

```
usage: animate_data.py [-h] [-f] [-i [IN_DIR]] [-o [OUT_DIR]] [-m [MONTH]]
                       [-v [VARIABLE]]

optional arguments:
  -h, --help            show this help message and exit
  -f, --figure          Whether to show or save
  -i [IN_DIR], --in_dir [IN_DIR]
                        Path to input directory
  -o [OUT_DIR], --out_dir [OUT_DIR]
                        Path to output directory
  -m [MONTH], --month [MONTH]
                        The month (2 digits)
  -v [VARIABLE], --variable [VARIABLE]
                        The netcdf variable name
```

1.2 Animate Map

```
usage: animate_map.py [-h] [-f] [-i [IN_FILE]] [-o [OUT_DIR]] [-d [DATA_DIR]]
                      [-t [TITLE]] [-w [WGT_PATH]] [-u [UWT_PATH]]

optional arguments:
  -h, --help            show this help message and exit
  -f, --figure          Whether to show or save
  -i [IN_FILE], --in_file [IN_FILE]
                        Path to input file
  -o [OUT_DIR], --out_dir [OUT_DIR]
                        Path to output directory
  -d [DATA_DIR], --data_dir [DATA_DIR]
                        Path to data directory
  -t [TITLE], --title [TITLE]
                        Title for the plot
  -w [WGT_PATH], --wgt_path [WGT_PATH]
                        Path to weighted adjacency list log
  -u [UWT_PATH], --uwt_path [UWT_PATH]
                        Path to unweighted adjacency list log
```

1.3 Animate Network

```
usage: animate_network.py [-h] [-f] [-i [IN_FILE]] [-o [OUT_FILE]] [-w]
```

```
optional arguments:
  -h, --help            show this help message and exit
  -f, --figure          Whether to show or save
  -i [IN_FILE], --in_file [IN_FILE]
                        Path to input file
  -o [OUT_FILE], --out_file [OUT_FILE]
                        Path to output directory
  -w, --weighted        Whether data is weighted
```

1.4 Plot Antenna

```
usage: plot_antenna.py [-h] [-f] [-i [IN_FILE]] [-o [OUT_DIR]]

optional arguments:
  -h, --help            show this help message and exit
  -f, --figure          Whether to show or save
  -i [IN_FILE], --in_file [IN_FILE]
                        Path to input file
  -o [OUT_DIR], --out_dir [OUT_DIR]
                        Path to output directory
```

1.5 Plot Channel

```
usage: plot_channel.py [-h] [-f] [-i [IN_DIR]] [-d [DATA_DIR]] [-o [OUT_DIR]]
                       [-n INDEX]

optional arguments:
  -h, --help            show this help message and exit
  -f, --figure          Whether to show or save
  -i [IN_DIR], --in_dir [IN_DIR]
                        Path to input directory
  -d [DATA_DIR], --data_dir [DATA_DIR]
                        Path to input directory
  -o [OUT_DIR], --out_dir [OUT_DIR]
                        Path to output directory
  -n INDEX, --index INDEX
                        The file index
```

1.6 Plot Data

```
usage: plot_data.py [-h] [-f] [-i [IN_DIR]] [-o [OUT_DIR]] [-n INDEX]
                    [-m [MONTH]] [-j [IN_DIR2]]
```

```
optional arguments:
  -h, --help            show this help message and exit
  -f, --figure          Whether to show or save
  -i [IN_DIR], --in_dir [IN_DIR]
                        Path to input directory
  -o [OUT_DIR], --out_dir [OUT_DIR]
                        Path to output directory
  -n INDEX, --index INDEX
                        The frame index
  -m [MONTH], --month [MONTH]
                        The month (2 digits)
  -j [IN_DIR2], --in_dir2 [IN_DIR2]
                        A second optional dataset
```

1.7 Plot Map

```
usage: plot_map.py [-h] [-f] [-i [IN_FILE]] [-o [OUT_DIR]] [-n INDEX]
                   [-w [WGT_PATH]] [-u [UWT_PATH]]

optional arguments:
  -h, --help            show this help message and exit
  -f, --figure          Whether to show or save
  -i [IN_FILE], --in_file [IN_FILE]
                        Path to input file
  -o [OUT_DIR], --out_dir [OUT_DIR]
                        Path to output directory
  -n INDEX, --index INDEX
                        The frame index
  -w [WGT_PATH], --wgt_path [WGT_PATH]
                        Path to weighted adjacency list log
  -u [UWT_PATH], --uwt_path [UWT_PATH]
                        Path to unweighted adjacency list log
```

1.8 Plot Measurement

```
usage: plot_measurement.py [-h] [-i [IN_DIR]] [-d [DATA_DIR]] [-o [OUT_DIR]]
                           [-n INDEX]

optional arguments:
  -h, --help            show this help message and exit
  -i [IN_DIR], --in_dir [IN_DIR]
                        Path to input directory
  -d [DATA_DIR], --data_dir [DATA_DIR]
                        Path to input directory
  -o [OUT_DIR], --out_dir [OUT_DIR]
```

```
                        Path to output directory
  -n INDEX, --index INDEX
                        The file index
```

1.9 Plot Network

```
usage: plot_network.py [-h] [-i [IN_FILE]] [-o [OUT_DIR]] [-n INDEX] [-w] [-f]

optional arguments:
  -h, --help            show this help message and exit
  -i [IN_FILE], --in_file [IN_FILE]
                        Path to input file
  -o [OUT_DIR], --out_dir [OUT_DIR]
                        Path to output directory
  -n INDEX, --index INDEX
                        The frame index
  -w, --weighted        Whether data is weighted
  -f, --figure          Whether to show or save
```

1.10 Plot Series

```
usage: plot_series.py [-h] [-i [IN_FILE]] [-o [OUT_DIR]] [-n INDEX] [-f]

optional arguments:
  -h, --help            show this help message and exit
  -i [IN_FILE], --in_file [IN_FILE]
                        Path to input file
  -o [OUT_DIR], --out_dir [OUT_DIR]
                        Path to output directory
  -n INDEX, --index INDEX
                        The frame index
  -f, --figure          Whether to show or save
```

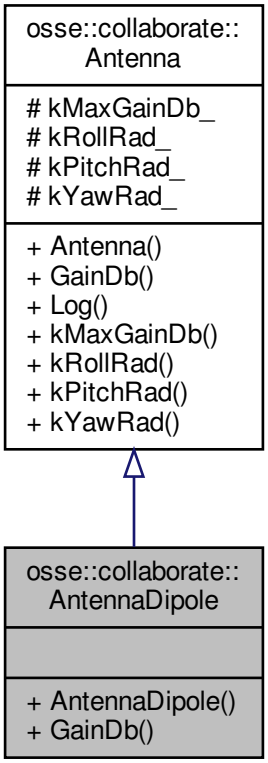
2 C++ Classes

2.1 osse::collaborate::AntennaDipole Class Reference

An approximation of a dipole antenna.

```
#include <antenna_dipole.h>
```

Inheritance diagram for osse::collaborate::AntennaDipole:



Public Member Functions

- AntennaDipole (const double &_max_gain_db, const double &_roll_rad, const double &_↩ pitch_rad, const double &_yaw_rad)

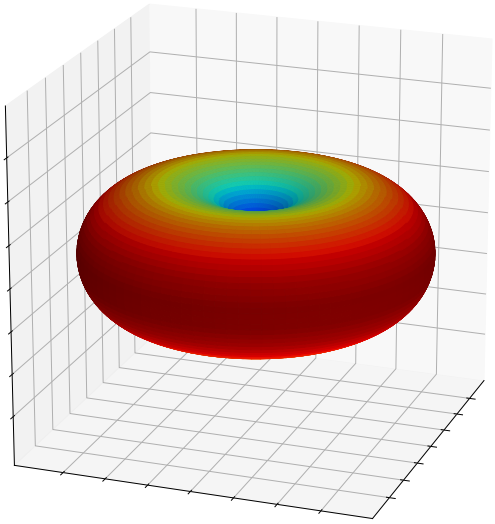
  Constructor.
- double GainDb (const double &_theta_rad, const double &_phi_rad) const

  Obtain directional gain (decibels)

Additional Inherited Members

2.1.1 Detailed Description

An approximation of a dipole antenna.



$$g = g_{max} \sin^2 \theta \quad (decibels)$$

2.1.2 Constructor & Destructor Documentation

2.1.2.1 AntennaDipole()

```
osse::collaborate::AntennaDipole::AntennaDipole (
            const double & _max_gain_db,
            const double & _roll_rad,
            const double & _pitch_rad,
            const double & _yaw_rad )
```

Constructor.

Parameters

in	*_max_gain_db*	The maximum gain (decibels)
in	*_roll_rad*	Roll angle to host body frame (radians)
in	*_pitch_rad*	Pitch angle to host body frame (radians)
in	*_yaw_rad*	Yaw angle to host body frame (radians)

2.1.3 Member Function Documentation

2.1.3.1 GainDb()

```
double osse::collaborate::AntennaDipole::GainDb (
            const double & _theta_rad,
            const double & _phi_rad ) const  [virtual]
```

Obtain directional gain (decibels)

Parameters

in	*_theta_rad*	Altitude angle from positive z-axis (radians)
in	*_phi_rad*	Azimuth angle (radians)

Returns

Directional gain (decibels)

$$g = g_{max} \sin^2 \theta \quad (decibels)$$

Implements osse::collaborate::Antenna.

The documentation for this class was generated from the following files:

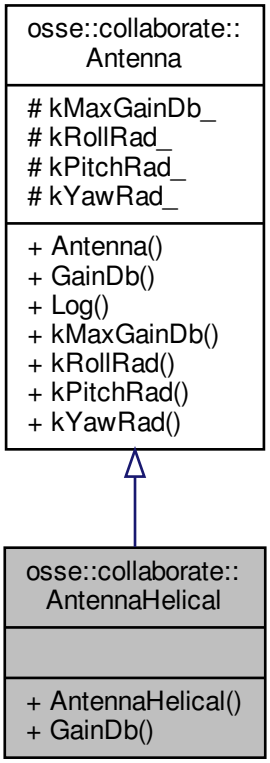
- libs/collaborate/include/collaborate/antenna_dipole.h
- libs/collaborate/src/antenna_dipole.cpp

2.2 osse::collaborate::AntennaHelical Class Reference

An approximation of a helical antenna (more directional than patch)

```
#include <antenna_helical.h>
```

Inheritance diagram for osse::collaborate::AntennaHelical:



Public Member Functions

- AntennaHelical (const double &_max_gain_db, const double &_roll_rad, const double &_↩ pitch_rad, const double &_yaw_rad)

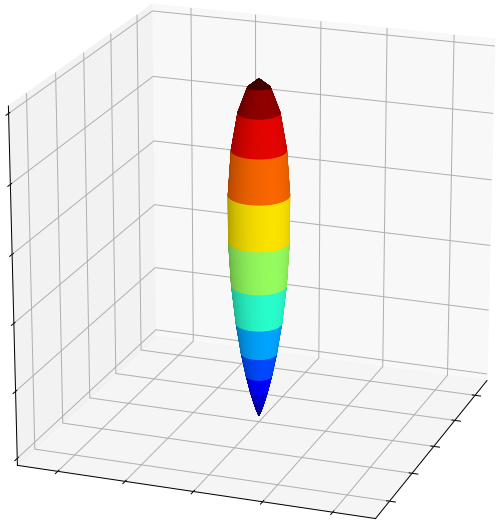
  Constructor.
- double GainDb (const double &_theta_rad, const double &_phi_rad) const

  Obtain directional gain (decibels)

Additional Inherited Members

2.2.1 Detailed Description

An approximation of a helical antenna (more directional than patch)



$$g = g_{max} \cos^{50} \theta \quad (decibels)$$

2.2.2 Constructor & Destructor Documentation

2.2.2.1 AntennaHelical()

```
osse::collaborate::AntennaHelical::AntennaHelical (
            const double & _max_gain_db,
            const double & _roll_rad,
            const double & _pitch_rad,
            const double & _yaw_rad )
```

Constructor.

Parameters

in	*_max_gain_db*	Maximum gain (decibels)
in	*_roll_rad*	Roll angle to host body reference frame (radians)
in	*_pitch_rad*	Pitch angle to host body reference frame (radians)
in	*_yaw_rad*	Yaw angle to host body reference frame (radians)

2.2.3 Member Function Documentation

2.2.3.1 GainDb()

```
double osse::collaborate::AntennaHelical::GainDb (
            const double & _theta_rad,
            const double & _phi_rad ) const  [virtual]
```

Obtain directional gain (decibels)

Parameters

in	*_theta_rad*	Altitude angle from positive z-axis (radians)
in	*_phi_rad*	Azimuth angle (radians)

Returns

Directional gain (decibels)

$$g = g_{max} \cos^{50} \theta \quad (decibels)$$

Implements osse::collaborate::Antenna.

The documentation for this class was generated from the following files:

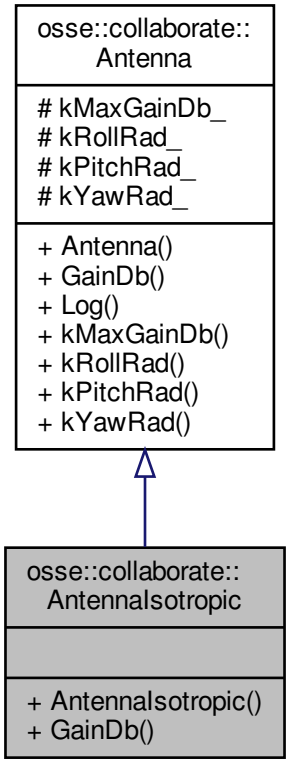
- libs/collaborate/include/collaborate/antenna_helical.h
- libs/collaborate/src/antenna_helical.cpp

2.3 osse::collaborate::AntennaIsotropic Class Reference

An isotropic antenna.

```
#include <antenna_isotropic.h>
```

Inheritance diagram for osse::collaborate::AntennaIsotropic:



Public Member Functions

- AntennaIsotropic (const double &_max_gain_db)

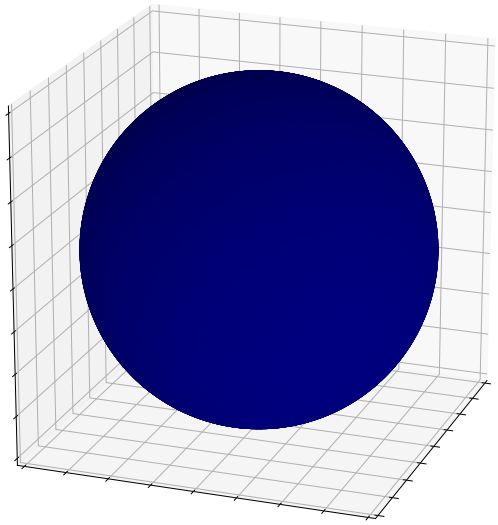
  Constructor.
- double GainDb (const double &_theta_rad, const double &_phi_rad) const

  Obtain directional gain (decibels)

Additional Inherited Members

2.3.1 Detailed Description

An isotropic antenna.



$$g = g_{max} \quad (decibels)$$

2.3.2 Constructor & Destructor Documentation

2.3.2.1 AntennaIsotropic()

```
osse::collaborate::AntennaIsotropic::AntennaIsotropic (
            const double & _max_gain_db ) [explicit]
```

Constructor.

Parameters

in	*_max_gain_db*	Maximum gain (decibels)

2.3.3 Member Function Documentation

2.3.3.1 GainDb()

```
double osse::collaborate::AntennaIsotropic::GainDb (
            const double & _theta_rad,
            const double & _phi_rad ) const  [virtual]
```

Obtain directional gain (decibels)

Parameters

in	*_theta_rad*	Altitude angle from positive z-axis (radians)
in	*_phi_rad*	Azimuth angle (radians)

Returns

Directional gain (decibels)

$$g = g_{max} \quad (decibels)$$

Implements osse::collaborate::Antenna.

The documentation for this class was generated from the following files:

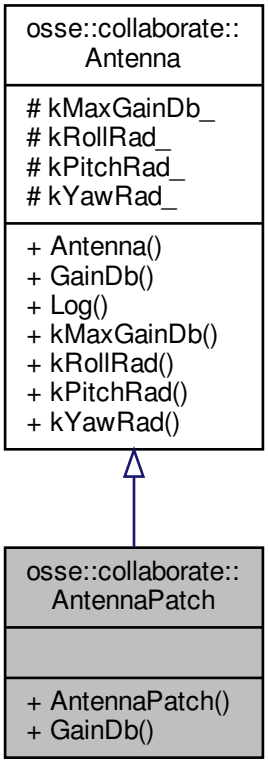
- libs/collaborate/include/collaborate/antenna_isotropic.h
- libs/collaborate/src/antenna_isotropic.cpp

2.4 osse::collaborate::AntennaPatch Class Reference

An approximation of a patch antenna.

```
#include <antenna_patch.h>
```

Inheritance diagram for osse::collaborate::AntennaPatch:



Public Member Functions

- AntennaPatch (const double &_max_gain_db, const double &_roll_rad, const double &_↩ pitch_rad, const double &_yaw_rad)

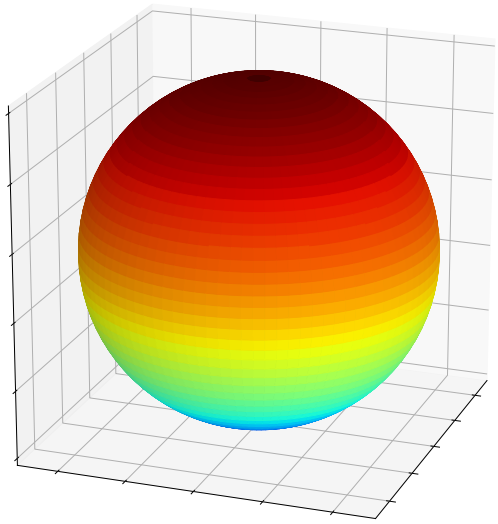
  Constructor.
- double GainDb (const double &_theta_rad, const double &_phi_rad) const

  Obtain directional gain (decibels)

Additional Inherited Members

2.4.1 Detailed Description

An approximation of a patch antenna.



$$g = g_{max} \cos\theta \quad (decibels), \qquad 0 < \theta \leq \pi/2$$

2.4.2 Constructor & Destructor Documentation

2.4.2.1 AntennaPatch()

```
osse::collaborate::AntennaPatch::AntennaPatch (
            const double & _max_gain_db,
            const double & _roll_rad,
            const double & _pitch_rad,
            const double & _yaw_rad )
```

Constructor.

Parameters

in	*_max_gain_db*	Maximum gain (decibels)
in	*_roll_rad*	Roll angle to host body reference frame (radians)
in	*_pitch_rad*	Pitch angle to host body reference frame (radians)
in	*_yaw_rad*	Yaw angle to host body reference frame (radians)

2.4.3 Member Function Documentation

2.4.3.1 GainDb()

```
double osse::collaborate::AntennaPatch::GainDb (
            const double & _theta_rad,
            const double & _phi_rad ) const  [virtual]
```

Obtain directional gain (decibels)

Parameters

in	*_theta_rad*	Altitude angle from positive z-axis (radians)
in	*_phi_rad*	Azimuth angle (radians)

Returns

Directional gain (decibels)

$$g = g_{max} \cos\theta \quad (decibels), \qquad 0 < \theta \leq \pi/2$$

Implements osse::collaborate::Antenna.

The documentation for this class was generated from the following files:

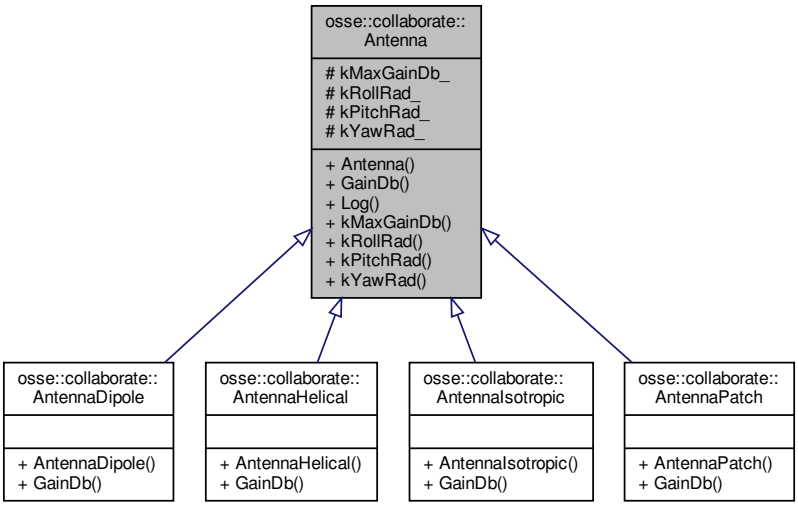
- libs/collaborate/include/collaborate/antenna_patch.h
- libs/collaborate/src/antenna_patch.cpp

2.5 osse::collaborate::Antenna Class Reference

Abstract antenna.

```
#include <antenna.h>
```

Inheritance diagram for osse::collaborate::Antenna:



Public Member Functions

- Antenna (const double &_max_gain_db, const double &_roll_rad, const double &_pitch_rad, const double &_yaw_rad)

  Constructor.
- virtual double GainDb (const double &_theta_rad, const double &_phi_rad) const =0

  Obtain directional gain (decibels)
- void Log (const std::string &_path) const

  Logs the antenna pattern to a file.
- const double & kMaxGainDb () const

  Get maximum gain (decibels)
- const double & kRollRad () const

  Get roll angle to host body reference frame (radians)
- const double & kPitchRad () const

  Get pitch angle to host body reference frame (radians)
- const double & kYawRad () const

  Get yaw angle to host body reference frame (radians)

Protected Attributes

- const double kMaxGainDb_

  Maximum gain (decibels)
- const double kRollRad_

Roll angle to host body reference frame (radians)

- const double kPitchRad_

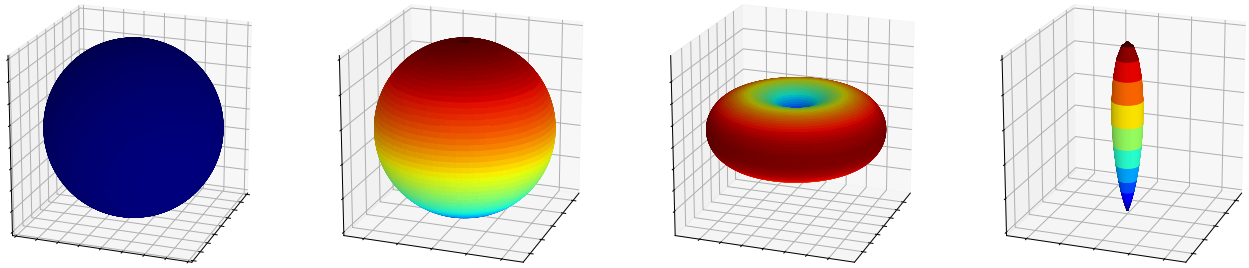
  Pitch angle to host body reference frame (radians)

- const double kYawRad_

  Yaw angle to host body reference frame (radians)

2.5.1 Detailed Description

Abstract antenna.



2.5.2 Constructor & Destructor Documentation

2.5.2.1 Antenna()

```
osse::collaborate::Antenna::Antenna (
            const double & _max_gain_db,
            const double & _roll_rad,
            const double & _pitch_rad,
            const double & _yaw_rad )
```

Constructor.

Parameters

in	*_max_gain_db*	Maximum gain (decibels)
in	*_roll_rad*	Roll angle to host body reference frame (radians)
in	*_pitch_rad*	Pitch angle to host body reference frame (radians)
in	*_yaw_rad*	Yaw angle to host body reference frame (radians)

2.5.3 Member Function Documentation

2.5.3.1 GainDb()

```
virtual double osse::collaborate::Antenna::GainDb (
            const double & _theta_rad,
            const double & _phi_rad ) const  [pure virtual]
```

Obtain directional gain (decibels)

Parameters

in	*_theta_rad*	Altitude angle from positive z-axis (radians)
in	*_phi_rad*	Azimuth angle (radians)

Returns

Directional gain (decibels)

Implemented in osse::collaborate::AntennaDipole, osse::collaborate::AntennaHelical, osse::collaborate::AntennaPatch, and osse::collaborate::AntennaIsotropic.

2.5.3.2 kMaxGainDb()

```
const double& osse::collaborate::Antenna::kMaxGainDb ( ) const  [inline]
```

Get maximum gain (decibels)

Returns

max_gain_db_ Maximum gain (decibels)

2.5.3.3 kPitchRad()

```
const double& osse::collaborate::Antenna::kPitchRad ( ) const  [inline]
```

Get pitch angle to host body reference frame (radians)

Returns

pitch_rad_ Pitch angle to host body reference frame (radians)

2.5.3.4 kRollRad()

```
const double& osse::collaborate::Antenna::kRollRad ( ) const  [inline]
```

Get roll angle to host body reference frame (radians)

Returns

roll_rad_ Roll angle to host body reference frame (radians)

2.5.3.5 kYawRad()

```
const double& osse::collaborate::Antenna::kYawRad ( ) const  [inline]
```

Get yaw angle to host body reference frame (radians)

Returns

yaw_rad_ Yaw angle to host body reference frame (radians)

2.5.3.6 Log()

```
void osse::collaborate::Antenna::Log (
            const std::string & _path ) const
```

Logs the antenna pattern to a file.

Parameters

in	*_path*	File path

The documentation for this class was generated from the following files:

- libs/collaborate/include/collaborate/antenna.h
- libs/collaborate/src/antenna.cpp

2.6 osse::collaborate::AttitudeMatrix Class Reference

A 2D (3x3) matrix for reference frame and attitude transformations.

`#include <attitude_matrix.h>`

Public Types

- typedef std::array< double, kColumns > Row

  A row if an Array.
- typedef std::array< Row, kColumns > Array

  Underlying (wrapped) array of matrix.

Public Member Functions

- AttitudeMatrix (double _r0c0, double _r0c1, double _r0c2, double _r1c0, double _r1c1, double _r1c2, double _r2c0, double _r2c1, double _r2c2)

  Constructor from explicit values.
- AttitudeMatrix (const Vector &_x_axis, const Vector &_y_axis, const Vector &_z_axis)

  Constructor From Axes.
- AttitudeMatrix (const double &_roll_rad, const double &_pitch_rad, const double &_yaw_rad)

  Constructor From Angles.
- Vector TransformVector (const Vector &_vector) const

  Transforms a vector to a new coordinate system.
- Vector InvertVector (const Vector &_vector) const

  Performs an inverse transformation on a vector.
- std::string ToString () const

  Outputs the matrix to a string.

Static Public Attributes

- static constexpr uint8_t kColumns = 3

  The number of columns in an Array.

Private Member Functions

- Row operator[] (int _column) const

  Overloads the [] operator to access a row.
- Array AttitudeMatrixFromAxes (const Vector &_x_axis, const Vector &_y_axis, const Vector &_z_axis) const

  Calculate From Axes.
- Array AttitudeMatrixFromAngles (const double &_roll_rad, const double &_pitch_rad, const double &_yaw_rad) const

  Calculate From Angles.
- Array InverseOfAttitudeMatrix () const

  Calculate the inverse matrix.
- double AttitudeMatrixDeterminant () const

  Calculate the determinant of the matrix.

Private Attributes

- Array m_
  AttitudeMatrix.
- Array i_
  Inverse matrix.

2.6.1 Detailed Description

A 2D (3x3) matrix for reference frame and attitude transformations.

$$\mathbf{M} = \begin{bmatrix} m_{0,0} & m_{0,1} & m_{0,2} \\ m_{1,0} & m_{1,1} & m_{1,2} \\ m_{2,0} & m_{2,1} & m_{2,2} \end{bmatrix}$$

$$\mathbf{I} = \mathbf{M}^{-1}$$

2.6.2 Constructor & Destructor Documentation

2.6.2.1 AttitudeMatrix() [1/3]

```
osse::collaborate::AttitudeMatrix::AttitudeMatrix (
            double _r0c0,
            double _r0c1,
            double _r0c2,
            double _r1c0,
            double _r1c1,
            double _r1c2,
            double _r2c0,
            double _r2c1,
            double _r2c2 )
```

Constructor from explicit values.

Parameters

in	*_r0c0*	Row 0, Column 0
in	*_r0c1*	Row 0, Column 1
in	*_r0c2*	Row 0, Column 2
in	*_r1c0*	Row 1, Column 0
in	*_r1c1*	Row 1, Column 1
in	*_r1c2*	Row 1, Column 2
in	*_r2c0*	Row 2, Column 0
in	*_r2c1*	Row 2, Column 1

$$\mathbf{M} = \begin{bmatrix} r_0c_0 & r_0c_1 & r_0c_2 \\ r_1c_0 & r_1c_1 & r_1c_2 \\ r_2c_0 & r_2c_1 & r_2c_2 \end{bmatrix}$$

2.6.2.2 AttitudeMatrix() [2/3]

```
osse::collaborate::AttitudeMatrix::AttitudeMatrix (
            const Vector & _x_axis,
            const Vector & _y_axis,
            const Vector & _z_axis )
```

Constructor From Axes.

Parameters

in	*_x_axis*	X axis
in	*_y_axis*	Y axis
in	*_z_axis*	Z axis

$$X = Roll\ Axis$$
$$Y = Pitch\ Axis$$
$$Z = Yaw\ Axis$$
$$\psi = Roll = \sin^{-1} -Y_Z \pmod{\pi} \quad (radians)$$
$$\theta = Pitch = \tan^{-1} \frac{X_Z}{Z_Z} \pmod{2\pi} \quad (radians)$$
$$\phi = Yaw = \tan^{-1} \frac{Y_X}{Y_Y} \pmod{2\pi} \quad (radians)$$
$$\mathbf{M} = \begin{bmatrix} \cos\psi\cos\theta + \sin\psi\sin\phi\sin\theta & \sin\psi\cos\phi & -\cos\psi\sin\theta + \sin\psi\sin\phi\cos\theta \\ -\sin\psi\cos\theta + \cos\psi\sin\phi\sin\theta & \cos\psi\cos\phi & \sin\psi\sin\theta + \cos\psi\sin\phi\cos\theta \\ \cos\psi\sin\phi & -\sin\phi & \cos\phi\cos\theta \end{bmatrix}$$

2.6.2.3 AttitudeMatrix() [3/3]

```
osse::collaborate::AttitudeMatrix::AttitudeMatrix (
            const double & _roll_rad,
            const double & _pitch_rad,
            const double & _yaw_rad )
```

Constructor From Angles.

Parameters

in	*_roll_rad*	Roll angle about the x axis (radians)
in	*_pitch_rad*	Pitch angle about the x axis (radians)
in	*_yaw_rad*	Yaw angle about the x axis (radians)

$$\psi = Roll \quad (radians)$$
$$\theta = Pitch \quad (radians)$$
$$\phi = Yaw \quad (radians)$$
$$\mathbf{M} = \begin{bmatrix} \cos\psi\cos\theta + \sin\psi\sin\phi\sin\theta & \sin\psi\cos\phi & -\cos\psi\sin\theta + \sin\psi\sin\phi\cos\theta \\ -\sin\psi\cos\theta + \cos\psi\sin\phi\sin\theta & \cos\psi\cos\phi & \sin\psi\sin\theta + \cos\psi\sin\phi\cos\theta \\ \cos\psi\sin\phi & -\sin\phi & \cos\phi\cos\theta \end{bmatrix}$$

2.6.3 Member Function Documentation

2.6.3.1 AttitudeMatrixDeterminant()

```
double osse::collaborate::AttitudeMatrix::AttitudeMatrixDeterminant ( ) const  [private]
```

Calculate the determinant of the matrix.

Returns

Determinant

$$\det \mathbf{M} =$$
$$(m_{0,0}(m_{1,1}m_{2,2} - m_{1,2}m_{2,1}))$$
$$-(m_{0,1}(m_{1,0}m_{2,2} - m_{2,0}m_{1,2}))$$
$$+(m_{0,2}(m_{1,0}m_{2,1} - m_{1,1}m_{2,0}))$$

2.6.3.2 AttitudeMatrixFromAngles()

```
AttitudeMatrix::Array osse::collaborate::AttitudeMatrix::AttitudeMatrixFromAngles (
            const double & _roll_rad,
            const double & _pitch_rad,
            const double & _yaw_rad ) const  [private]
```

Calculate From Angles.

Parameters

in	*_roll_rad*	Roll angle about the x axis (radians)
in	*_pitch_rad*	Pitch angle about the x axis (radians)
in	*_yaw_rad*	Yaw angle about the x axis (radians)

Returns

An attitude matrix calculated from angles

$$\psi = Roll \quad (radians)$$
$$\theta = Pitch \quad (radians)$$
$$\phi = Yaw \quad (radians)$$
$$\mathbf{M} = \begin{bmatrix} \cos\psi\cos\theta + \sin\psi\sin\phi\sin\theta & \sin\psi\cos\phi & -\cos\psi\sin\theta + \sin\psi\sin\phi\cos\theta \\ -\sin\psi\cos\theta + \cos\psi\sin\phi\sin\theta & \cos\psi\cos\phi & \sin\psi\sin\theta + \cos\psi\sin\phi\cos\theta \\ \cos\psi\sin\phi & -\sin\phi & \cos\phi\cos\theta \end{bmatrix}$$

2.6.3.3 AttitudeMatrixFromAxes()

```
AttitudeMatrix::Array osse::collaborate::AttitudeMatrix::AttitudeMatrixFromAxes (
            const Vector & _x_axis,
            const Vector & _y_axis,
            const Vector & _z_axis ) const  [private]
```

Calculate From Axes.

Parameters

in	*_x_axis*	X axis
in	*_y_axis*	Y axis
in	*_z_axis*	Z axis

Returns

An attitude matrix calculated from axes

$$X = Roll\ Axis$$
$$Y = Pitch\ Axis$$
$$Z = Yaw\ Axis$$
$$\psi = Roll = \sin^{-1} -Y_Z \pmod{\pi} \quad (radians)$$

$$\theta = Pitch = \tan^{-1}\frac{X_Z}{Z_Z} \pmod{2\pi} \quad (radians)$$

$$\phi = Yaw = \tan^{-1}\frac{Y_X}{Y_Y} \pmod{2\pi} \quad (radians)$$

$$\mathbf{M} = \begin{bmatrix} \cos\psi\cos\theta + \sin\psi\sin\phi\sin\theta & \sin\psi\cos\phi & -\cos\psi\sin\theta + \sin\psi\sin\phi\cos\theta \\ -\sin\psi\cos\theta + \cos\psi\sin\phi\sin\theta & \cos\psi\cos\phi & \sin\psi\sin\theta + \cos\psi\sin\phi\cos\theta \\ \cos\psi\sin\phi & -\sin\phi & \cos\phi\cos\theta \end{bmatrix}$$

2.6.3.4 InverseOfAttitudeMatrix()

`AttitudeMatrix::Array osse::collaborate::AttitudeMatrix::InverseOfAttitudeMatrix ( ) const [private]`

Calculate the inverse matrix.

Returns

Inverse matrix

$$\mathbf{I} = \left(\frac{1}{\det\mathbf{M}}\right) \begin{bmatrix} m_{1,1}m_{2,2} - m_{1,2}m_{2,1} & m_{0,2}m_{2,1} - m_{0,1}m_{2,2} & m_{0,1}m_{1,2} - m_{0,2}m_{1,1} \\ m_{1,2}m_{2,0} - m_{1,0}m_{2,2} & m_{0,0}m_{2,2} - m_{0,2}m_{2,0} & m_{0,2}m_{1,0} - m_{0,0}m_{1,2} \\ m_{1,0}m_{2,1} - m_{1,1}m_{2,0} & m_{0,1}m_{2,0} - m_{0,0}m_{2,1} & m_{0,0}m_{1,1} - m_{0,1}m_{1,0} \end{bmatrix}$$

2.6.3.5 InvertVector()

`Vector osse::collaborate::AttitudeMatrix::InvertVector ( const Vector & _vector ) const [inline]`

Performs an inverse transformation on a vector.

Parameters

in	*_vector*	Vector

Returns

Inverted vector

$$\mathbf{I}\vec{v} = \mathbf{M}^{-1}\begin{bmatrix} v_{0,0} \\ v_{1,0} \\ v_{2,0} \end{bmatrix} = \vec{i} = \begin{bmatrix} i_{0,0} \\ i_{1,0} \\ i_{2,0} \end{bmatrix}$$

2.6.3.6 operator[]()

```
Row osse::collaborate::AttitudeMatrix::operator[] (
            int _column ) const  [inline], [private]
```

Overloads the [] operator to access a row.

Parameters

in	*_column*	The number of the column

Returns

Row

2.6.3.7 ToString()

```
std::string osse::collaborate::AttitudeMatrix::ToString ( ) const
```

Outputs the matrix to a string.

Returns

AttitudeMatrix as a string

2.6.3.8 TransformVector()

```
Vector osse::collaborate::AttitudeMatrix::TransformVector (
            const Vector & _vector ) const  [inline]
```

Transforms a vector to a new coordinate system.

Parameters

in	*_vector*	Vector

Returns

Transformed vector

$$\mathbf{M}\vec{v} = \begin{bmatrix} M_{0,0} & M_{0,1} & M_{0,2} \\ M_{1,0} & M_{1,1} & M_{1,2} \\ M_{2,0} & M_{2,1} & M_{2,2} \end{bmatrix} \begin{bmatrix} v_{0,0} \\ v_{1,0} \\ v_{2,0} \end{bmatrix} = \vec{t} = \begin{bmatrix} t_{0,0} \\ t_{1,0} \\ t_{2,0} \end{bmatrix}$$

The documentation for this class was generated from the following files:

- libs/collaborate/include/collaborate/attitude_matrix.h
- libs/collaborate/src/attitude_matrix.cpp

2.7 osse::collaborate::Battery Class Reference

A battery.

`#include <battery.h>`

Public Member Functions

- Battery (const double &_cell_amp_hr, const double &_num_cells, const double &_voltage_v, const double &_charging_efficiency_percent)

  Constructor.
- void IntroduceEnergy (const double &_energy_w_hr)

  Add or subtract from total energy.
- const double & energy_w_hr () const

  Get Energy stored in the battery (Watt hours)

Private Attributes

- const double kCapacityWHr_

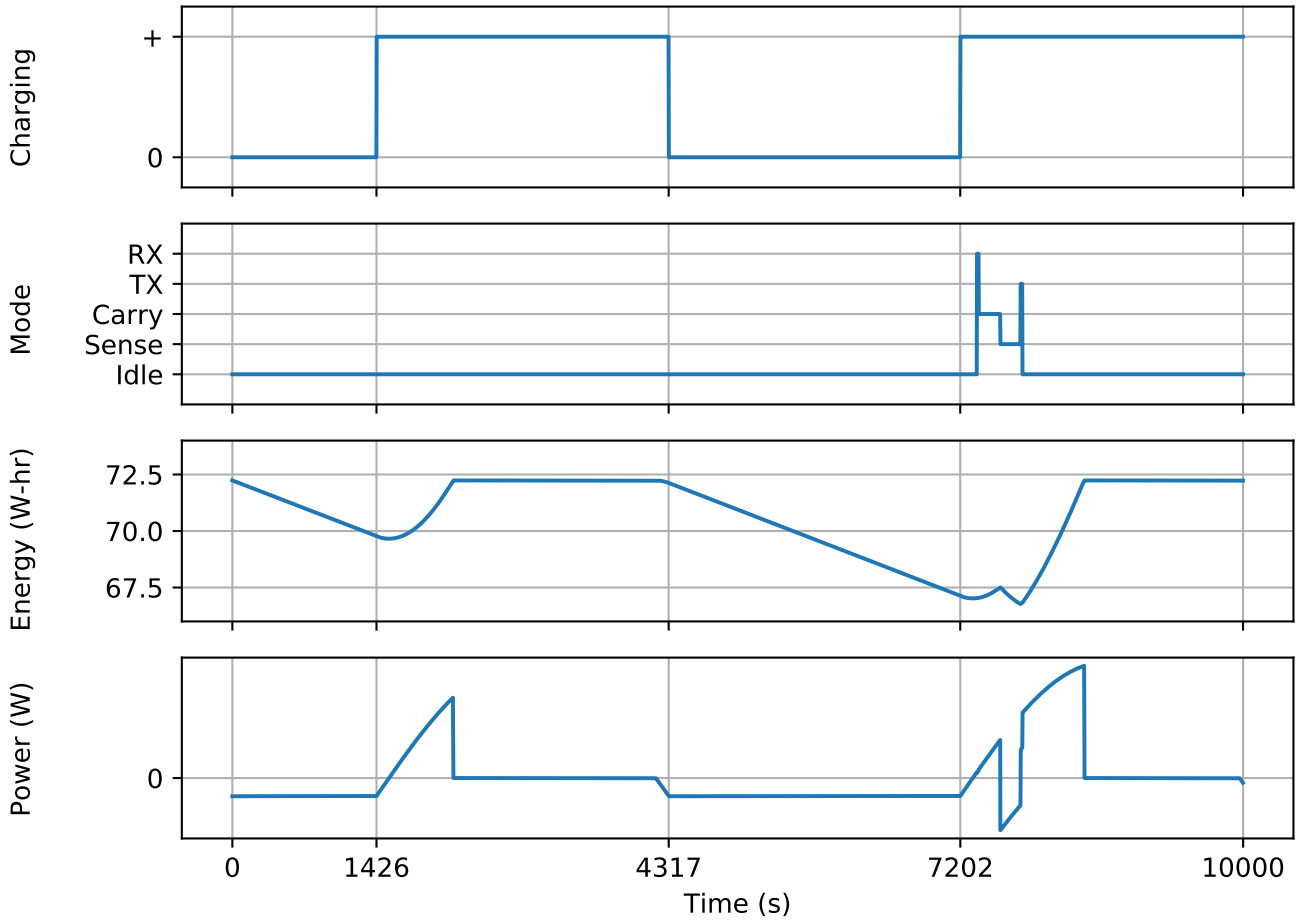
  Capacity of battery (Watt hours)
- const double kChargeEfficiencyPercent_

  Charge efficiency (percent)
- double energy_w_hr_

  Energy stored in battery (Watt hours)

2.7.1 Detailed Description

A battery.



2.7.2 Constructor & Destructor Documentation

2.7.2.1 Battery()

```
osse::collaborate::Battery::Battery (
            const double & _cell_amp_hr,
            const double & _num_cells,
            const double & _voltage_v,
            const double & _charging_efficiency_percent )
```

Constructor.

Parameters

in	*_cell_amp_hr*	Cell capacity (Amp hours)
in	*_num_cells*	Number of cells in battery
in	*_voltage_v*	Full charge voltage (Volts)
in	*_charging_efficiency_percent*	Charge efficiency (percent)

$$Q_{capacity} = n_{cells} * Q_{cell} * V \quad (Watt \cdot hour)$$

2.7.3 Member Function Documentation

2.7.3.1 energy_w_hr()

```
const double& osse::collaborate::Battery::energy_w_hr ( ) const  [inline]
```

Get Energy stored in the battery (Watt hours)

Returns

energy_watt_hr_ Energy stored in the battery (Watt hours)

2.7.3.2 IntroduceEnergy()

```
void osse::collaborate::Battery::IntroduceEnergy (
            const double & _energy_w_hr )
```

Add or subtract from total energy.

Parameters

in	*_energy_w_hr*	Energy (Watt hours)

The documentation for this class was generated from the following files:

- libs/collaborate/include/collaborate/battery.h
- libs/collaborate/src/battery.cpp

2.8 osse::collaborate::Channel Class Reference

Describes properties of a communication channel between nodes.

`#include <channel.h>`

Data Structures

- struct LogBuffer

  A buffer for logged node data.

Public Types

- typedef struct osse::collaborate::Channel::LogBuffer LogBuffer

  A buffer for logged node data.

Public Member Functions

- Channel (Node *_tx_node, Node *_rx_node)

  Constructor.
- void Start ()

  Starts fake transmission of data.
- void Update (const SimulationClock &_clock)

  Updates member variables and performs fake transfers.
- uint64_t PredictTransferDurationS () const

  Calculates expected transfer duration (seconds)
- uint64_t PredictTransferSizeBytes () const

  Calculates Expected transfer size (bytes)
- Node * tx_node () const

  Get transmitter.
- Node * rx_node () const

  Get receiver.
- const double & rx_gain_db () const

  Get received gain (decibels)
- const double & tx_gain_db () const

  Get transmitted gain (decibels)
- const bool & active () const

  Get activity.
- const bool & error_flag () const

  Get error status.
- const bool & success_flag () const

  Get success status.
- const bool & open () const

  Get potential contact status.

Static Public Attributes

- static constexpr double kSpeedOfLightMPerS = 299792458.0

  Speed of light in a vacuum (meters per second)

Private Member Functions

- void FakeTransfer (const SimulationClock &_clock)

  Uses fake data buffers to simulate the movement of data.
- void RealTransfer ()

  Moves data from real transmit buffer to real receive buffer.
- void Buffer (const SimulationClock &_clock)

  Appends data to log buffer.
- void Flush (const SimulationClock &_clock)

  Writes the log buffer to file.
- double CalculateDataRateBitsPerS () const

  Calculates data rate (bits per second)
- void UpdateOmegaRadPerS ()

  Calculates frequency (radians per second)
- void UpdateDistanceM ()

  Updates distance (meters)
- void UpdateLosUnit ()

  Updates line-of-sight unit vectors.
- void UpdateLosSpeedMPerS ()

  Updates line-of-sight speed (meters per second)
- void UpdateGainDb ()

  Updates directional gain values (decibels)
- void UpdateOpen ()

  Updates potential contact status.
- void UpdatePowerW ()

  Updates power (Watts)
- void UpdateDelayS ()

  Updates delay (seconds)

Private Attributes

- Node * tx_node_

  Transmitter.
- Node * rx_node_

  Receiver.
- double data_rate_bits_per_s_

  Data rate (bits per second)
- double omega_rad_per_s_

  Frequency (radians per second)
- double rx_power_w_

  Received power (Watts)
- double rx_gain_db_

  Received gain (decibels)
- Vector rx_los_unit_

  Received line-of-sight vector (unit)
- double tx_power_w_

  Transmitted power (watts)
- double tx_gain_db_

  Transmitted gain (decibels)
- Vector tx_los_unit_

  Transmitted line-of-sight vector (unit)
- double los_speed_m_per_s_

  Line-of-sight speed (meters per second)
- double distance_m_

  Distance (meters)
- double delay_s_

  Delay (seconds)
- bool active_

  Activity status.
- uint64_t fake_rx_buffer_bytes_

  Fake receive buffer size (bytes)
- uint64_t fake_tx_buffer_bytes_

  Fake transmit buffer size (bytes)
- std::vector< double > log_buffer_

  Buffer for log data.
- bool error_flag_

  Error status.
- bool success_flag_

  Success status.
- bool open_

Potential contact status.

- LogBuffer log_

  Buffer for logged communication parameters.

2.8.1 Detailed Description

Describes properties of a communication channel between nodes.

2.8.2 Constructor & Destructor Documentation

2.8.2.1 Channel()

```
osse::collaborate::Channel::Channel (
            Node * _tx_node,
            Node * _rx_node )
```

Constructor.

Parameters

in	*_tx_node*	Transmitter
in	*_rx_node*	Receiver

2.8.3 Member Function Documentation

2.8.3.1 active()

```
const bool& osse::collaborate::Channel::active ( ) const  [inline]
```

Get activity.

Returns

active_ activity

2.8.3.2 Buffer()

```
void osse::collaborate::Channel::Buffer (
            const SimulationClock & _clock )  [private]
```

Appends data to log buffer.

Parameters

in	*_clock*	Simulation clock

2.8.3.3 CalculateDataRateBitsPerS()

```
double osse::collaborate::Channel::CalculateDataRateBitsPerS ( ) const  [private]
```

Calculates data rate (bits per second)

Returns

Data rate (bits per second)

$$\delta = \min\left(\delta_{tx}, \delta_{rx}\right) \quad (bits\ per\ second)$$

2.8.3.4 error_flag()

```
const bool& osse::collaborate::Channel::error_flag ( ) const  [inline]
```

Get error status.

Returns

error_flag_ Error status

2.8.3.5 FakeTransfer()

```
void osse::collaborate::Channel::FakeTransfer (
            const SimulationClock & _clock )  [private]
```

Uses fake data buffers to simulate the movement of data.

Parameters

in	*_clock*	Simulation clock

2.8.3.6 Flush()

```
void osse::collaborate::Channel::Flush (
            const SimulationClock & _clock )  [private]
```

Writes the log buffer to file.

Parameters

in	*_clock*	Simulation clock

2.8.3.7 open()

```
const bool& osse::collaborate::Channel::open ( ) const  [inline]
```

Get potential contact status.

Returns

open_ Potential contact status

2.8.3.8 PredictTransferDurationS()

```
uint64_t osse::collaborate::Channel::PredictTransferDurationS ( ) const
```

Calculates expected transfer duration (seconds)

Returns

Expected transfer duration (seconds)

2.8.3.9 PredictTransferSizeBytes()

```
uint64_t osse::collaborate::Channel::PredictTransferSizeBytes ( ) const
```

Calculates Expected transfer size (bytes)

Returns

Expected transfer size (bytes)

2.8.3.10 rx_gain_db()

```
const double& osse::collaborate::Channel::rx_gain_db ( ) const  [inline]
```

Get received gain (decibels)

Returns

rx_gain_db_ Received gain (decibels)

2.8.3.11 rx_node()

```
Node* osse::collaborate::Channel::rx_node ( ) const  [inline]
```

Get receiver.

Returns

rx_node_ Receiver

2.8.3.12 success_flag()

```
const bool& osse::collaborate::Channel::success_flag ( ) const  [inline]
```

Get success status.

Returns

success_flag_ Success status

2.8.3.13 tx_gain_db()

```
const double& osse::collaborate::Channel::tx_gain_db ( ) const  [inline]
```

Get transmitted gain (decibels)

Returns

tx_gain_db_ Transmitted gain (decibels)

2.8.3.14 tx_node()

```
Node* osse::collaborate::Channel::tx_node ( ) const  [inline]
```

Get transmitter.

Returns

tx_node_ Transmitter

2.8.3.15 Update()

```
void osse::collaborate::Channel::Update (
            const SimulationClock & _clock )
```

Updates member variables and performs fake transfers.

Parameters

in	*_clock*	SimulationClock

2.8.3.16 UpdateDelayS()

```
void osse::collaborate::Channel::UpdateDelayS ( )  [private]
```

Updates delay (seconds)

$$\delta = d/c$$

2.8.3.17 UpdateDistanceM()

`void osse::collaborate::Channel::UpdateDistanceM ( ) [private]`

Updates distance (meters)

$$d = |\vec{p}_{tx} - \vec{p}_{rx}| \quad (meters)$$

2.8.3.18 UpdateGainDb()

`void osse::collaborate::Channel::UpdateGainDb ( ) [private]`

Updates directional gain values (decibels)

$$(\ for \ both \ TX \ and \ RX \ nodes \ )$$
$$\mathbf{O}, \mathbf{B}, \mathbf{A} = Orbit, Body, Antenna \ Frame \ Attitude \ Matrices$$
$$\hat{u} = LOS \ Unit \ Vector$$
$$\hat{a} = Antenna \ LOS \ Unit \ Vector$$
$$\hat{a} = \left(\left(\left(\hat{u}\mathbf{O^{-1}}\right)\mathbf{B^{-1}}\right)\mathbf{A^{-1}}\right)$$
$$g = G_a(\theta_a, \phi_a, \omega_a) \quad (decibels)$$

2.8.3.19 UpdateLosSpeedMPerS()

`void osse::collaborate::Channel::UpdateLosSpeedMPerS ( ) [private]`

Updates line-of-sight speed (meters per second)

$$s = (\vec{v}_t - \vec{v}_r) \cdot \hat{p}_{tx,rx} \quad (meters \ per \ second)$$

2.8.3.20 UpdateLosUnit()

`void osse::collaborate::Channel::UpdateLosUnit ( ) [private]`

Updates line-of-sight unit vectors.

$$\vec{r}_{tx,rx} = |\vec{p}_{rx} - \vec{p}_{tx}| \ , \ \vec{r}_{rx,tx} = |\vec{p}_{tx} - \vec{p}_{rx}|$$
$$\hat{r}_{tx,rx} = \frac{\vec{r}_{tx,rx}}{|\vec{r}_{tx,rx}|}, \ \hat{r}_{rx,tx} = \frac{\vec{r}_{rx,tx}}{|\vec{r}_{rx,tx}|}$$

2.8.3.21 UpdateOmegaRadPerS()

`void osse::collaborate::Channel::UpdateOmegaRadPerS ( ) [private]`

Calculates frequency (radians per second)

Returns

Frequency (radians per second)

$$\omega = \max(\omega_{tx}, \omega_{rx}) \quad (radians\ per\ second)$$

2.8.3.22 UpdateOpen()

`void osse::collaborate::Channel::UpdateOpen ( ) [private]`

Updates potential contact status.

$$\delta = Gain\ Threshold$$
$$Open = (g_{tx} > \delta) \wedge (g_{rx} > \delta)$$

2.8.3.23 UpdatePowerW()

`void osse::collaborate::Channel::UpdatePowerW ( ) [private]`

Updates power (Watts)

$$P_{rx} = P_{tx} * g_{tx} * g_{rx} * \frac{\frac{c}{\frac{\omega}{2\pi}}}{4\pi d}^2$$

The documentation for this class was generated from the following files:

- libs/collaborate/include/collaborate/channel.h
- libs/collaborate/src/channel.cpp

2.9 osse::collaborate::DataLogger Class Reference

An interface to the netcdf library.

`#include <data_logger.h>`

Public Member Functions

- DataLogger (const std::string &_path)

  Constructor.
- void Simulation (const uint16_t &_num_nodes, const uint64_t &_ticks)

  Sets up the netcdf file for logging node data.
- void Measurement (const uint64_t &_ticks)

  Sets up the netcdf file for logging measurement data.
- void Channel (const uint64_t &_ticks)

  Sets up the netcdf file for logging communication parameters.
- template<class T >
  void LogParameter (const int &_node_index, const std::string &_variable, const T *_values, const uint64_t &_index, const uint64_t &_count)

  Logs a buffer of node data.
- template<class T >
  void LogSeries (const std::string &_variable, const T *_values, const uint64_t &_count)

  Logs a buffer of time series data.
- void UnweightedNetwork (const uint16_t &_num_nodes, const uint64_t &_ticks)

  Setup for unweighted network log.
- void WeightedNetwork (const uint16_t &_num_nodes, const uint64_t &_ticks)

  Setup for weighted network log.
- void LogDateTime (const std::string &_variable, const int *_values, const uint64_t &_index, const uint64_t &_count)

  Logs date and time data.
- void LogAntenna (const uint64_t &_theta_ticks, const uint64_t &_phi_ticks, const double *_gain_array)

  Logs antenna gain pattern.
- void LogUnweightedGraph (const uint64_t &_tick, const bool *_edges, const uint16_t _num↩ _nodes)

  Logs unweighted graph.
- void LogWeightedGraph (const uint64_t &_tick, const double *_edges, const uint16_t _num↩ _nodes)

  Logs weighted graph.
- void LogMeasurement ()

  Logs measurement.
- void LogCommunication ()

  Logs communication.

Private Attributes

- netCDF::NcFile ncfile_

  NetCDF file.
- std::vector< netCDF::NcGroup > groups_

  List of NetCDF groups.

2.9.1 Detailed Description

An interface to the netcdf library.

2.9.2 Constructor & Destructor Documentation

2.9.2.1 DataLogger()

```
osse::collaborate::DataLogger::DataLogger (
            const std::string & _path )  [explicit]
```

Constructor.

Parameters

in	*_path*	File sink path

2.9.3 Member Function Documentation

2.9.3.1 Channel()

```
void osse::collaborate::DataLogger::Channel (
            const uint64_t & _ticks )
```

Sets up the netcdf file for logging communication parameters.

Parameters

in	*_ticks*	Number of simulation ticks

2.9.3.2 LogAntenna()

```
void osse::collaborate::DataLogger::LogAntenna (
            const uint64_t & _theta_ticks,
            const uint64_t & _phi_ticks,
            const double * _gain_array )
```

Logs antenna gain pattern.

Parameters

in	*_theta_ticks*	Number of theta values
in	*_phi_ticks*	Number of phi values
in	*_gain_array*	Array of gain values

2.9.3.3 LogDateTime()

```
void osse::collaborate::DataLogger::LogDateTime (
            const std::string & _variable,
            const int * _values,
            const uint64_t & _index,
            const uint64_t & _count )
```

Logs date and time data.

Parameters

in	*_variable*	NetCDF variable name
in	*_values*	Integer values
in	*_index*	Index in the NetCDF variable
in	*_count*	Count of elements to transfer

2.9.3.4 LogParameter()

```
template<class T >
void osse::collaborate::DataLogger::LogParameter (
            const int & _node_index,
            const std::string & _variable,
            const T * _values,
            const uint64_t & _index,
            const uint64_t & _count )  [inline]
```

Logs a buffer of node data.

Parameters

in	*_node_index*	Index of the source node
in	*_variable*	NetCDF variable name
in	*_values*	Array of values
in	*_index*	Index in NetCDF variable
in	*_count*	Count of elements to transfer

2.9.3.5 LogSeries()

```
template<class T >
void osse::collaborate::DataLogger::LogSeries (
            const std::string & _variable,
            const T * _values,
            const uint64_t & _count )  [inline]
```

Logs a buffer of time series data.

Parameters

in	*_variable*	NetCDF variable name
in	*_values*	Array of values
in	*_count*	Count of elements to transfer

2.9.3.6 LogUnweightedGraph()

```
void osse::collaborate::DataLogger::LogUnweightedGraph (
            const uint64_t & _tick,
            const bool * _edges,
            const uint16_t _num_nodes )
```

Logs unweighted graph.

Parameters

in	*_tick*	Tick of the simulation clock
in	*_edges*	Edge flags
in	*_num_nodes*	Number of nodes in network

2.9.3.7 LogWeightedGraph()

```
void osse::collaborate::DataLogger::LogWeightedGraph (
            const uint64_t & _tick,
            const double * _edges,
            const uint16_t _num_nodes )
```

Logs weighted graph.

Parameters

in	*_tick*	Tick of the simulation clock
in	*_edges*	Edge weightes
in	*_num_nodes*	Number of nodes in network

2.9.3.8 Measurement()

```
void osse::collaborate::DataLogger::Measurement (
            const uint64_t & _ticks )
```

Sets up the netcdf file for logging measurement data.

Parameters

in	*_ticks*	Number of simulation ticks

2.9.3.9 Simulation()

```
void osse::collaborate::DataLogger::Simulation (
            const uint16_t & _num_nodes,
            const uint64_t & _ticks )
```

Sets up the netcdf file for logging node data.

Parameters

in	*_num_nodes*	Number of nodes
in	*_ticks*	Number of simulation ticks

2.9.3.10 UnweightedNetwork()

```
void osse::collaborate::DataLogger::UnweightedNetwork (
            const uint16_t & _num_nodes,
            const uint64_t & _ticks )
```

Setup for unweighted network log.

Parameters

in	*_num_nodes*	Number of nodes in network
in	*_ticks*	Number of ticks in simulation

2.9.3.11 WeightedNetwork()

```
void osse::collaborate::DataLogger::WeightedNetwork (
            const uint16_t & _num_nodes,
            const uint64_t & _ticks )
```

Setup for weighted network log.

Parameters

in	*_num_nodes*	Number of nodes in network
in	*_ticks*	Number of ticks in simulation

The documentation for this class was generated from the following files:

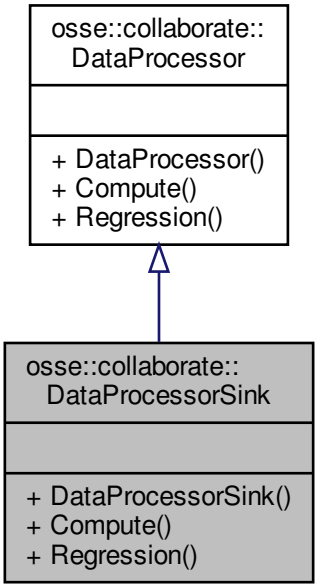
- libs/collaborate/include/collaborate/data_logger.h
- libs/collaborate/src/data_logger.cpp

2.10 osse::collaborate::DataProcessorSink Class Reference

A sink satellite's internal data processor.

`#include <data_processor_sink.h>`

Inheritance diagram for osse::collaborate::DataProcessorSink:



Public Member Functions

- DataProcessorSink ()

  Constructor.
- void Compute (const std::vector< PacketRaw > &_raw_packets, const uint16_t &_source_index, const SimulationClock &_clock, std::vector< Geodetic > *_min_list, std::vector< Geodetic > *_max_list, std::vector< std::pair< bool, uint16_t >> *_feedback) const

  Processes a sensor measurement.
- void Regression (const bool &_success, const uint16_t &_constellation)

  Adapts thresholds in response to feedback.

2.10.1 Detailed Description

A sink satellite's internal data processor.

2.10.2 Member Function Documentation

2.10.2.1 Compute()

```
void osse::collaborate::DataProcessorSink::Compute (
            const std::vector< PacketRaw > & _raw_packets,
            const uint16_t & _source_index,
            const SimulationClock & _clock,
            std::vector< Geodetic > * _min_list,
            std::vector< Geodetic > * _max_list,
            std::vector< std::pair< bool, uint16_t >> * _feedback ) const  [virtual]
```

Processes a sensor measurement.

Parameters

in	*_raw_packets*	List of raw measurement packets
in	*_source_index*	Index of source node
in	*_clock*	Simulation clock
in	*_min_list*	List of minimal suggestions
in	*_max_list*	List of maximum suggestions
in	*_feedback*	List of feedback target node indices

Implements osse::collaborate::DataProcessor.

2.10.2.2 Regression()

```
void osse::collaborate::DataProcessorSink::Regression (
            const bool & _success,
            const uint16_t & _constellation )  [virtual]
```

Adapts thresholds in response to feedback.

Parameters

in	*_success*	Whether the measurement exceeded the threshold
in	*_constellation*	Constellation of the target satellite

Implements osse::collaborate::DataProcessor.

The documentation for this class was generated from the following files:

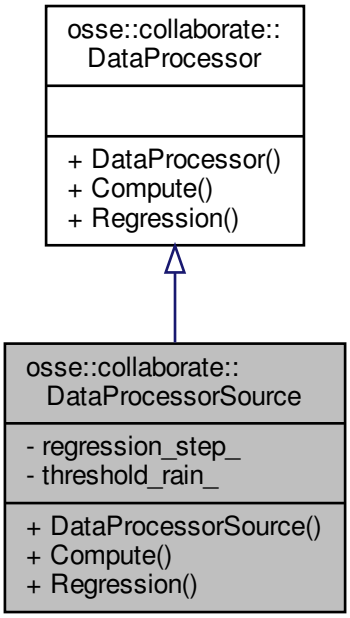
- libs/collaborate/include/collaborate/data_processor_sink.h
- libs/collaborate/src/data_processor_sink.cpp

2.11 osse::collaborate::DataProcessorSource Class Reference

A source/informer satellite's internal data processor.

`#include <data_processor_source.h>`

Inheritance diagram for osse::collaborate::DataProcessorSource:



Public Member Functions

- DataProcessorSource ()

  Constructor.
- void Compute (const std::vector< PacketRaw > &_raw_packets, const uint16_t &_source_index, const SimulationClock &_clock, std::vector< Geodetic > *_min_list, std::vector< Geodetic > *_max_list, std::vector< std::pair< bool, uint16_t >> *_feedback) const

  Processes a sensor measurement.
- void Regression (const bool &_success, const uint16_t &_constellation)

  Adapts thresholds in response to feedback.

Private Attributes

- double regression_step_

  Size of correction in regression action.
- double threshold_rain_

  Cloud depth minimum for existence of rain.

2.11.1 Detailed Description

A source/informer satellite's internal data processor.

2.11.2 Member Function Documentation

2.11.2.1 Compute()

```
void osse::collaborate::DataProcessorSource::Compute (
            const std::vector< PacketRaw > & _raw_packets,
            const uint16_t & _source_index,
            const SimulationClock & _clock,
            std::vector< Geodetic > * _min_list,
            std::vector< Geodetic > * _max_list,
            std::vector< std::pair< bool, uint16_t >> * _feedback ) const  [virtual]
```

Processes a sensor measurement.

Parameters

in	*_raw_packets*	List of raw measurement packets
in	*_source_index*	Index of source node
in	*_clock*	Simulation clock
in	*_min_list*	List of minimal suggestions
in	*_max_list*	List of maximum suggestions
in	*_feedback*	List of feedback target node indices

Implements osse::collaborate::DataProcessor.

2.11.2.2 Regression()

```
void osse::collaborate::DataProcessorSource::Regression (
            const bool & _success,
            const uint16_t & _constellation )  [virtual]
```

Adapts thresholds in response to feedback.

Parameters

in	*_success*	Whether the measurement exceeded the threshold
in	*_constellation*	Constellation of the target satellite

Implements osse::collaborate::DataProcessor.

The documentation for this class was generated from the following files:

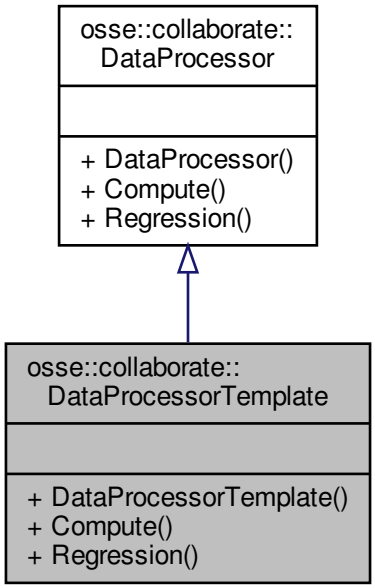
- libs/collaborate/include/collaborate/data_processor_source.h
- libs/collaborate/src/data_processor_source.cpp

2.12 osse::collaborate::DataProcessorTemplate Class Reference

A template satellite's internal data processor.

`#include <data_processor_template.h>`

Inheritance diagram for osse::collaborate::DataProcessorTemplate:



Public Member Functions

- DataProcessorTemplate ()

  Constructor.
- void Compute (const std::vector< PacketRaw > &_raw_packets, const uint16_t &_source_index, const SimulationClock &_simulation_clock, std::vector< Geodetic > *_min_list, std::vector< Geodetic > *_max_list, std::vector< std::pair< bool, uint16_t >> *_feedback) const

  Processes a sensor measurement.
- void Regression (const bool &_success, const uint16_t &_constellation)

  Adapts thresholds in response to feedback.

2.12.1 Detailed Description

A template satellite's internal data processor.

2.12.2 Member Function Documentation

2.12.2.1 Compute()

```
void osse::collaborate::DataProcessorTemplate::Compute (
            const std::vector< PacketRaw > & _raw_packets,
            const uint16_t & _source_index,
            const SimulationClock & _simulation_clock,
            std::vector< Geodetic > * _min_list,
            std::vector< Geodetic > * _max_list,
            std::vector< std::pair< bool, uint16_t >> * _feedback ) const  [virtual]
```

Processes a sensor measurement.

Parameters

in	*_raw_packets*	List of raw measurement packets
in	*_source_index*	Index of source node
in	*_simulation_clock*	Simulation simulation_clock
in	*_min_list*	List of minimal suggestions
in	*_max_list*	List of minimal suggestions
in	*_feedback*	List of feedback target node indices

Implements osse::collaborate::DataProcessor.

2.12.2.2 Regression()

```
void osse::collaborate::DataProcessorTemplate::Regression (
            const bool & _success,
            const uint16_t & _constellation )  [virtual]
```

Adapts thresholds in response to feedback.

Parameters

in	*_success*	Whether the measurement exceeded the threshold
in	*_constellation*	Constellation of the target satellite

Implements osse::collaborate::DataProcessor.

The documentation for this class was generated from the following files:

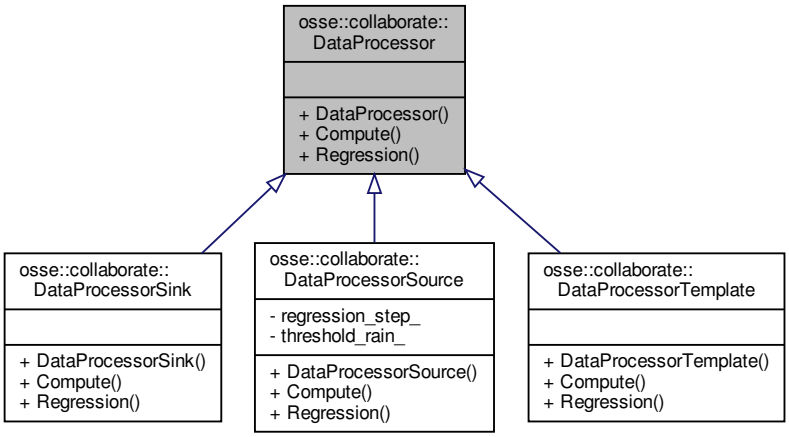
- libs/collaborate/include/collaborate/data_processor_template.h
- libs/collaborate/src/data_processor_template.cpp

2.13 osse::collaborate::DataProcessor Class Reference

A satellite's internal data processor.

`#include <data_processor.h>`

Inheritance diagram for osse::collaborate::DataProcessor:



Public Member Functions

- DataProcessor ()

  Constructor.
- virtual void Compute (const std::vector< PacketRaw > &_raw_packets, const uint16_t &↩ _source_index, const SimulationClock &_clock, std::vector< Geodetic > *_min_list, std↩ ::vector< Geodetic > *_max_list, std::vector< std::pair< bool, uint16_t >> *_feedback) const =0

  Processes a sensor measurement.
- virtual void Regression (const bool &_success, const uint16_t &_constellation)=0

  Adapts thresholds in response to feedback.

2.13.1 Detailed Description

A satellite's internal data processor.

2.13.2 Member Function Documentation

2.13.2.1 Compute()

```
virtual void osse::collaborate::DataProcessor::Compute (
            const std::vector< PacketRaw > & _raw_packets,
            const uint16_t & _source_index,
            const SimulationClock & _clock,
            std::vector< Geodetic > * _min_list,
            std::vector< Geodetic > * _max_list,
            std::vector< std::pair< bool, uint16_t >> * _feedback ) const  [pure virtual]
```

Processes a sensor measurement.

Parameters

in	*_raw_packets*	List of raw measurement packets
in	*_source_index*	Index of source node
in	*_clock*	Simulation clock
in	*_min_list*	List of minimal suggestions
in	*_max_list*	List of maximum suggestions
in	*_feedback*	List of feedback target node indices

Implemented in osse::collaborate::DataProcessorSink, osse::collaborate::DataProcessorSource, and osse::collaborate::DataProcessorTemplate.

2.13.2.2 Regression()

```
virtual void osse::collaborate::DataProcessor::Regression (
            const bool & _success,
            const uint16_t & _constellation )  [pure virtual]
```

Adapts thresholds in response to feedback.

Parameters

in	*_success*	Whether the measurement exceeded the threshold
in	*_constellation*	Constellation of the target satellite

Implemented in osse::collaborate::DataProcessorSink, osse::collaborate::DataProcessorSource, and osse::collaborate::DataProcessorTemplate.

The documentation for this class was generated from the following files:

- libs/collaborate/include/collaborate/data_processor.h
- libs/collaborate/src/data_processor.cpp

2.14 osse::collaborate::EarthData Class Reference

A map of scientific measurement data.

`#include <earth_data.h>`

Public Member Functions

- EarthData (const std::string &_root)

  Constructor.
- void Update (const SimulationClock &_clock, const std::string &_variable)

  Updates the truth data with a new frame of measurement data.
- float Measure (const double &_latitude_rad, const double &_longitude_rad) const

  Obtains a data sample at the nearest location on the discrete map.

Private Member Functions

- void Buffer (const std::string &_variable)

  Loads netcdf data into the current data frame.
- uint16_t IndexLatitude (const double &_latitude_rad) const

  Obtains the index of the nearest latitude value.
- uint16_t IndexLongitude (const double &_longitude_rad) const

  Obtains the index of the nearest longitude value.
- std::vector< std::string > FindDataPaths (const std::string &_root) const

  Finds all regular paths in a directory.

Private Attributes

- const std::vector< std::string > data_paths_

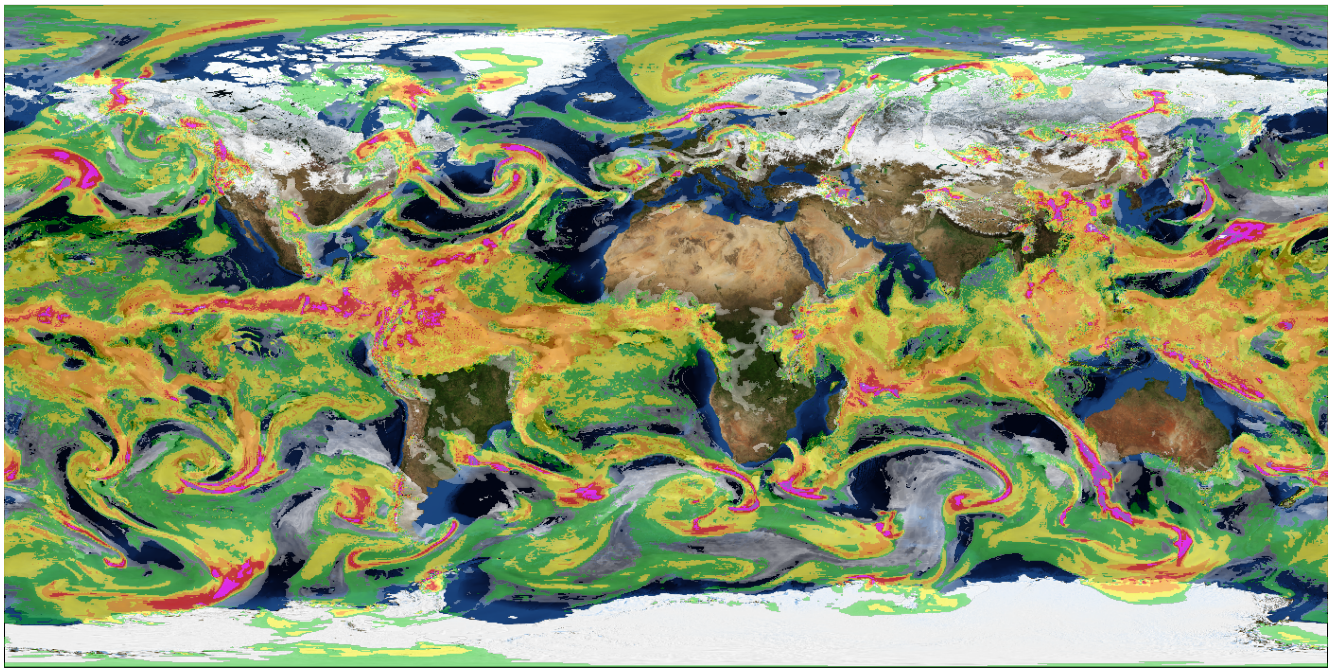
  The list of netcdf files.
- uint64_t current_index_

  Current data-set's index.
- std::array< float, earth::kNumPositions > kData_

  Current data set.

2.14.1 Detailed Description

A map of scientific measurement data.



2.14.2 Constructor & Destructor Documentation

2.14.2.1 EarthData()

```
osse::collaborate::EarthData::EarthData (
            const std::string & _root )  [explicit]
```

Constructor.

Parameters

in	*_root*	Root directory path

2.14.3 Member Function Documentation

2.14.3.1 Buffer()

```
void osse::collaborate::EarthData::Buffer (
            const std::string & _variable )  [private]
```

Loads netcdf data into the current data frame.

Parameters

in	*_variable*	Variable in nercdf file

2.14.3.2 FindDataPaths()

```
std::vector< std::string > osse::collaborate::EarthData::FindDataPaths (
            const std::string & _root ) const  [private]
```

Finds all regular paths in a directory.

Parameters

in	*_root*	Root directory path name

Returns

List of all regular paths in a directory

2.14.3.3 IndexLatitude()

```
uint16_t osse::collaborate::EarthData::IndexLatitude (
            const double & _latitude_rad ) const  [private]
```

Obtains the index of the nearest latitude value.

Parameters

in	*_latitude_rad*	Latitude (radians)

Returns

Index of the nearest latitude value

2.14.3.4 IndexLongitude()

```
uint16_t osse::collaborate::EarthData::IndexLongitude (
            const double & _longitude_rad ) const  [private]
```

Obtains the index of the nearest longitude value.

Parameters

in	*_longitude_rad*	Longitude (radians)

Returns

Index of the nearest longitude value

2.14.3.5 Measure()

```
float osse::collaborate::EarthData::Measure (
            const double & _latitude_rad,
            const double & _longitude_rad ) const
```

Obtains a data sample at the nearest location on the discrete map.

Parameters

in	*_latitude_rad*	Latitude (radians)
in	*_longitude_rad*	Longitude (radians)

Returns

The data sample

2.14.3.6 Update()

```
void osse::collaborate::EarthData::Update (
            const SimulationClock & _clock,
            const std::string & _variable )
```

Updates the truth data with a new frame of measurement data.

Parameters

in	*_clock*	Simulation clock
in	*_variable*	Variable in nercdf file

The documentation for this class was generated from the following files:

- libs/collaborate/include/collaborate/earth_data.h
- libs/collaborate/src/earth_data.cpp

2.15 osse::collaborate::EventLogger Class Reference

An interface to the spdlog logger.

```
#include <event_logger.h>
```

Public Member Functions

- EventLogger (const std::string &_path)

  Constructor.
- void Initialize (const std::string &_level, const std::string &_console_level, const bool &_utc)

  Initialize the event logger's behavor.
- spdlog::logger * log ()

  Get the spdlog.

Private Attributes

- std::shared_ptr< spdlog::sinks::stdout_color_sink_mt > console_

  Console sink.
- std::shared_ptr< spdlog::sinks::basic_file_sink_mt > file_

  File sink.
- std::unique_ptr< spdlog::logger > log_

  Multi-sink.

2.15.1 Detailed Description

An interface to the spdlog logger.

2.15.2 Constructor & Destructor Documentation

2.15.2.1 EventLogger()

```
osse::collaborate::EventLogger::EventLogger (
            const std::string & _path )  [explicit]
```

Constructor.

Parameters

in	*_path*	File sink path

2.15.3 Member Function Documentation

2.15.3.1 Initialize()

```
void osse::collaborate::EventLogger::Initialize (
            const std::string & _level,
            const std::string & _console_level,
            const bool & _utc )
```

Initialize the event logger's behavor.

Parameters

in	*_level*	Base level for logging
in	*_console_level*	Level for console logging
in	*_utc*	Whether to use UTC or local time

2.15.3.2 log()

`spdlog::logger* osse::collaborate::EventLogger::log ( ) [inline]`

Get the spdlog.

Returns

log_ Pointer to spdlog

The documentation for this class was generated from the following files:

- libs/collaborate/include/collaborate/event_logger.h
- libs/collaborate/src/event_logger.cpp

2.16 osse::collaborate::Geodetic Class Reference

Latitude longitude, and altitude.

`#include <geodetic.h>`

Public Member Functions

- Geodetic ()

  Constructor.
- Geodetic (const double &_latitude_rad, const double &_longitude_rad, const double &_altitude_m)

  Constructor from LLH.
- Geodetic (const Vector &_position_m_rad, const SimulationClock &_clock, const uint64_t &_offset_s)

  Constructor from position and current time.
- Geodetic (const std::array< double, 3 > &_triple)

  Constructor from LLH.

- Geodetic (const Vector &_position_m_rad, const Vector &_direction, const SimulationClock &_clock, const uint64_t &_offset_s)

  Constructor from ray intersection with the Earth.
- std::array< double, 3 > Intersection (const Vector &_position_m_rad, const Vector &_↩ direction, const SimulationClock &_clock, const uint64_t &_offset_s)

  Calculates ray intersection with Earth's surface.
- double Haversine (const Geodetic &_other) const

  Calculates a great-circle distance to another location.
- Vector ToVector (const SimulationClock &_clock) const

  Converts to a vector.
- std::string ToString () const

  Outputs the geodetic to a string.
- std::vector< double > ObtainLog () const

  Generates a log of coordinates.
- const double & latitude_rad () const

  Get latitude (radians)
- const double & longitude_rad () const

  Get longitude (radians)
- const double & altitude_m () const

  Get altitude (meters)

Private Member Functions

- double CalculateLatitudeRad (const Vector &_position_m_rad, const SimulationClock &_↩ clock, const uint64_t &_offset_s) const

  Calculate latitude from position and current time.
- double CalculateLongitudeRad (const Vector &_position_m_rad, const SimulationClock &↩ _clock, const uint64_t &_offset_s) const

  Calculate longitude from position and current time.
- double CalculateAltitudeM (const Vector &_position_m_rad, const SimulationClock &_clock, const uint64_t &_offset_s) const

  Calculate alttitude from position and current time.

Private Attributes

- double latitude_rad_

  Latitude (radians)
- double longitude_rad_

  Longitude (radians)
- double altitude_m_

  Altitude (meters)

2.16.1 Detailed Description

Latitude longitude, and altitude.

2.16.2 Constructor & Destructor Documentation

2.16.2.1 Geodetic() [1/4]

```
osse::collaborate::Geodetic::Geodetic (
            const double & _latitude_rad,
            const double & _longitude_rad,
            const double & _altitude_m )
```

Constructor from LLH.

Parameters

in	*_latitude_rad*	Latitude (radians)
in	*_longitude_rad*	Longitude (radians)
in	*_altitude_m*	Altitude (meters)

2.16.2.2 Geodetic() [2/4]

```
osse::collaborate::Geodetic::Geodetic (
            const Vector & _position_m_rad,
            const SimulationClock & _clock,
            const uint64_t & _offset_s )
```

Constructor from position and current time.

Parameters

in	*_position_m_rad*	Position of node (meters and radians)
in	*_clock*	Simulation clock
in	*_offset_s*	Offset from current time (seconds)

2.16.2.3 Geodetic() [3/4]

```
osse::collaborate::Geodetic::Geodetic (
            const std::array< double, 3 > & _triple )  [explicit]
```

Constructor from LLH.

Parameters

in	*_triple*	Array containing latitude, longitude, and altitude

2.16.2.4 Geodetic() [4/4]

```
osse::collaborate::Geodetic::Geodetic (
            const Vector & _position_m_rad,
            const Vector & _direction,
            const SimulationClock & _clock,
            const uint64_t & _offset_s )
```

Constructor from ray intersection with the Earth.

Parameters

in	*_position_m_rad*	Position of node
in	*_direction*	Direction of the line of sight
in	*_clock*	Simulation clock
in	*_offset_s*	Offset from current time (seconds)

Returns

The position of closest intersection (or empty if none)

$$\vec{p} = Position$$

$$\hat{r} = Unit\ Focus\ Direction$$

$$a = b = EARTH\ semimajor\ axis,\ c = EARTH\ semiminor\ axis$$

$$\vec{q_1} = \begin{bmatrix} 1 \div a & 1 \div b & 1 \div c \end{bmatrix}$$

$$\vec{q_2} = \begin{bmatrix} a & b & c \end{bmatrix}$$

$$\vec{p_q} = \vec{p}\vec{q_1}$$

$$Quadratic\ Formula$$

$$A = |\vec{r}|^2$$

$$B = 2(\vec{p}_q \cdot \vec{r})$$
$$C = |\vec{p}_q|^2 - 1$$
$$t_1 = \frac{-B + \sqrt{B^2 - 4AC}}{2A}$$
$$t_2 = \frac{-B - \sqrt{B^2 - 4AC}}{2A}$$
$$Intersection = \min\left((\vec{p}_q - ((\vec{p}_q + \hat{r} * t_1) * \vec{q}_2)), (\vec{p}_q - ((\vec{p}_q + \hat{r} * t_2) * \vec{q}_2))\right)$$

2.16.3 Member Function Documentation

2.16.3.1 altitude_m()

```
const double& osse::collaborate::Geodetic::altitude_m ( ) const  [inline]
```

Get altitude (meters)

Returns

altitude_m_ Altitude (meters)

2.16.3.2 CalculateAltitudeM()

```
double osse::collaborate::Geodetic::CalculateAltitudeM (
            const Vector & _position_m_rad,
            const SimulationClock & _clock,
            const uint64_t & _offset_s ) const  [private]
```

Calculate alttitude from position and current time.

Parameters

in	*_position_m_rad*	Position of node (meters and radians)
in	*_clock*	Simulation clock
in	*_offset_s*	Offset from current time (seconds)

Returns

The altitude (meters)

2.16.3.3 CalculateLatitudeRad()

```
double osse::collaborate::Geodetic::CalculateLatitudeRad (
            const Vector & _position_m_rad,
            const SimulationClock & _clock,
            const uint64_t & _offset_s ) const  [private]
```

Calculate latitude from position and current time.

Parameters

in	*_position_m_rad*	Position of node (meters and radians)
in	*_clock*	Simulation clock
in	*_offset_s*	Offset from current time (seconds)

Returns

The latitude (radians)

2.16.3.4 CalculateLongitudeRad()

```
double osse::collaborate::Geodetic::CalculateLongitudeRad (
            const Vector & _position_m_rad,
            const SimulationClock & _clock,
            const uint64_t & _offset_s ) const  [private]
```

Calculate longitude from position and current time.

Parameters

in	*_position_m_rad*	Position of node (meters and radians)
in	*_clock*	Simulation clock
in	*_offset_s*	Offset from current time (seconds)

Returns

The longitude (radians)

2.16.3.5 Haversine()

```
double osse::collaborate::Geodetic::Haversine (
            const Geodetic & _other ) const
```

Calculates a great-circle distance to another location.

Parameters

in	*_other*	Other geodetic location

Returns

distance Great circle distance

$$a = EARTH\ Semimajor\ Axis$$

$$d = 2a \arcsin \sqrt{\left(\sin\left(\frac{lat_1 - lat_2}{2}\right)\right)^2 + \left(\left(\sin\left(\frac{lon_1 - lon_2}{2}\right)\right)^2 \cos(lat_1 * lat_2)\right)}$$

2.16.3.6 Intersection()

```
std::array< double, 3 > osse::collaborate::Geodetic::Intersection (
            const Vector & _position_m_rad,
            const Vector & _direction,
            const SimulationClock & _clock,
            const uint64_t & _offset_s )
```

Calculates ray intersection with Earth's surface.

Parameters

in	*_position_m_rad*	Position of node
in	*_direction*	Direction of the line of sight
in	*_clock*	Simulation clock
in	*_offset_s*	Offset from current time (seconds)

Returns

The position of closest intersection (or empty if none)

$$\vec{p} = Position$$
$$\hat{r} = Unit\ Focus\ Direction$$
$$a = b = EARTH\ semimajor\ axis,\ c = EARTH\ semiminor\ axis$$
$$\vec{q_1} = \begin{bmatrix} 1 \div a & 1 \div b & 1 \div c \end{bmatrix}$$
$$\vec{q_2} = \begin{bmatrix} a & b & c \end{bmatrix}$$
$$\vec{p_q} = \vec{p}\vec{q_1}$$
$$Quadratic\ Formula$$
$$A = |\vec{r}|^2$$
$$B = 2(\vec{p_q} \cdot \vec{r})$$
$$C = |\vec{p_q}|^2 - 1$$
$$t_1 = \frac{-B + \sqrt{B^2 - 4AC}}{2A}$$
$$t_2 = \frac{-B - \sqrt{B^2 - 4AC}}{2A}$$
$$Intersection = \min\left((\vec{p_q} - ((\vec{p_q} + \hat{r} * t_1) * \vec{q_2})), (\vec{p_q} - ((\vec{p_q} + \hat{r} * t_2) * \vec{q_2}))\right)$$

2.16.3.7 latitude_rad()

```
const double& osse::collaborate::Geodetic::latitude_rad ( ) const  [inline]
```

Get latitude (radians)

Returns

latitude_rad_ Latitude (radians)

2.16.3.8 longitude_rad()

```
const double& osse::collaborate::Geodetic::longitude_rad ( ) const  [inline]
```

Get longitude (radians)

Returns

longitude_rad_ Longitude (radians)

2.16.3.9 ObtainLog()

```
std::vector< double > osse::collaborate::Geodetic::ObtainLog ( ) const
```

Generates a log of coordinates.

Returns

A log of coordinates

2.16.3.10 ToString()

```
std::string osse::collaborate::Geodetic::ToString ( ) const
```

Outputs the geodetic to a string.

Returns

Geodetic as a string

2.16.3.11 ToVector()

```
Vector osse::collaborate::Geodetic::ToVector (
            const SimulationClock & _clock ) const
```

Converts to a vector.

Parameters

in	*_clock*	Simulation clock

Returns

The vector

The documentation for this class was generated from the following files:

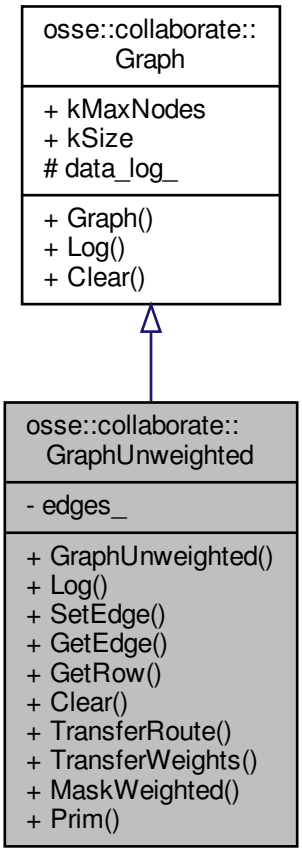
- libs/collaborate/include/collaborate/geodetic.h
- libs/collaborate/src/geodetic.cpp

2.17 osse::collaborate::GraphUnweighted Class Reference

Concrete graph with unweighted edges.

`#include <graph_unweighted.h>`

Inheritance diagram for osse::collaborate::GraphUnweighted:



Public Member Functions

- GraphUnweighted (DataLogger *_data_log)

  Constructor.
- void Log (const uint16_t &_num_nodes, const uint64_t &_tick) const

  Writes unweighted graph to a binary file.
- void SetEdge (const uint16_t &_row, const uint16_t &_col, const bool &_value)

  Sets the edge at a row and a column to.
- bool GetEdge (const uint16_t &_row, const uint16_t &_col) const

Gets the edge at a row and a column to.

- std::set< uint16_t > GetRow (const uint16_t &_row, const uint16_t &_num_nodes) const

  Gets an entire row from the graph.

- void Clear ()

  Sets all edges to false.

- void TransferRoute (const std::vector< uint16_t > &_route)

  The route is copied into the graph.

- void TransferWeights (const GraphWeighted &_weighted_graph, const uint16_t &_num_↩ nodes)

  Edges with weights are true and edges without are false.

- void MaskWeighted (const uint16_t &_num_nodes, GraphWeighted *_weighted_graph)

  Mask the edges of a weighted graph.

- bool Prim (const GraphWeighted &_weighted, const uint16_t &_num_nodes)

  Uses Prim's algorithm to find the minimum spanning tree in a graph.

Private Types

- typedef std::array< bool, Graph::kSize > Edges

  Underlying adjacency attitude_matrix.

Private Attributes

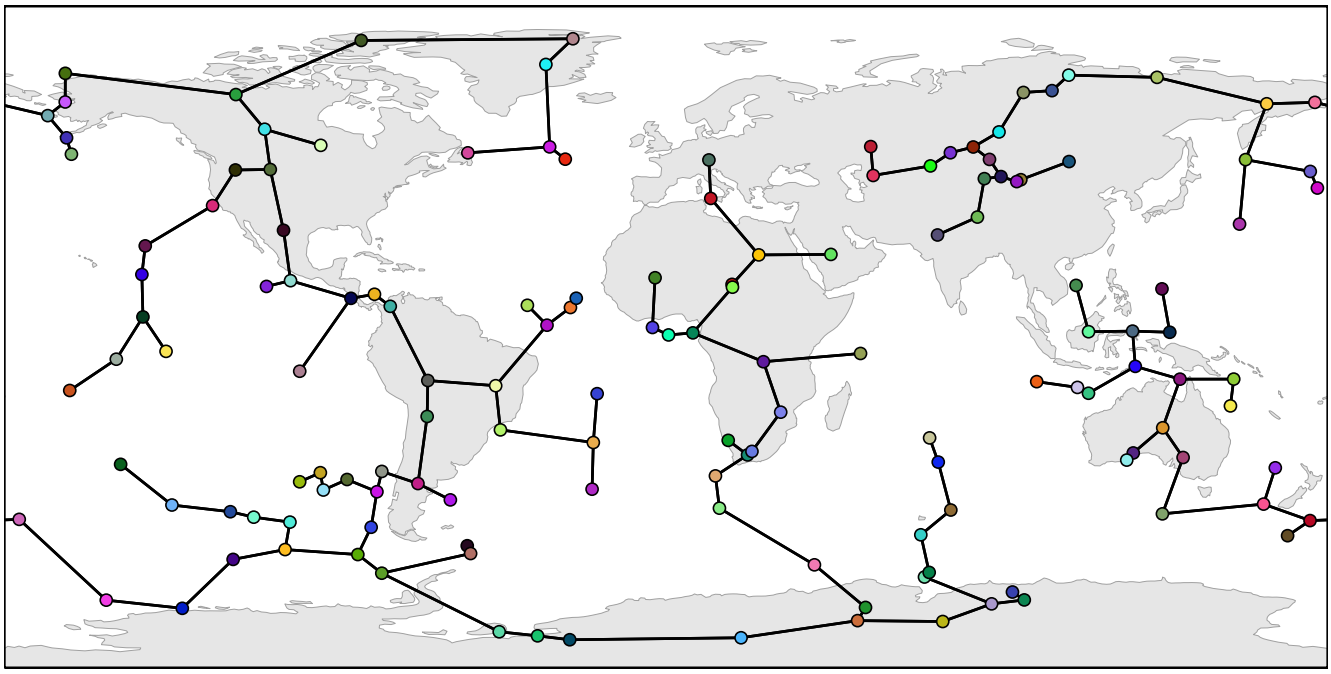
- std::unique_ptr< Edges > edges_

  Unweighted edges.

Additional Inherited Members

2.17.1 Detailed Description

Concrete graph with unweighted edges.



$$n_x n_x \in \{True, False\}$$

$$\mathbf{E} = Unweighted\ Edges = \begin{bmatrix} 0 & n_0n_1 & \dots & n_0n_{N-1} & n_0n_N \\ n_1n_0 & 0 & \dots & n_1n_{N-1} & n_1n_N \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ n_{N-1}n_0 & n_{N-1}n_1 & \dots & 0 & n_{N-1}n_N \\ n_Nn_0 & n_Nn_1 & \dots & n_Nn_{N-1} & 0 \end{bmatrix}$$

2.17.2 Constructor & Destructor Documentation

2.17.2.1 GraphUnweighted()

```
osse::collaborate::GraphUnweighted::GraphUnweighted (
            DataLogger * _data_log ) [explicit]
```

Constructor.

Parameters

in	*_data_log*	Data logger

2.17.3 Member Function Documentation

2.17.3.1 GetEdge()

```
bool osse::collaborate::GraphUnweighted::GetEdge (
            const uint16_t & _row,
            const uint16_t & _col ) const
```

Gets the edge at a row and a column to.

Parameters

in	*_row*	Row
in	*_col*	Column

Returns

Value

2.17.3.2 GetRow()

```
std::set< uint16_t > osse::collaborate::GraphUnweighted::GetRow (
            const uint16_t & _row,
            const uint16_t & _num_nodes ) const
```

Gets an entire row from the graph.

Parameters

in	*_row*	Row number
in	*_num_nodes*	Number of nodes

Returns

Set of transmitters

2.17.3.3 Log()

```
void osse::collaborate::GraphUnweighted::Log (
            const uint16_t & _num_nodes,
            const uint64_t & _tick ) const  [virtual]
```

Writes unweighted graph to a binary file.

Parameters

in	*_num_nodes*	Number of nodes (rows and columns)
in	*_tick*	The simulation clock tick

Implements osse::collaborate::Graph.

2.17.3.4 MaskWeighted()

```
void osse::collaborate::GraphUnweighted::MaskWeighted (
            const uint16_t & _num_nodes,
            GraphWeighted * _weighted_graph )
```

Mask the edges of a weighted graph.

Parameters

in	*_weighted_graph*	Weighted graph
in	*_num_nodes*	Number of nodes

2.17.3.5 Prim()

```
bool osse::collaborate::GraphUnweighted::Prim (
            const GraphWeighted & _weighted,
            const uint16_t & _num_nodes )
```

Uses Prim's algorithm to find the minimum spanning tree in a graph.

Parameters

in	*_weighted*	Weighted graph

Returns

connected_ Whether or not the graph is connected

Parameters

in	*_num_nodes*	Number of nodes (rows and columns)

2.17.3.6 SetEdge()

```
void osse::collaborate::GraphUnweighted::SetEdge (
            const uint16_t & _row,
            const uint16_t & _col,
            const bool & _value )
```

Sets the edge at a row and a column to.

Parameters

in	*_row*	Row
in	*_col*	Column
in	*_value*	Value

2.17.3.7 TransferRoute()

```
void osse::collaborate::GraphUnweighted::TransferRoute (
            const std::vector< uint16_t > & _route )
```

The route is copied into the graph.

Parameters

in	*_route*	Route

2.17.3.8 TransferWeights()

```
void osse::collaborate::GraphUnweighted::TransferWeights (
            const GraphWeighted & _weighted_graph,
            const uint16_t & _num_nodes )
```

Edges with weights are true and edges without are false.

Parameters

in	*_weighted_graph*	Weighted graph
in	*_num_nodes*	Number of nodes

The documentation for this class was generated from the following files:

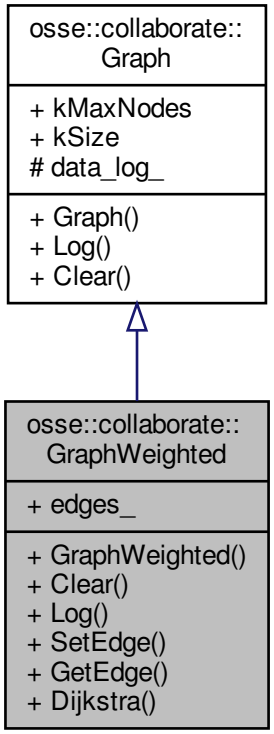
- libs/collaborate/include/collaborate/graph_unweighted.h
- libs/collaborate/src/graph_unweighted.cpp

2.18 osse::collaborate::GraphWeighted Class Reference

Concrete graph with weighted edges.

`#include <graph_weighted.h>`

Inheritance diagram for osse::collaborate::GraphWeighted:



Public Types

- typedef std::array< double, Graph::kSize > Edges

  Underlying adjacency attitude_matrix.

Public Member Functions

- GraphWeighted (DataLogger *_data_log)

  Constructor.
- void Clear ()

  Sets all edges to 0.0.
- void Log (const uint16_t &_num_nodes, const uint64_t &_tick) const

  Writes the weighted graph to a binary file.
- void SetEdge (const uint16_t &_row, const uint16_t &_col, const double &_value)

  Sets the edge at a row and a column to.
- double GetEdge (const uint16_t &_row, const uint16_t &_col) const

  Gets the edge at a row and a column to.
- std::vector< uint16_t > Dijkstra (const uint16_t &_start, const uint16_t &_end)

  Uses Dijkstra's algorithm to find shortest path between two nodes.

Data Fields

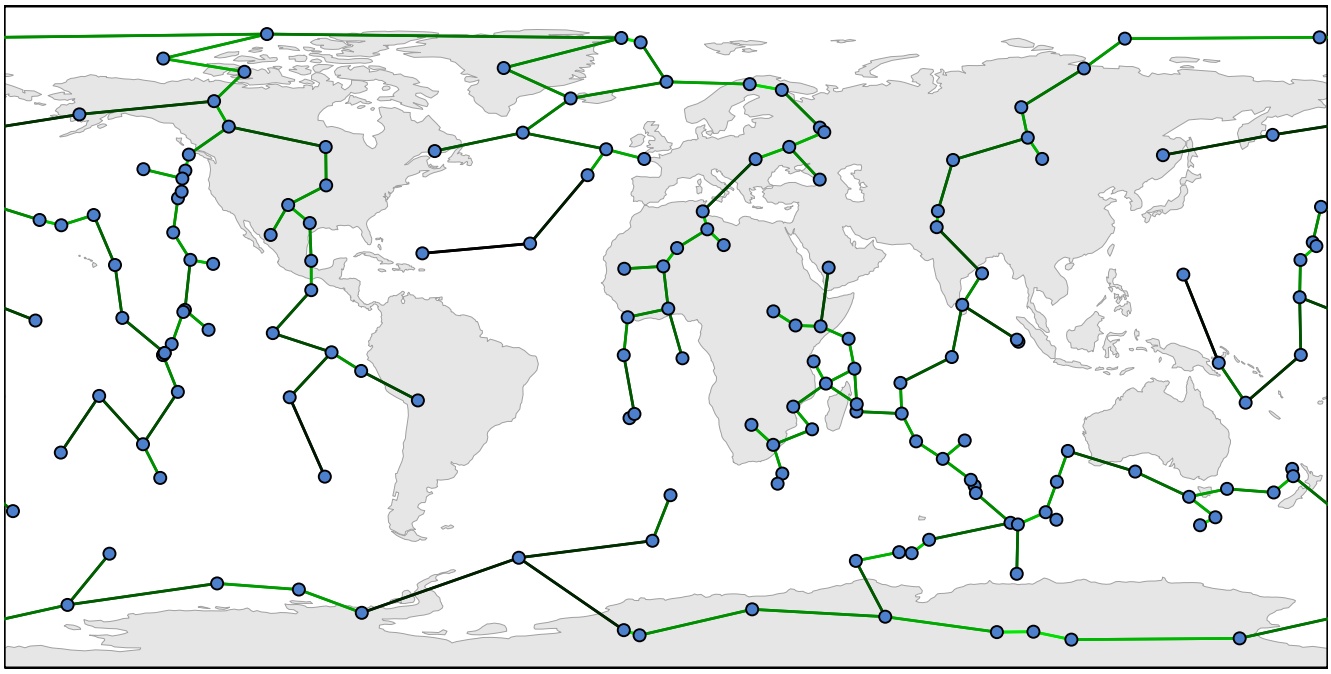
- std::unique_ptr< Edges > edges_

  Weighted edges.

Additional Inherited Members

2.18.1 Detailed Description

Concrete graph with weighted edges.



$$\mathbf{E} = Weighted\ Edges = \begin{matrix} n_x n_x \in \mathbb{R} \\ \begin{bmatrix} 0 & n_0 n_1 & \dots & n_0 n_{N-1} & n_0 n_N \\ n_1 n_0 & 0 & \dots & n_1 n_{N-1} & n_1 n_N \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ n_{N-1} n_0 & n_{N-1} n_1 & \dots & 0 & n_{N-1} n_N \\ n_N n_0 & n_N n_1 & \dots & n_N n_{N-1} & 0 \end{bmatrix} \end{matrix}$$

2.18.2 Constructor & Destructor Documentation

2.18.2.1 GraphWeighted()

```
osse::collaborate::GraphWeighted::GraphWeighted (
            DataLogger * _data_log ) [explicit]
```

Constructor.

Parameters

in	*_data_log*	Data logger

2.18.3 Member Function Documentation

2.18.3.1 Dijkstra()

```
std::vector< uint16_t > osse::collaborate::GraphWeighted::Dijkstra (
            const uint16_t & _start,
            const uint16_t & _end )
```

Uses Dijkstra's algorithm to find shortest path between two nodes.

Parameters

in	*_start*	Beginning index
in	*_end*	End index

Returns

Shortest path between the nodes

2.18.3.2 GetEdge()

```
double osse::collaborate::GraphWeighted::GetEdge (
            const uint16_t & _row,
            const uint16_t & _col ) const
```

Gets the edge at a row and a column to.

Parameters

in	*_row*	Row
in	*_col*	Column

Returns

Value

2.18.3.3 Log()

```
void osse::collaborate::GraphWeighted::Log (
            const uint16_t & _num_nodes,
            const uint64_t & _tick ) const  [virtual]
```

Writes the weighted graph to a binary file.

Parameters

in	*_num_nodes*	Number of nodes (rows and columns)
in	*_tick*	The simulation clock tick

Implements osse::collaborate::Graph.

2.18.3.4 SetEdge()

```
void osse::collaborate::GraphWeighted::SetEdge (
            const uint16_t & _row,
            const uint16_t & _col,
            const double & _value )
```

Sets the edge at a row and a column to.

Parameters

in	*_row*	Row
in	*_col*	Column
in	*_value*	Value

The documentation for this class was generated from the following files:

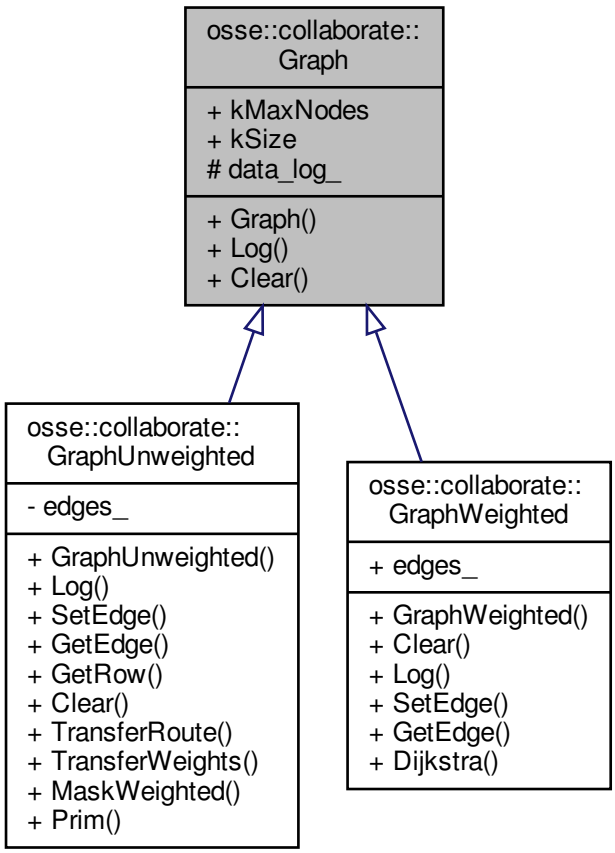
- libs/collaborate/include/collaborate/graph_weighted.h
- libs/collaborate/src/graph_weighted.cpp

2.19 osse::collaborate::Graph Class Reference

Abstract graph.

```
#include <graph.h>
```

Inheritance diagram for osse::collaborate::Graph:



Public Member Functions

- Graph (DataLogger *_data_log)

  Constructor.
- virtual void Log (const uint16_t &_num_nodes, const uint64_t &_tick) const =0

  Writes graph to a binary file.
- virtual void Clear ()=0

  Clears all edges.

Static Public Attributes

- static constexpr uint16_t kMaxNodes = 200

  Maximum number of nodes.
- static constexpr uint16_t kSize = std::pow(kMaxNodes, 2)

  Maximum size of of a graph.

Protected Attributes

- DataLogger * data_log_

  Data logger.

2.19.1 Detailed Description

Abstract graph.

2.19.2 Constructor & Destructor Documentation

2.19.2.1 Graph()

```
osse::collaborate::Graph::Graph (
            DataLogger * _data_log ) [explicit]
```

Constructor.

Parameters

in	*_data_log*	Data logger

2.19.3 Member Function Documentation

2.19.3.1 Log()

```
virtual void osse::collaborate::Graph::Log (
            const uint16_t & _num_nodes,
            const uint64_t & _tick ) const [pure virtual]
```

Writes graph to a binary file.

Parameters

in	*_num_nodes*	Number of nodes (rows and columns)
in	*_tick*	The simulation clock tick

Implemented in osse::collaborate::GraphWeighted, and osse::collaborate::GraphUnweighted.

The documentation for this class was generated from the following files:

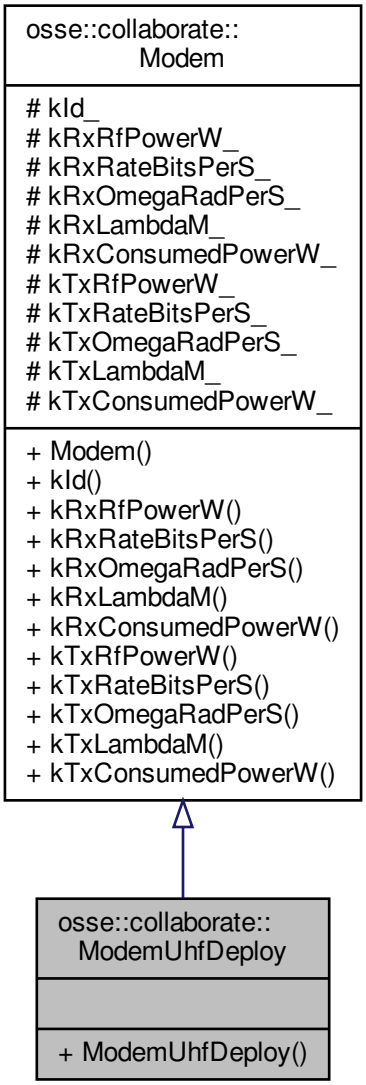
- libs/collaborate/include/collaborate/graph.h
- libs/collaborate/src/graph.cpp

2.20 osse::collaborate::ModemUhfDeploy Class Reference

Concrete UHF modem (deployed on a satellite)

`#include <modem_uhf_deploy.h>`

Inheritance diagram for osse::collaborate::ModemUhfDeploy:



Public Member Functions

- ModemUhfDeploy ()
  Constructor.

Additional Inherited Members

2.20.1 Detailed Description

Concrete UHF modem (deployed on a satellite)

The documentation for this class was generated from the following files:

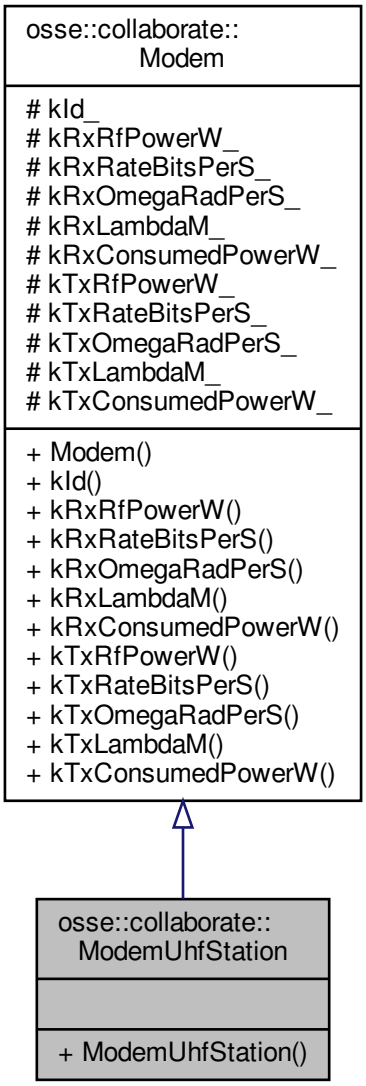
- libs/collaborate/include/collaborate/modem_uhf_deploy.h
- libs/collaborate/src/modem_uhf_deploy.cpp

2.21 osse::collaborate::ModemUhfStation Class Reference

Concrete UHF modem (base station)

`#include <modem_uhf_station.h>`

Inheritance diagram for osse::collaborate::ModemUhfStation:



Public Member Functions

- ModemUhfStation ()

  Constructor.

Additional Inherited Members

2.21.1 Detailed Description

Concrete UHF modem (base station)

The documentation for this class was generated from the following files:

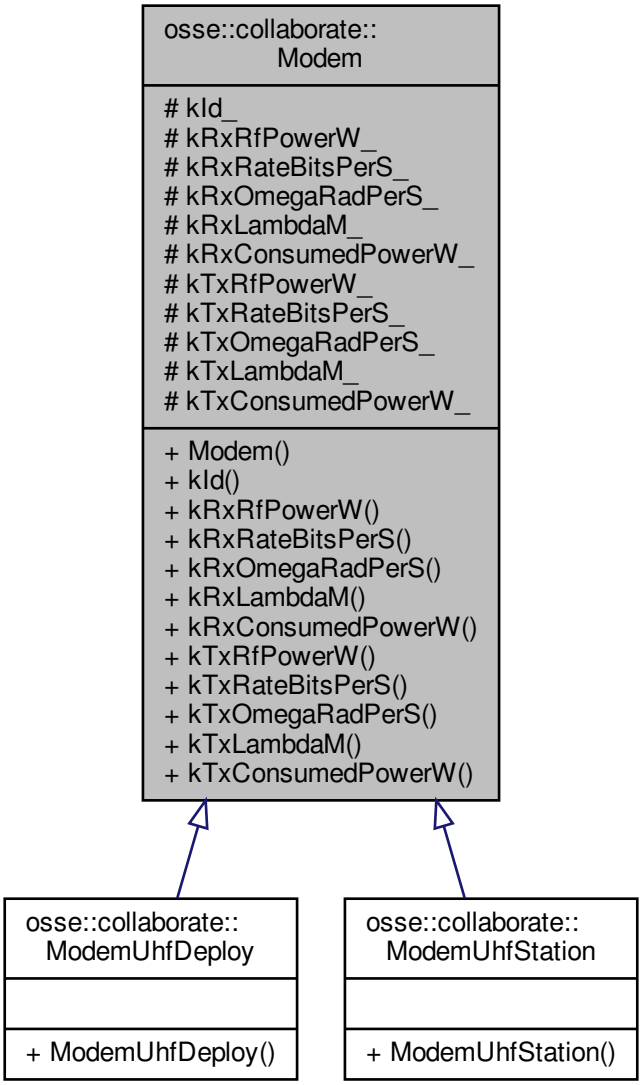
- libs/collaborate/include/collaborate/modem_uhf_station.h
- libs/collaborate/src/modem_uhf_station.cpp

2.22 osse::collaborate::Modem Class Reference

Abstract modem.

`#include <modem.h>`

Inheritance diagram for osse::collaborate::Modem:



Public Member Functions

- Modem (const std::string &_kId, const double &_kRxRfPowerW, const uint64_t &_kRx↩ RateBitsPerS, const double &_kRxOmegaRadPerS, const double &_kRxLambdaM, const double &_kRxConsumedPowerW, const double &_kTxRfPowerW, const uint64_t &_kTx↩ RateBitsPerS, const double &_kTxOmegaRadPerS, const double &_kTxLambdaM, const double &_kTxConsumedPowerW)

  Constructor.
- const std::string & kId ()

  Get ID.
- const double & kRxRfPowerW () const

  Get RX RF power (watts)
- const uint64_t & kRxRateBitsPerS () const

  Get RX data rate (bits per second)
- const double & kRxOmegaRadPerS () const

  Get RX frequency (radians per second)
- const double & kRxLambdaM () const

  Get RX wavelength (meters)
- const double & kRxConsumedPowerW () const

  Get RX consumed power (watts)
- const double & kTxRfPowerW () const

  Get TX RF power (watts)
- const uint64_t & kTxRateBitsPerS () const

  Get TX data rate (bits per second)
- const double & kTxOmegaRadPerS () const

  Get TX frequency (radians per second)
- const double & kTxLambdaM () const

  Get TX wavelength (meters)
- const double & kTxConsumedPowerW () const

  Get TX consumed power (watts)

Protected Attributes

- const std::string kId_

  Identifying string.
- const double kRxRfPowerW_

  RX RF power (watts)
- const uint64_t kRxRateBitsPerS_

  RX data rate (bits per second)
- const double kRxOmegaRadPerS_

  RX frequency (radians per second)

- const double kRxLambdaM_

  RX wavelength (meters)
- const double kRxConsumedPowerW_

  RX consumed power (watts)
- const double kTxRfPowerW_

  TX RF power (watts)
- const uint64_t kTxRateBitsPerS_

  TX data rate (bits per second)
- const double kTxOmegaRadPerS_

  TX frequency (radians per second)
- const double kTxLambdaM_

  TX wavelength (meters)
- const double kTxConsumedPowerW_

  TX consumed power (watts)

2.22.1 Detailed Description

Abstract modem.

2.22.2 Constructor & Destructor Documentation

2.22.2.1 Modem()

```
osse::collaborate::Modem::Modem (
            const std::string & _kId,
            const double & _kRxRfPowerW,
            const uint64_t & _kRxRateBitsPerS,
            const double & _kRxOmegaRadPerS,
            const double & _kRxLambdaM,
            const double & _kRxConsumedPowerW,
            const double & _kTxRfPowerW,
            const uint64_t & _kTxRateBitsPerS,
            const double & _kTxOmegaRadPerS,
            const double & _kTxLambdaM,
            const double & _kTxConsumedPowerW )
```

Constructor.

Parameters

in	*_kId*	String identifier for modem
in	*_kRxRfPowerW*	RX RF power (watts)
in	*_kRxRateBitsPerS*	RX data rate (bits per second)
in	*_kRxOmegaRadPerS*	RX frequency (radians per second)
in	*_kRxLambdaM*	RX wavelength (meters)
in	*_kRxConsumedPowerW*	RX consumed power (watts)
in	*_kTxRfPowerW*	TX RF power (watts)
in	*_kTxRateBitsPerS*	TX data rate (bits per second)
in	*_kTxOmegaRadPerS*	TX frequency (radians per second)
in	*_kTxLambdaM*	TX wavelength (meters)
in	*_kTxConsumedPowerW*	TX consumed power (watts)

2.22.3 Member Function Documentation

2.22.3.1 kId()

```
const std::string& osse::collaborate::Modem::kId ( )  [inline]
```

Get ID.

Returns

kId_ Identifying string

2.22.3.2 kRxConsumedPowerW()

```
const double& osse::collaborate::Modem::kRxConsumedPowerW ( ) const  [inline]
```

Get RX consumed power (watts)

Returns

kRxConsumedPowerW_ RX consumed power (watts)

2.22.3.3 kRxLambdaM()

```
const double& osse::collaborate::Modem::kRxLambdaM ( ) const  [inline]
```

Get RX wavelength (meters)

Returns

kRxLambdaM_ RX wavelength (meters)

2.22.3.4 kRxOmegaRadPerS()

```
const double& osse::collaborate::Modem::kRxOmegaRadPerS ( ) const  [inline]
```

Get RX frequency (radians per second)

Returns

kRxOmegaRadPerS_ RX frequency (radians per second)

2.22.3.5 kRxRateBitsPerS()

```
const uint64_t& osse::collaborate::Modem::kRxRateBitsPerS ( ) const  [inline]
```

Get RX data rate (bits per second)

Returns

kRxRateBitsPerS_ RX data rate (bits per second)

2.22.3.6 kRxRfPowerW()

```
const double& osse::collaborate::Modem::kRxRfPowerW ( ) const  [inline]
```

Get RX RF power (watts)

Returns

kRxRfPowerW_ RX RF power (watts)

2.22.3.7 kTxConsumedPowerW()

```
const double& osse::collaborate::Modem::kTxConsumedPowerW ( ) const  [inline]
```

Get TX consumed power (watts)

Returns

kTxConsumedPowerW_ TX consumed power (watts)

2.22.3.8 kTxLambdaM()

```
const double& osse::collaborate::Modem::kTxLambdaM ( ) const  [inline]
```

Get TX wavelength (meters)

Returns

kTxLambdaM_ TX wavelength (meters)

2.22.3.9 kTxOmegaRadPerS()

```
const double& osse::collaborate::Modem::kTxOmegaRadPerS ( ) const  [inline]
```

Get TX frequency (radians per second)

Returns

kTxOmegaRadPerS_ TX frequency (radians per second)

2.22.3.10 kTxRateBitsPerS()

```
const uint64_t& osse::collaborate::Modem::kTxRateBitsPerS ( ) const  [inline]
```

Get TX data rate (bits per second)

Returns

kTxRateBitsPerS_ TX data rate (bits per second)

2.22.3.11 kTxRfPowerW()

```
const double& osse::collaborate::Modem::kTxRfPowerW ( ) const  [inline]
```

Get TX RF power (watts)

Returns

kTxRfPowerW_ TX RF power (watts)

The documentation for this class was generated from the following files:

- libs/collaborate/include/collaborate/modem.h
- libs/collaborate/src/modem.cpp

2.23 osse::collaborate::Node Class Reference

A member of the network (ground, air, or space)

```
#include <node.h>
```

Data Structures

- struct LogBuffer

  A buffer for logged node data.
- struct PartialLog

  A geodetic log for a node.

Public Types

- enum kMode { **Free**, **Carrying**, **Sensing** }

  Possible modes of operation.
- typedef struct osse::collaborate::Node::PartialLog PartialLog

  A geodetic log for a node.
- typedef struct osse::collaborate::Node::LogBuffer LogBuffer

  A buffer for logged node data.

Public Member Functions

- Node (const std::string &_name, const uint16_t &_index, const uint16_t &_constellation, const Platform *_platform, const SubsystemComm &_comm_if, const SubsystemSensing &_sensing_if, const SubsystemPower &_subsystem_power, const SimulationClock *_clock, DataProcessor *_data_processor, EventLogger *_event_log, DataLogger *_data_log)

  Construct a node.
- void Update (const uint64_t &_offset_s, const bool &_comm_orient, const bool &_sensing_↩ orient, const bool &_measure, const bool &_charge, const bool &_power_update, const bool &_communicate)

  Update the orbital_state and antenna reference_frames.
- void PlanMeasurement (const uint64_t &_start_s, const uint16_t &_return_index)

  Adds a measurement to the list of planned measurements.
- void AddressCommBuffer ()

  Processes Communication buffer to take action.
- void SwitchCommunication (const SubsystemComm::kMode &_mode)

  Switches the mode of the communication interface.
- void MoveSensorDataToCommBuffer ()

  Moves all data from sensing interface to communication interface.
- void BufferDataLog ()

  Buffers entire data frame into the data log.
- void Flush ()

  Buffers remainder data fram into the data log.
- void SetCommBuffer (std::vector< uint8_t > _comm_buffer)

  Sets the data buffer of the communication interface.
- void SetSensingBuffer (std::vector< uint8_t > _sensing_buffer)

  Sets the data buffer of the sensing interface.
- void EraseCommBuffer ()

  Erases the data buffer of the communication interface.
- void EraseSensingBuffer ()

  Erases the data buffer of the sensing interface.
- std::vector< uint8_t > GetCommBuffer ()

  Gets the data buffer of the communication interface.
- std::vector< uint8_t > GetSensingBuffer ()

  Gets the data buffer of the sensing interface.
- void set_mode (const kMode &_mode)

  Set mode of operation.
- std::string name () const

  Get name.
- uint16_t index () const

  Get index.
- uint64_t constellation () const

Get constellation.

- const OrbitalState & orbital_state () const

  Get orbital_state.

- const Platform * kPlatform () const

  Get kPlatform.

- const SubsystemComm & comm_if () const

  Get communication interface.

- const SubsystemSensing & sensing_if () const

  Get sensing interface.

- const kMode & mode () const

  Get mode of operation.

- const SubsystemPower & subsystem_power () const

  Get power subsystem.

- const std::vector< Geodetic > & min_suggestions () const

  Get suggested minimal places to visit.

- const std::vector< Geodetic > & max_suggestions () const

  Get suggested maximal places to visit.

- const std::vector< std::pair< bool, uint16_t > > & feedback () const

  Get feedback.

- void set_min_suggestions (const std::vector< Geodetic > &_min_suggestions)

  Set minimum suggestions.

- void set_max_suggestions (const std::vector< Geodetic > &_max_suggestions)

  Set maximum suggestions.

- void set_feedback (const std::vector< std::pair< bool, uint16_t >> &_feedback)

  Set feedback.

- void set_target_index (const int &_target_index)

  Set target index.

- void set_num_neighbors (const uint16_t &_num_neighbors)

  Set number of neighbors.

- const int & target_index () const

  Get target index.

Static Public Attributes

- static constexpr int kLogBufferSize = 1000

  Number of simulation increments in a single data log.

Private Member Functions

- void UpdateOrbitalState (const uint64_t &_offset_s)

  Update orbital_state.
- void UpdateCommAntenna ()

  Update communication antenna orientation.
- void UpdateSensingAntenna ()

  Update sensing antenna orientation.
- void UpdateMeasurement ()

  Update measurements.
- void UpdatePower (const bool &_charge)

  Update power management.
- void UpdateCommunication ()

  Update commmunication interface.

Private Attributes

- const std::string name_

  Name.
- const uint16_t index_

  Index.
- const uint16_t constellation_

  Constellation.
- const Platform * kPlatform_

  Platform.
- DataProcessor * kDataProcessor_

  Data processor.
- OrbitalState orbital_state_

  OrbitalState.
- SubsystemComm comm_if_

  Communication interface.
- SubsystemSensing sensing_if_

  Sensing interface.
- kMode mode_

  Mode of operation.
- SubsystemPower subsystem_power_

  Power sybsystem.
- std::vector< std::pair< uint64_t, uint16_t > > measurements_

  List of planned measurements.
- std::vector< Geodetic > min_suggestions_

  List of suggested target minimal locations.

- std::vector< Geodetic > max_suggestions_

  List of suggested target maximal locations.
- int target_index_

  Target node index.
- std::vector< std::pair< bool, uint16_t > > feedback_

  List of informer indices.
- uint16_t num_neighbors_

  The number of other visible nodes.
- const SimulationClock * clock_

  Simulation clock.
- EventLogger * event_log_

  Event log.
- LogBuffer log_buffer_

  Buffer for data log.
- DataLogger * data_log_

  Data log.
- uint64_t num_logs_

  Number of completed data buffer logs.

2.23.1 Detailed Description

A member of the network (ground, air, or space)

2.23.2 Constructor & Destructor Documentation

2.23.2.1 Node()

```
osse::collaborate::Node::Node (
            const std::string & _name,
            const uint16_t & _index,
            const uint16_t & _constellation,
            const Platform * _platform,
            const SubsystemComm & _comm_if,
            const SubsystemSensing & _sensing_if,
            const SubsystemPower & _subsystem_power,
            const SimulationClock * _clock,
            DataProcessor * _data_processor,
            EventLogger * _event_log,
            DataLogger * _data_log )
```

Construct a node.

Parameters

in	*_name*	Name
in	*_index*	Index
in	*_constellation*	Which constellation it belongs to
in	*_platform*	Platform
in	*_comm_if*	Communication interface
in	*_sensing_if*	Sensing interface
in	*_subsystem_power*	Power subsystem
in	*_clock*	Simulation clock
in	*_data_processor*	Data processor
in	*_event_log*	Event log
in	*_data_log*	Data log

2.23.3 Member Function Documentation

2.23.3.1 comm_if()

```
const SubsystemComm& osse::collaborate::Node::comm_if ( ) const  [inline]
```

Get communication interface.

Returns

comm_if_ Communication interface

2.23.3.2 constellation()

```
uint64_t osse::collaborate::Node::constellation ( ) const  [inline]
```

Get constellation.

Returns

constellation_ Constellation it belongs to

2.23.3.3 feedback()

```
const std::vector<std::pair<bool, uint16_t> >& osse::collaborate::Node::feedback ( ) const
[inline]
```

Get feedback.

Returns

feedback_ feedback

2.23.3.4 GetCommBuffer()

```
std::vector< uint8_t > osse::collaborate::Node::GetCommBuffer ( )
```

Gets the data buffer of the communication interface.

Returns

Data buffer for the communication interface

2.23.3.5 GetSensingBuffer()

```
std::vector< uint8_t > osse::collaborate::Node::GetSensingBuffer ( )
```

Gets the data buffer of the sensing interface.

Returns

Data buffer for the sensing interface

2.23.3.6 index()

```
uint16_t osse::collaborate::Node::index ( ) const  [inline]
```

Get index.

Returns

index_ Index

2.23.3.7 kPlatform()

```
const Platform* osse::collaborate::Node::kPlatform ( ) const  [inline]
```

Get kPlatform.

Returns

kPlatform_ Platform

2.23.3.8 max_suggestions()

```
const std::vector<Geodetic>& osse::collaborate::Node::max_suggestions ( ) const  [inline]
```

Get suggested maximal places to visit.

Returns

max_suggestions_ Suggested maximal places to visit

2.23.3.9 min_suggestions()

```
const std::vector<Geodetic>& osse::collaborate::Node::min_suggestions ( ) const  [inline]
```

Get suggested minimal places to visit.

Returns

min_suggestions_ Suggested minimal places to visit

2.23.3.10 mode()

```
const kMode& osse::collaborate::Node::mode ( ) const  [inline]
```

Get mode of operation.

Returns

mode_ Mode of operation

2.23.3.11 name()

```
std::string osse::collaborate::Node::name ( ) const  [inline]
```

Get name.

Returns

name_ Name

2.23.3.12 orbital_state()

```
const OrbitalState& osse::collaborate::Node::orbital_state ( ) const  [inline]
```

Get orbital_state.

Returns

orbital_state_ OrbitalState

2.23.3.13 PlanMeasurement()

```
void osse::collaborate::Node::PlanMeasurement (
            const uint64_t & _start_s,
            const uint16_t & _return_index )
```

Adds a measurement to the list of planned measurements.

Parameters

in	*_start_s*	Start time (seconds)
in	*_return_index*	The index of the informer node

2.23.3.14 sensing_if()

```
const SubsystemSensing& osse::collaborate::Node::sensing_if ( ) const  [inline]
```

Get sensing interface.

Returns

sensing_if_ Sensing interface

2.23.3.15 set_feedback()

```
void osse::collaborate::Node::set_feedback (
            const std::vector< std::pair< bool, uint16_t >> & _feedback ) [inline]
```

Set feedback.

Parameters

in	*_feedback*	feedback

2.23.3.16 set_max_suggestions()

```
void osse::collaborate::Node::set_max_suggestions (
            const std::vector< Geodetic > & _max_suggestions ) [inline]
```

Set maximum suggestions.

Parameters

in	*_max_suggestions*	Maximum suggestions

2.23.3.17 set_min_suggestions()

```
void osse::collaborate::Node::set_min_suggestions (
            const std::vector< Geodetic > & _min_suggestions ) [inline]
```

Set minimum suggestions.

Parameters

in	*_min_suggestions*	Minimum suggestions

2.23.3.18 set_mode()

```
void osse::collaborate::Node::set_mode (
            const kMode & _mode )
```

Set mode of operation.

Parameters

in	*_mode*	Mode of operation

2.23.3.19 set_num_neighbors()

```
void osse::collaborate::Node::set_num_neighbors (
            const uint16_t & _num_neighbors )  [inline]
```

Set number of neighbors.

Parameters

in	*_num_neighbors*	number of neighbors

2.23.3.20 set_target_index()

```
void osse::collaborate::Node::set_target_index (
            const int & _target_index )  [inline]
```

Set target index.

Parameters

in	*_target_index*	Target index

2.23.3.21 SetCommBuffer()

```
void osse::collaborate::Node::SetCommBuffer (
            std::vector< uint8_t > _comm_buffer )
```

Sets the data buffer of the communication interface.

Parameters

in	*_comm_buffer*	The data buffer for the communication interface

2.23.3.22 SetSensingBuffer()

```
void osse::collaborate::Node::SetSensingBuffer (
            std::vector< uint8_t > _sensing_buffer )
```

Sets the data buffer of the sensing interface.

Parameters

in	*_sensing_buffer*	The data buffer for the sensing interface

2.23.3.23 subsystem_power()

```
const SubsystemPower& osse::collaborate::Node::subsystem_power ( ) const  [inline]
```

Get power subsystem.

Returns

subsystem_power_ Power subsystem

2.23.3.24 SwitchCommunication()

```
void osse::collaborate::Node::SwitchCommunication (
            const SubsystemComm::kMode & _mode )
```

Switches the mode of the communication interface.

Parameters

in	*_mode*	Interface node

2.23.3.25 target_index()

```
const int& osse::collaborate::Node::target_index ( ) const  [inline]
```

Get target index.

Returns

target_index_ Target index

2.23.3.26 Update()

```
void osse::collaborate::Node::Update (
            const uint64_t & _offset_s,
            const bool & _comm_orient,
            const bool & _sensing_orient,
            const bool & _measure,
            const bool & _charge,
            const bool & _power_update,
            const bool & _communicate )
```

Update the orbital_state and antenna reference_frames.

Parameters

in	*_offset_s*	Offset from the current time (seconds)
in	*_comm_orient*	Whether to orient the comm interface
in	*_sensing_orient*	Whether to orient the sensing interface
in	*_measure*	Whether to update the sensing interface
in	*_charge*	Whether to charge the battery
in	*_power_update*	Whether to update the power subsystem
in	*_communicate*	Whether to consider communcation

2.23.3.27 UpdateOrbitalState()

```
void osse::collaborate::Node::UpdateOrbitalState (
            const uint64_t & _offset_s )  [private]
```

Update orbital_state.

Parameters

in	_↩ *offset*↩ _*s*	Time offset (seconds)

2.23.3.28 UpdatePower()

```
void osse::collaborate::Node::UpdatePower (
            const bool & _charge )  [private]
```

Update power management.

Parameters

in	_*charge*	Whether to charge the battery

The documentation for this class was generated from the following files:

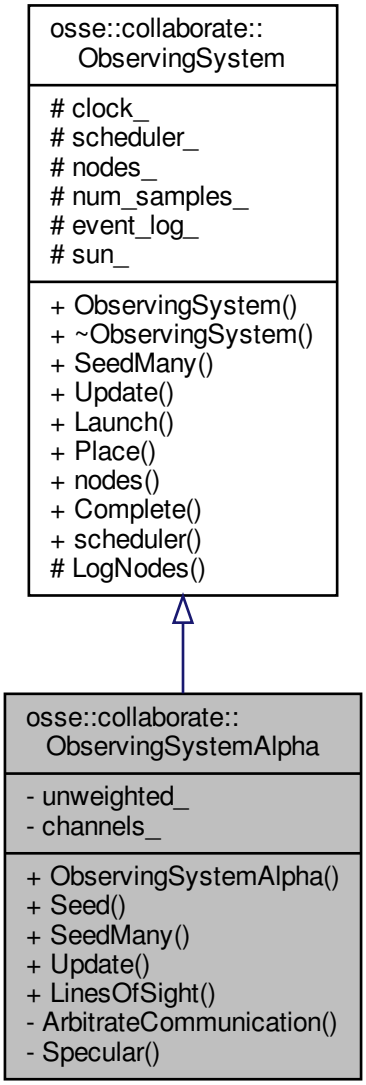
- libs/collaborate/include/collaborate/node.h
- libs/collaborate/src/node.cpp

2.24 osse::collaborate::ObservingSystemAlpha Class Reference

Concrete satellite observing system.

```
#include <observing_system_alpha.h>
```

Inheritance diagram for osse::collaborate::ObservingSystemAlpha:



Public Member Functions

- ObservingSystemAlpha (Sun *_sun, SimulationClock *_clock, Scheduler *_collaborate, EventLogger *_event_log, DataLogger *_network_log)

  Constructor.
- void Seed (const uint64_t &_span_s)

  Generates random list of samples to start with.
- void SeedMany (const uint64_t &_span_s, const uint16_t &_constellation)

  Generates random list of samples to start with for a constellation.

- void Update ()

  Update everything.
- void LinesOfSight ()

  Calculate all lines of sight between satellites and write log.

Private Member Functions

- void ArbitrateCommunication ()

  Creates new channels for nodes needing to communicate.
- void Specular ()

  Finds all specular points.

Private Attributes

- GraphUnweighted unweighted_

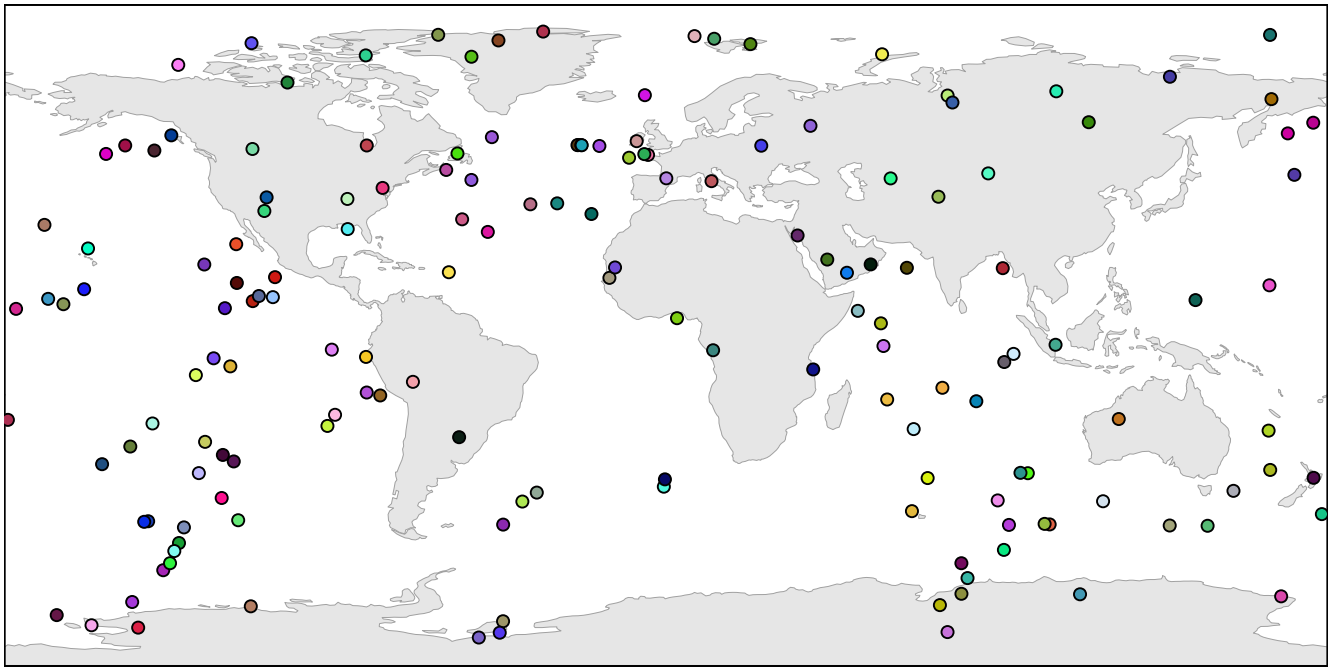
  Active channels.
- std::vector< Channel > channels_

  An adjacency attitude_matrix.

Additional Inherited Members

2.24.1 Detailed Description

Concrete satellite observing system.



2.24.2 Constructor & Destructor Documentation

2.24.2.1 ObservingSystemAlpha()

```
osse::collaborate::ObservingSystemAlpha::ObservingSystemAlpha (
            Sun * _sun,
            SimulationClock * _clock,
            Scheduler * _collaborate,
            EventLogger * _event_log,
            DataLogger * _network_log )
```

Constructor.

Parameters

in	*_sun*	Star at the center of the solar system
in	*_clock*	Simulation clock
in	*_collaborate*	Autonomous network collaborate
in	*_event_log*	Event logger
in	*_network_log*	Network logger

2.24.3 Member Function Documentation

2.24.3.1 Seed()

```
void osse::collaborate::ObservingSystemAlpha::Seed (
            const uint64_t & _span_s )
```

Generates random list of samples to start with.

Parameters

in	*_span_s*	Total time span of the simulation

2.24.3.2 SeedMany()

```
void osse::collaborate::ObservingSystemAlpha::SeedMany (
            const uint64_t & _span_s,
            const uint16_t & _constellation ) [virtual]
```

Generates random list of samples to start with for a constellation.

Parameters

in	*_span_s*	Total time span of the simulation
in	*_constellation*	Constellation

Implements osse::collaborate::ObservingSystem.

The documentation for this class was generated from the following files:

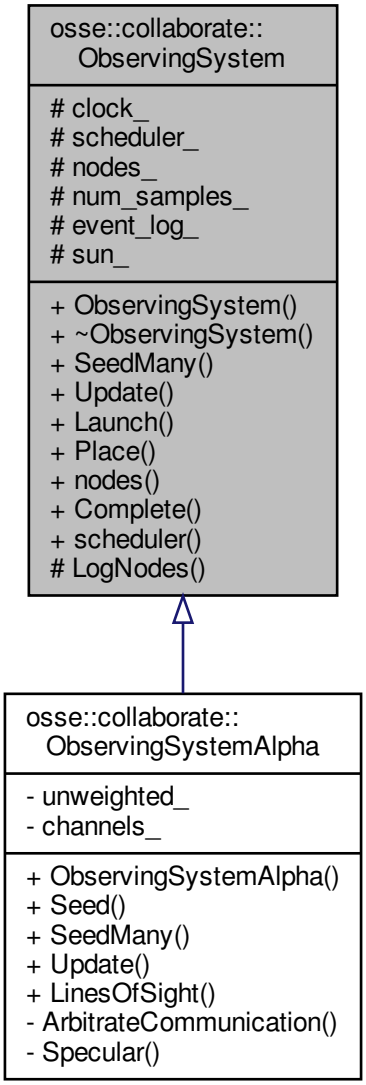
- libs/collaborate/include/collaborate/observing_system_alpha.h
- libs/collaborate/src/observing_system_alpha.cpp

2.25 osse::collaborate::ObservingSystem Class Reference

Abstract satellite network.

`#include <observing_system.h>`

Inheritance diagram for osse::collaborate::ObservingSystem:



Public Member Functions

- ObservingSystem (Sun *_sun, SimulationClock *_clock, Scheduler *_scheduler, EventLogger *_event_log)

  Constructor.
- ~ObservingSystem ()

  Destructor.
- virtual void SeedMany (const uint64_t &_span_s, const uint16_t &_constellation)=0

  Generates random list of samples to start with.

- virtual void Update ()=0

  Update the nodes and protocol.
- void Launch (const std::vector< PlatformOrbit > &_orbits, const uint16_t &_constellation, const bool &_separate, const SubsystemComm &_comm_if, const SubsystemSensing &↩ _sensing_if, const SubsystemPower &_subsystem_power, DataProcessor *_data_processor, DataLogger *_data_log)

  Make new nodes from a list of platforms.
- void Place (const std::vector< PlatformEarth > &_earths, const uint16_t &_constellation, const bool &_separate, const SubsystemComm &_comm_if, const SubsystemSensing &↩ _sensing_if, const SubsystemPower &_subsystem_power, DataProcessor *_data_processor, DataLogger *_data_log)

  Make new nodes from a list of platforms.
- std::vector< Node * > nodes () const

  Get the earth_data of nodes.
- void Complete () const

  Logs the nodes at the end.
- Scheduler * scheduler ()

  Get the scheduler.

Protected Member Functions

- void LogNodes () const

  Logs the nodes.

Protected Attributes

- const SimulationClock * clock_

  SimulationClock.
- Scheduler * scheduler_

  Autonomous network scheduler.
- std::vector< Node * > nodes_

  Ordered list of nodes.
- uint64_t num_samples_

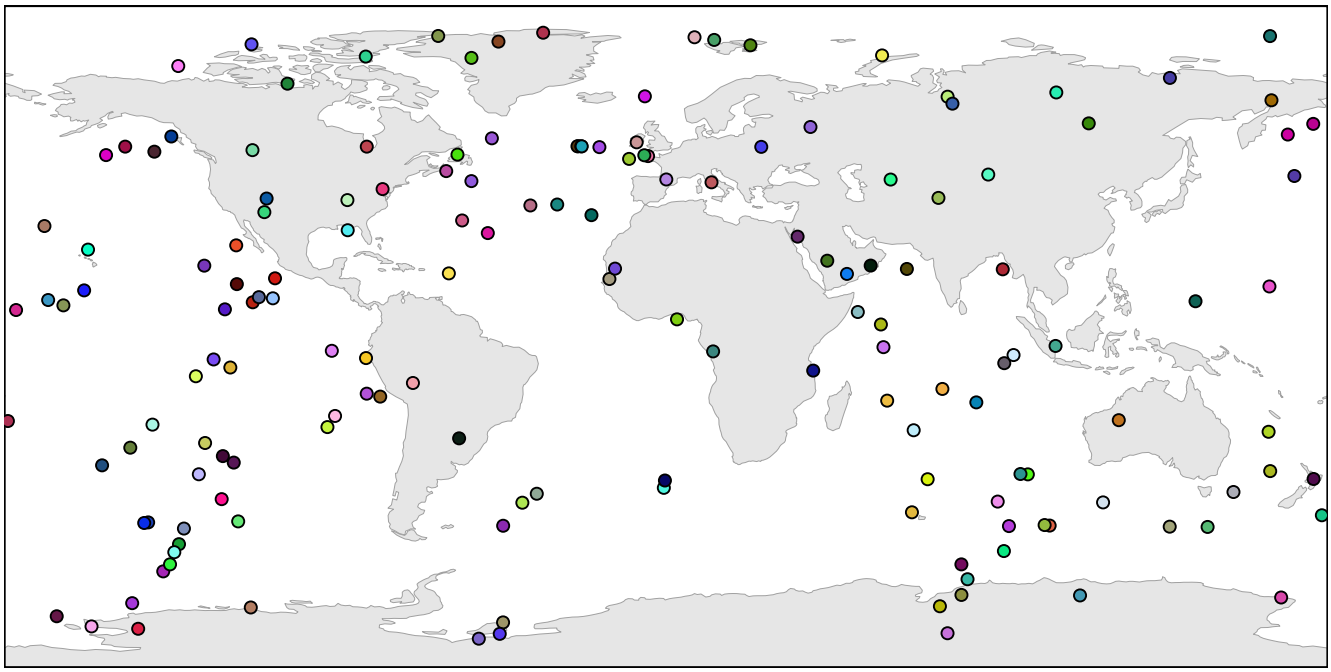
  Number of samples seeded.
- EventLogger * event_log_

  Event logger.
- Sun * sun_

  Sun.

2.25.1 Detailed Description

Abstract satellite network.



2.25.2 Constructor & Destructor Documentation

2.25.2.1 ObservingSystem()

```
osse::collaborate::ObservingSystem::ObservingSystem (
            Sun * _sun,
            SimulationClock * _clock,
            Scheduler * _scheduler,
            EventLogger * _event_log )
```

Constructor.

Parameters

in	*_sun*	Star at the center of the solar system
in	*_clock*	Simulation clock
in	*_scheduler*	Scheduler
in	*_event_log*	Event log

2.25.3 Member Function Documentation

2.25.3.1 Launch()

```
void osse::collaborate::ObservingSystem::Launch (
            const std::vector< PlatformOrbit > & _orbits,
            const uint16_t & _constellation,
            const bool & _separate,
            const SubsystemComm & _comm_if,
            const SubsystemSensing & _sensing_if,
            const SubsystemPower & _subsystem_power,
            DataProcessor * _data_processor,
            DataLogger * _data_log )
```

Make new nodes from a list of platforms.

Parameters

in	*_orbits*	List of orbits
in	*_constellation*	Starting constellation identifier
in	*_separate*	Whether the nodess are separate constellations
in	*_comm_if*	Communication interface
in	*_sensing_if*	Sensing interface
in	*_subsystem_power*	Power subsystem
in	*_data_processor*	Data processor
in	*_data_log*	Data logger

2.25.3.2 nodes()

```
std::vector<Node*> osse::collaborate::ObservingSystem::nodes ( ) const  [inline]
```

Get the earth_data of nodes.

Returns

nodes_ The earth_data of nodes

2.25.3.3 Place()

```
void osse::collaborate::ObservingSystem::Place (
            const std::vector< PlatformEarth > & _earths,
            const uint16_t & _constellation,
            const bool & _separate,
            const SubsystemComm & _comm_if,
            const SubsystemSensing & _sensing_if,
            const SubsystemPower & _subsystem_power,
            DataProcessor * _data_processor,
            DataLogger * _data_log )
```

Make new nodes from a list of platforms.

Parameters

in	*_earths*	List of earth platforms
in	*_constellation*	Starting constellation identifier
in	*_separate*	Whether the nodess are separate constellations
in	*_comm_if*	The communication interface
in	*_sensing_if*	Sensing interface
in	*_subsystem_power*	Power subsystem
in	*_data_processor*	Data processor
in	*_data_log*	Data logger

2.25.3.4 scheduler()

```
Scheduler* osse::collaborate::ObservingSystem::scheduler ( )  [inline]
```

Get the scheduler.

Returns

scheduler_ Scheduler

2.25.3.5 SeedMany()

```
virtual void osse::collaborate::ObservingSystem::SeedMany (
            const uint64_t & _span_s,
            const uint16_t & _constellation )  [pure virtual]
```

Generates random list of samples to start with.

Parameters

in	*_span_s*	Total time span of the simulation (seconds)
in	*_constellation*	Constellation

Implemented in osse::collaborate::ObservingSystemAlpha.

The documentation for this class was generated from the following files:

- libs/collaborate/include/collaborate/observing_system.h
- libs/collaborate/src/observing_system.cpp

2.26 osse::collaborate::OrbitalState Class Reference

The position and orientation of a node.

`#include <orbital_state.h>`

Public Member Functions

- OrbitalState (double _x_m, double _y_m, double _z_m, double _latitude_rad, double _longitude_rad, double _altitude_m, double _dx_m_per_s, double _dy_m_per_s, double _dz_m_per_s, double _roll_rad, double _pitch_rad, double _yaw_rad)

  Constructor.
- ReferenceFrame CalculatePlatformOrbitReferenceFrame () const

  Calculates the orbit reference_frame.
- void Update (double _x_m, double _y_m, double _z_m, double _latitude_rad, double _longitude_rad, double _altitude_m, double _dx_m_per_s, double _dy_m_per_s, double _dz_m_per_s)

  Update the orbital_state.
- std::vector< double > ObtainLog () const

  Logs the OrbitalState.
- std::vector< double > ObtainGeodeticLog () const

  Logs the OrbitalState Geodetic Coordinates.
- const Vector & position_m_rad () const

  Get The position (meters and radians)
- const Vector & velocity_m_per_s () const

  Get the velocity.
- const Geodetic & geodetic_rad_m () const

  Get The geodetic position (radians and meters)
- const ReferenceFrame & orbit_frame () const

  Get the orbit_frame.
- const ReferenceFrame & body_frame () const

  Get the body_frame.

Private Attributes

- Vector position_m_rad_

  Position (meters and radians)
- Vector velocity_m_per_s_

  Velocity (meters per second)
- Geodetic geodetic_rad_m_

  Geodetic position (radians and meters)
- ReferenceFrame orbit_frame_

  PlatformOrbit ReferenceFrame (unit)
- ReferenceFrame body_frame_

  Body ReferenceFrame (unit)

2.26.1 Detailed Description

The position and orientation of a node.

2.26.2 Constructor & Destructor Documentation

2.26.2.1 OrbitalState()

```
osse::collaborate::OrbitalState::OrbitalState (
            double _x_m,
            double _y_m,
            double _z_m,
            double _latitude_rad,
            double _longitude_rad,
            double _altitude_m,
            double _dx_m_per_s,
            double _dy_m_per_s,
            double _dz_m_per_s,
            double _roll_rad,
            double _pitch_rad,
            double _yaw_rad )
```

Constructor.

Parameters

in	*_x_m*	X-component of position (meters)
in	*_y_m*	Y-component of position (meters)

Parameters

in	*_z_m*	Z-component of position (meters)
in	*_latitude_rad*	Latitude (radians)
in	*_longitude_rad*	Longitude (radians)
in	*_altitude_m*	Altitude (meters)
in	*_dx_m_per_s*	X-component of velocity (meters per second)
in	*_dy_m_per_s*	Y-component of velocity (meters per second)
in	*_dz_m_per_s*	Z-component of velocity (meters per second)
in	*_roll_rad*	Satellite's roll angle (radians)
in	*_pitch_rad*	Satellite's pitch angle (radians)
in	*_yaw_rad*	Satellite's yaw angle (radians)

2.26.3 Member Function Documentation

2.26.3.1 body_frame()

```
const ReferenceFrame& osse::collaborate::OrbitalState::body_frame ( ) const  [inline]
```

Get the body_frame.

Returns

body_frame_ Body_reference_frame (unit)

2.26.3.2 CalculatePlatformOrbitReferenceFrame()

```
ReferenceFrame osse::collaborate::OrbitalState::CalculatePlatformOrbitReferenceFrame ( ) const
```

Calculates the orbit reference_frame.

Returns

Orbit reference_frame

$$\hat{y} = \frac{-\vec{p} \times \vec{v}}{|-\vec{p} \times \vec{v}|}$$

$$\hat{z} = \frac{-\vec{p}}{|-\vec{p}|}$$

$$\hat{x} = \hat{y} \times \hat{z}$$

2.26.3.3 geodetic_rad_m()

```
const Geodetic& osse::collaborate::OrbitalState::geodetic_rad_m ( ) const  [inline]
```

Get The geodetic position (radians and meters)

Returns

geodetic_rad_m_ Geodetic position (radians and meters)

2.26.3.4 ObtainGeodeticLog()

```
std::vector< double > osse::collaborate::OrbitalState::ObtainGeodeticLog ( ) const
```

Logs the OrbitalState Geodetic Coordinates.

Returns

A vector of doubles

2.26.3.5 ObtainLog()

```
std::vector< double > osse::collaborate::OrbitalState::ObtainLog ( ) const
```

Logs the OrbitalState.

Returns

A vector of doubles

2.26.3.6 orbit_frame()

```
const ReferenceFrame& osse::collaborate::OrbitalState::orbit_frame ( ) const  [inline]
```

Get the orbit_frame.

Returns

orbit_frame_ Orbit_reference_frame (unit)

2.26.3.7 position_m_rad()

```
const Vector& osse::collaborate::OrbitalState::position_m_rad ( ) const  [inline]
```

Get The position (meters and radians)

Returns

position_m_rad_ Position (meters and radians)

2.26.3.8 Update()

```
void osse::collaborate::OrbitalState::Update (
            double _x_m,
            double _y_m,
            double _z_m,
            double _latitude_rad,
            double _longitude_rad,
            double _altitude_m,
            double _dx_m_per_s,
            double _dy_m_per_s,
            double _dz_m_per_s )
```

Update the orbital_state.

Parameters

in	*_x_m*	X-component of position (meters)
in	*_y_m*	Y-component of position (meters)
in	*_z_m*	Z-component of position (meters)
in	*_latitude_rad*	Latitude (radians)
in	*_longitude_rad*	Longitude (radians)
in	*_altitude_m*	Altitude (meters)
in	*_dx_m_per_s*	X-component of velocity (meters per second)
in	*_dy_m_per_s*	Y-component of velocity (meters per second)
in	*_dz_m_per_s*	Z-component of velocity (meters per second)

2.26.3.9 velocity_m_per_s()

```
const Vector& osse::collaborate::OrbitalState::velocity_m_per_s ( ) const  [inline]
```

Get the velocity.

Returns

velocity_ Velocity (meters per second)

The documentation for this class was generated from the following files:

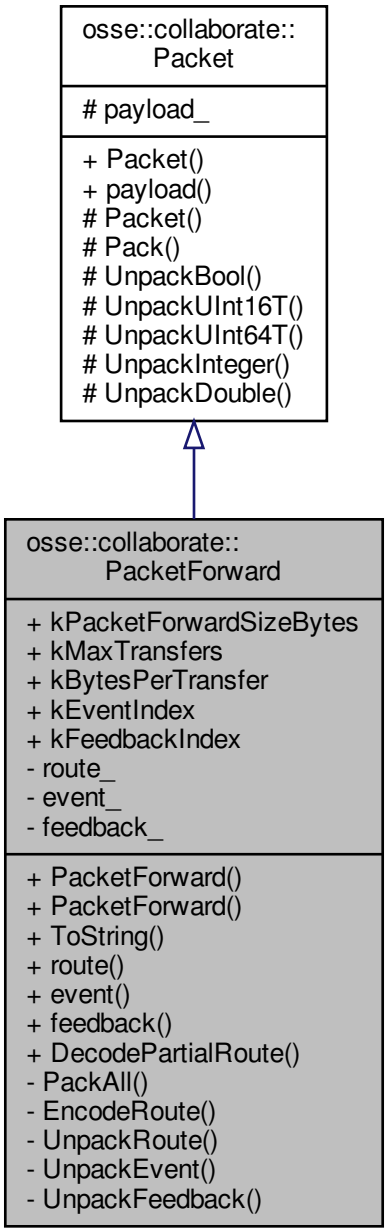
- libs/collaborate/include/collaborate/orbital_state.h
- libs/collaborate/src/orbital_state.cpp

2.27 osse::collaborate::PacketForward Class Reference

A packet of control data.

`#include <packet_forward.h>`

Inheritance diagram for osse::collaborate::PacketForward:



Public Types

- typedef std::pair< uint16_t, uint64_t > Event

  A route containing node and transfer pairs.
- typedef std::vector< Event > PartialRoute

  A partial route containing node and transfer pairs.

- typedef std::array< Event, kMaxTransfers > Route

  A route containing node and transfer pairs.

Public Member Functions

- PacketForward (const std::vector< uint8_t > &_payload)

  Constructor from payload.
- PacketForward (const PartialRoute &_partial_route, const Event &_event, const uint16_t &_feedback)

  Constructor from data members.
- std::string ToString () const

  Converts the packet to a string.
- const Route & route () const

  Gets route.
- const Event & event () const

  Gets event.
- const uint16_t & feedback () const

  Gets feedback.
- PartialRoute DecodePartialRoute ()

  Decodes the route into a standard vector.

Static Public Attributes

- static constexpr int kPacketForwardSizeBytes = 312

  The size (bytes)
- static constexpr int kMaxTransfers = 30

  The maximum size of a route.
- static constexpr int kBytesPerTransfer = 10

  The number of bytes per transfer.
- static constexpr int kEventIndex = 300

  The index of the event.
- static constexpr int kFeedbackIndex = 310

  The index of the feedback.

Private Member Functions

- std::vector< uint8_t > PackAll (const PartialRoute &_partial_route, const Event &_event, const uint16_t &_feedback) const

  Packs member elements into a payload.
- Route EncodeRoute (const PartialRoute &_partial_route) const

  Translates a vector to an array.
- Route UnpackRoute (const std::vector< uint8_t > &_payload) const

  Unpacks the route from the payload.
- Event UnpackEvent (const std::vector< uint8_t > &_payload) const

  Unpacks the event from the payload.
- uint16_t UnpackFeedback (const std::vector< uint8_t > &_payload) const

  Unpacks the feedback from the payload.

Private Attributes

- Route route_

  Route containing individual transfer.
- Event event_

  Node index and time of sensor reading.
- uint16_t feedback_

  Feedback node index.

Additional Inherited Members

2.27.1 Detailed Description

A packet of control data.

2.27.2 Constructor & Destructor Documentation

2.27.2.1 PacketForward() [1/2]

```
osse::collaborate::PacketForward::PacketForward (
            const std::vector< uint8_t > & _payload )  [explicit]
```

Constructor from payload.

Parameters

in	*_payload*	Payload

2.27.2.2 PacketForward() [2/2]

```
osse::collaborate::PacketForward::PacketForward (
            const PartialRoute & _partial_route,
            const Event & _event,
            const uint16_t & _feedback )
```

Constructor from data members.

Parameters

in	*_partial_route*	Route
in	*_event*	Measurement event
in	*_feedback*	The index of the feedback node

2.27.3 Member Function Documentation

2.27.3.1 DecodePartialRoute()

```
PacketForward::PartialRoute osse::collaborate::PacketForward::DecodePartialRoute ( )
```

Decodes the route into a standard vector.

Returns

Route as a vector

2.27.3.2 EncodeRoute()

```
PacketForward::Route osse::collaborate::PacketForward::EncodeRoute (
            const PartialRoute & _partial_route ) const  [private]
```

Translates a vector to an array.

Parameters

in	*_partial_route*	Route as a vector

Returns

Route (array)

2.27.3.3 event()

```
const Event& osse::collaborate::PacketForward::event ( ) const  [inline]
```

Gets event.

Returns

event_ Event

2.27.3.4 feedback()

```
const uint16_t& osse::collaborate::PacketForward::feedback ( ) const  [inline]
```

Gets feedback.

Returns

feedback_ Feedback

2.27.3.5 PackAll()

```
std::vector< uint8_t > osse::collaborate::PacketForward::PackAll (
            const PartialRoute & _partial_route,
            const Event & _event,
            const uint16_t & _feedback ) const  [private]
```

Packs member elements into a payload.

Parameters

in	*_partial_route*	Partial route
in	*_event*	Event
in	*_feedback*	Feedback

Returns

Payload

2.27.3.6 route()

```
const Route& osse::collaborate::PacketForward::route ( ) const  [inline]
```

Gets route.

Returns

route_ Route

2.27.3.7 ToString()

```
std::string osse::collaborate::PacketForward::ToString ( ) const
```

Converts the packet to a string.

Returns

Packet as a string

2.27.3.8 UnpackEvent()

```
PacketForward::Event osse::collaborate::PacketForward::UnpackEvent (
            const std::vector< uint8_t > & _payload ) const  [private]
```

Unpacks the event from the payload.

Parameters

in	*_payload*	Payload

Returns

Event

2.27.3.9 UnpackFeedback()

```
uint16_t osse::collaborate::PacketForward::UnpackFeedback (
            const std::vector< uint8_t > & _payload ) const  [private]
```

Unpacks the feedback from the payload.

Parameters

in	*_payload*	Payload

Returns

Feedback

2.27.3.10 UnpackRoute()

```
PacketForward::Route osse::collaborate::PacketForward::UnpackRoute (
            const std::vector< uint8_t > & _payload ) const  [private]
```

Unpacks the route from the payload.

Parameters

in	*_payload*	Payload

Returns

Route (array)

The documentation for this class was generated from the following files:

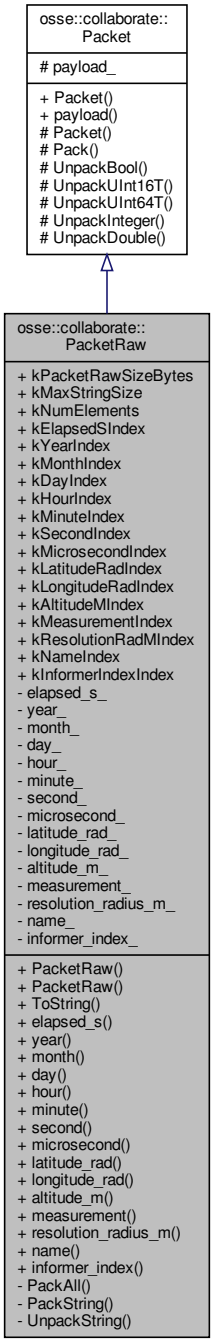
- libs/collaborate/include/collaborate/packet_forward.h
- libs/collaborate/src/packet_forward.cpp

2.28 osse::collaborate::PacketRaw Class Reference

A packet of raw measurement data.

`#include <packet_raw.h>`

Inheritance diagram for osse::collaborate::PacketRaw:



Public Member Functions

- PacketRaw (std::vector< uint8_t > _payload)

  Constructor from payload.
- PacketRaw (const uint64_t &_elapsed_s, const int &_year, const int &_month, const int &↩ _day, const int &_hour, const int &_minute, const int &_second, const int &_microsecond, const double &_latitude_rad, const double &_longitude_rad, const double &_altitude_m, const double &_measurement, const double &_resolution_radius_m, const std::string &_name, const uint16_t &_informer_index)

  Constructor from data members.
- std::string ToString () const

  Converts the packet to a string.
- const uint64_t & elapsed_s () const

  Get Year.
- const int & year () const

  Get Year.
- const int & month () const

  Get Month.
- const int & day () const

  Get Day.
- const int & hour () const

  Get Hour.
- const int & minute () const

  Get Minute.
- const int & second () const

  Get Second.
- const int & microsecond () const

  Get Microsecond.
- const double & latitude_rad () const

  Get Latitude (radians)
- const double & longitude_rad () const

  Get Longitude (radians)
- const double & altitude_m () const

  Get Altitude (meters)
- const double & measurement () const

  Get Measurement.
- const double & resolution_radius_m () const

  Get Resolution radius (meters)
- const std::string & name () const

  Get Variable name of the data-set.
- const uint16_t & informer_index () const

  Get informer index.

Static Public Attributes

- static constexpr int kPacketRawSizeBytes = 108

  Maximum size of a route.
- static constexpr int kMaxStringSize = 30

  Maximum size of a route.
- static constexpr int kNumElements = 15

  Number of element in a raw packet.
- static constexpr int kElapsedSIndex = 0

  Index of elapsed (seconds)
- static constexpr int kYearIndex = 8

  Index of year.
- static constexpr int kMonthIndex = 12

  Index of month.
- static constexpr int kDayIndex = 16

  Index of day.
- static constexpr int kHourIndex = 20

  Index of hour.
- static constexpr int kMinuteIndex = 24

  Index of minute.
- static constexpr int kSecondIndex = 28

  Index of second.
- static constexpr int kMicrosecondIndex = 32

  Index of microsecond.
- static constexpr int kLatitudeRadIndex = 36

  Index of latitude (radians)
- static constexpr int kLongitudeRadIndex = 44

  Index of longitude (radians)
- static constexpr int kAltitudeMIndex = 52

  Index of altitude (meters)
- static constexpr int kMeasurementIndex = 60

  Index of measurement.
- static constexpr int kResolutionRadMIndex = 68

  Index of resolution radius (meters)
- static constexpr int kNameIndex = 76

  Index of name.
- static constexpr int kInformerIndexIndex = 106

  Index of informer index.

Private Member Functions

- std::vector< uint8_t > PackAll (const uint64_t &_elapsed_s, const int &_year, const int &↩ _month, const int &_day, const int &_hour, const int &_minute, const int &_second, const int &_microsecond, const double &_latitude_rad, const double &_longitude_rad, const double &_altitude_m, const double &_measurement, const double &_resolution_radius_m, const std↩ ::string &_name, const uint16_t &_informer_index)

  Inserts data members into the payload.
- void PackString (const std::string &_string, std::vector< uint8_t > *_payload) const

  Inserts a string into the payload.
- std::string UnpackString (const std::vector< uint8_t > &_payload, const uint16_t &_index) const

  Unpacks a string from the payload.

Private Attributes

- uint64_t elapsed_s_

  Time elapsed since the beginning of the simulation (seconds)
- int year_

  Year.
- int month_

  Month.
- int day_

  Day.
- int hour_

  Hour.
- int minute_

  Minute.
- int second_

  Second.
- int microsecond_

  Microsecond.
- double latitude_rad_

  Latitude (radians)
- double longitude_rad_

  Longitude (radians)
- double altitude_m_

  Altitude (meters)
- double measurement_

  Measurement.
- double resolution_radius_m_

  Resolution radius (meters)

- std::string name_

  Variable name of the data-set.
- uint16_t informer_index_

  Informer index.

Additional Inherited Members

2.28.1 Detailed Description

A packet of raw measurement data.

2.28.2 Constructor & Destructor Documentation

2.28.2.1 PacketRaw() [1/2]

```
osse::collaborate::PacketRaw::PacketRaw (
            std::vector< uint8_t > _payload ) [explicit]
```

Constructor from payload.

Parameters

in	*_payload*	Payload

2.28.2.2 PacketRaw() [2/2]

```
osse::collaborate::PacketRaw::PacketRaw (
            const uint64_t & _elapsed_s,
            const int & _year,
            const int & _month,
            const int & _day,
            const int & _hour,
            const int & _minute,
            const int & _second,
            const int & _microsecond,
            const double & _latitude_rad,
            const double & _longitude_rad,
            const double & _altitude_m,
            const double & _measurement,
            const double & _resolution_radius_m,
            const std::string & _name,
            const uint16_t & _informer_index )
```

Constructor from data members.

Parameters

in	*_elapsed_s*	Time elapsed in the simulation (seconds)
in	*_year*	Year
in	*_month*	Month
in	*_day*	Day
in	*_hour*	Hour
in	*_minute*	Minute
in	*_second*	Second
in	*_microsecond*	Microsecond
in	*_latitude_rad*	Latitude (radians)
in	*_longitude_rad*	Longitude (radians)
in	*_altitude_m*	Altitude (meters)
in	*_measurement*	Measurement
in	*_resolution_radius_m*	Resolution (meters)
in	*_name*	Variable name of data set
in	*_informer_index*	Informer index index

2.28.3 Member Function Documentation

2.28.3.1 altitude_m()

```
const double& osse::collaborate::PacketRaw::altitude_m ( ) const  [inline]
```

Get Altitude (meters)

Returns

altitude_m_ Altitude (meters)

2.28.3.2 day()

```
const int& osse::collaborate::PacketRaw::day ( ) const  [inline]
```

Get Day.

Returns

day_ Day

2.28.3.3 elapsed_s()

```
const uint64_t& osse::collaborate::PacketRaw::elapsed_s ( ) const  [inline]
```

Get Year.

Returns

year_ Year

2.28.3.4 hour()

```
const int& osse::collaborate::PacketRaw::hour ( ) const  [inline]
```

Get Hour.

Returns

hour_ Hour

2.28.3.5 informer_index()

```
const uint16_t& osse::collaborate::PacketRaw::informer_index ( ) const  [inline]
```

Get informer index.

Returns

informer_index_index_ Informer index

2.28.3.6 latitude_rad()

```
const double& osse::collaborate::PacketRaw::latitude_rad ( ) const  [inline]
```

Get Latitude (radians)

Returns

latitude_rad_ Latitude (radians)

2.28.3.7 longitude_rad()

```
const double& osse::collaborate::PacketRaw::longitude_rad ( ) const  [inline]
```

Get Longitude (radians)

Returns

longitude_rad_ Longitude (radians)

2.28.3.8 measurement()

```
const double& osse::collaborate::PacketRaw::measurement ( ) const  [inline]
```

Get Measurement.

Returns

measurement_ Measurement

2.28.3.9 microsecond()

```
const int& osse::collaborate::PacketRaw::microsecond ( ) const  [inline]
```

Get Microsecond.

Returns

microsecond_ Microsecond

2.28.3.10 minute()

```
const int& osse::collaborate::PacketRaw::minute ( ) const  [inline]
```

Get Minute.

Returns

minute_ Minute

2.28.3.11 month()

```
const int& osse::collaborate::PacketRaw::month ( ) const  [inline]
```

Get Month.

Returns

month_ Month

2.28.3.12 name()

```
const std::string& osse::collaborate::PacketRaw::name ( ) const  [inline]
```

Get Variable name of the data-set.

Returns

name_ Variable name of the data-set

2.28.3.13 PackAll()

```
std::vector< uint8_t > osse::collaborate::PacketRaw::PackAll (
            const uint64_t & _elapsed_s,
            const int & _year,
            const int & _month,
            const int & _day,
            const int & _hour,
            const int & _minute,
            const int & _second,
            const int & _microsecond,
            const double & _latitude_rad,
            const double & _longitude_rad,
            const double & _altitude_m,
            const double & _measurement,
            const double & _resolution_radius_m,
            const std::string & _name,
            const uint16_t & _informer_index ) [private]
```

Inserts data members into the payload.

Parameters

in	*_elapsed_s*	Time elapsed in the simulation (seconds)
in	*_year*	Year
in	*_month*	Month
in	*_day*	Day
in	*_hour*	Hour
in	*_minute*	Minute
in	*_second*	Second
in	*_microsecond*	Microsecond
in	*_latitude_rad*	Latitude (radians)
in	*_longitude_rad*	Longitude (radians)
in	*_altitude_m*	Altitude (meters)
in	*_measurement*	Measurement
in	*_resolution_radius_m*	Resolution (meters)
in	*_name*	Variable name of data set
in	*_informer_index*	Informer index index

Returns

Serialized version of the raw packet

2.28.3.14 PackString()

```
void osse::collaborate::PacketRaw::PackString (
            const std::string & _string,
            std::vector< uint8_t > * _payload ) const  [private]
```

Inserts a string into the payload.

Parameters

in	*_string*	String
in	*_payload*	Payload

2.28.3.15 resolution_radius_m()

```
const double& osse::collaborate::PacketRaw::resolution_radius_m ( ) const  [inline]
```

Get Resolution radius (meters)

Returns

resolution_radius_m_ Resolution radius (meters)

2.28.3.16 second()

```
const int& osse::collaborate::PacketRaw::second ( ) const  [inline]
```

Get Second.

Returns

second_ Second

2.28.3.17 ToString()

```
std::string osse::collaborate::PacketRaw::ToString ( ) const
```

Converts the packet to a string.

Returns

The packet as a string

2.28.3.18 UnpackString()

```
std::string osse::collaborate::PacketRaw::UnpackString (
            const std::vector< uint8_t > & _payload,
            const uint16_t & _index ) const  [private]
```

Unpacks a string from the payload.

Parameters

in	*_payload*	Payload
in	*_index*	Index

Returns

Unpacked string

2.28.3.19 year()

```
const int& osse::collaborate::PacketRaw::year ( ) const  [inline]
```

Get Year.

Returns

year_ Year

The documentation for this class was generated from the following files:

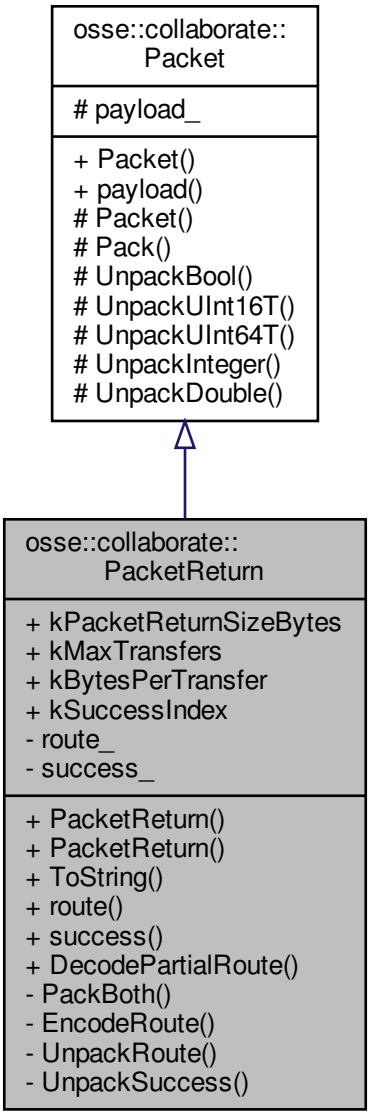
- libs/collaborate/include/collaborate/packet_raw.h
- libs/collaborate/src/packet_raw.cpp

2.29 osse::collaborate::PacketReturn Class Reference

A packet of return data.

`#include <packet_return.h>`

Inheritance diagram for osse::collaborate::PacketReturn:



Public Types

- typedef std::pair< uint16_t, uint64_t > Event

A route containing node and transfer pairs.

- typedef std::vector< Event > PartialRoute

  A partial route containing node and transfer pairs.

- typedef std::array< Event, kMaxTransfers > Route

  A route containing node and transfer pairs.

Public Member Functions

- PacketReturn (const std::vector< uint8_t > &_payload)

  Constructor from payload.

- PacketReturn (const PartialRoute &_partial_route, const std::pair< bool, uint16_t > &_success)

  Constructor from data members.

- std::string ToString () const

  Converts the packet to a string.

- const Route & route () const

  Gets route.

- const std::pair< bool, uint16_t > & success () const

  Gets success.

- PartialRoute DecodePartialRoute ()

  Decodes the route into a standard vector.

Static Public Attributes

- static constexpr int kPacketReturnSizeBytes = 303

  The size (bytes)

- static constexpr int kMaxTransfers = 30

  The maximum size of a route.

- static constexpr int kBytesPerTransfer = 10

  The number of bytes per transfer.

- static constexpr int kSuccessIndex = kMaxTransfers * kBytesPerTransfer

  The index of the event.

Private Member Functions

- std::vector< uint8_t > PackBoth (const PartialRoute &_partial_route, const std::pair< bool, uint16_t > &_success) const

  Packs member elements into a payload.

- Route EncodeRoute (const PartialRoute &_partial_route) const

  Translates a vector to an array.

- Route UnpackRoute (const std::vector< uint8_t > &_payload) const

  Unpacks the route from the payload.

- std::pair< bool, uint16_t > UnpackSuccess (const std::vector< uint8_t > &_payload) const

  Unpacks the success from the payload.

Private Attributes

- Route route_

  Route containing individual transfer.
- std::pair< bool, uint16_t > success_

  Whether a measurements exceed the desired threshold.

Additional Inherited Members

2.29.1 Detailed Description

A packet of return data.

2.29.2 Constructor & Destructor Documentation

2.29.2.1 PacketReturn() [1/2]

```
osse::collaborate::PacketReturn::PacketReturn (
            const std::vector< uint8_t > & _payload ) [explicit]
```

Constructor from payload.

Parameters

in	*_payload*	Payload

2.29.2.2 PacketReturn() [2/2]

```
osse::collaborate::PacketReturn::PacketReturn (
            const PartialRoute & _partial_route,
            const std::pair< bool, uint16_t > & _success )
```

Constructor from data members.

Parameters

in	*_partial_route*	Route
in	*_success*	Whether the threshold was exceeded

2.29.3 Member Function Documentation

2.29.3.1 DecodePartialRoute()

```
PacketReturn::PartialRoute osse::collaborate::PacketReturn::DecodePartialRoute ( )
```

Decodes the route into a standard vector.

Returns

Route as a vector

2.29.3.2 EncodeRoute()

```
PacketReturn::Route osse::collaborate::PacketReturn::EncodeRoute (
            const PartialRoute & _partial_route ) const  [private]
```

Translates a vector to an array.

Parameters

in	*_partial_route*	Route as a vector

Returns

Route (array)

2.29.3.3 PackBoth()

```
std::vector< uint8_t > osse::collaborate::PacketReturn::PackBoth (
            const PartialRoute & _partial_route,
            const std::pair< bool, uint16_t > & _success ) const  [private]
```

Packs member elements into a payload.

Parameters

in	*_partial_route*	Partial route
in	*_success*	Success

Returns

Payload

2.29.3.4 route()

```
const Route& osse::collaborate::PacketReturn::route ( ) const  [inline]
```

Gets route.

Returns

route_ Route

2.29.3.5 success()

```
const std::pair<bool, uint16_t>& osse::collaborate::PacketReturn::success ( ) const  [inline]
```

Gets success.

Returns

success_ Success

2.29.3.6 ToString()

```
std::string osse::collaborate::PacketReturn::ToString ( ) const
```

Converts the packet to a string.

Returns

Packet as a string

2.29.3.7 UnpackRoute()

```
PacketReturn::Route osse::collaborate::PacketReturn::UnpackRoute (
            const std::vector< uint8_t > & _payload ) const  [private]
```

Unpacks the route from the payload.

Parameters

in	*_payload*	Payload

Returns

Route (array)

2.29.3.8 UnpackSuccess()

```
std::pair< bool, uint16_t > osse::collaborate::PacketReturn::UnpackSuccess (
            const std::vector< uint8_t > & _payload ) const  [private]
```

Unpacks the success from the payload.

Parameters

in	*_payload*	Payload

Returns

Success

The documentation for this class was generated from the following files:

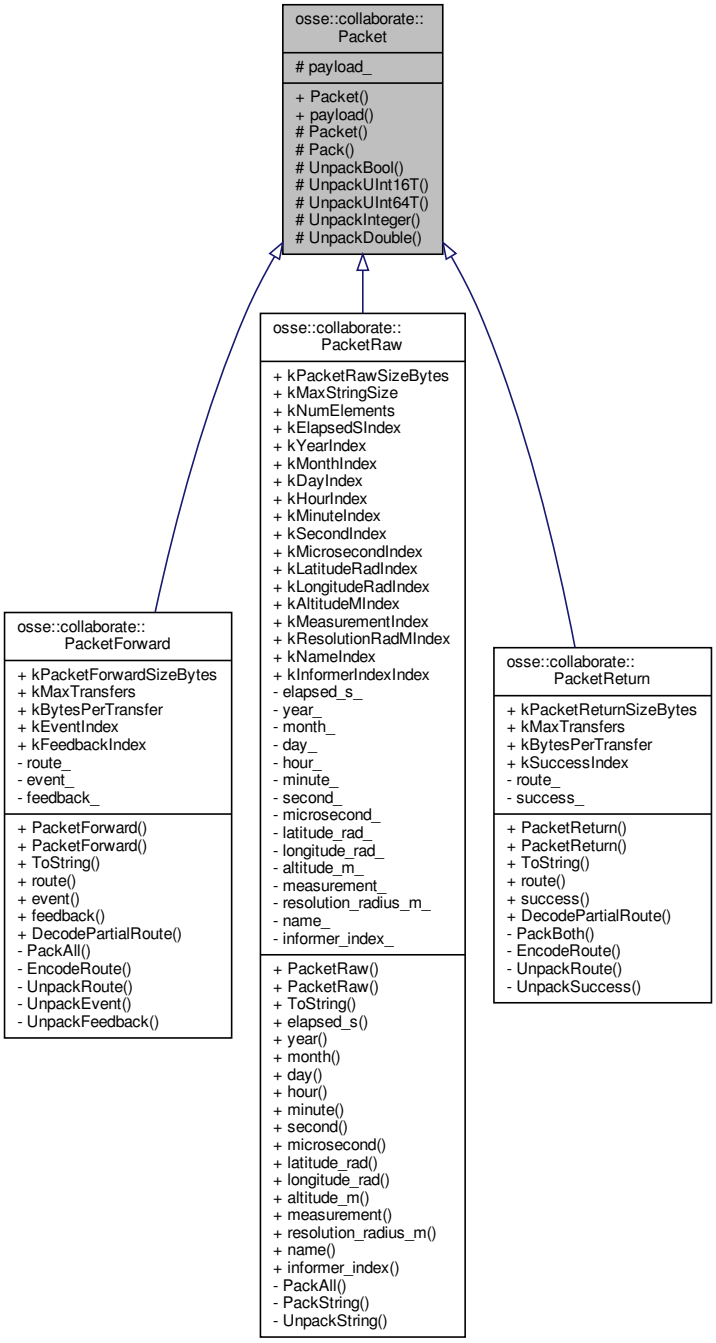
- libs/collaborate/include/collaborate/packet_return.h
- libs/collaborate/src/packet_return.cpp

2.30 osse::collaborate::Packet Class Reference

An abstract container for information.

```
#include <packet.h>
```

Inheritance diagram for osse::collaborate::Packet:



Public Member Functions

- Packet ()

  Default Constructor.
- const std::vector< uint8_t > & payload () const

  Get payload.

Protected Member Functions

- Packet (const std::vector< uint8_t > &_payload)

  Default Constructor.
- template<class T >
  void Pack (const T &_value, std::vector< uint8_t > *_payload) const

  Inserts a generic value into the payload.
- bool UnpackBool (const std::vector< uint8_t > &_payload, const uint16_t &_index) const

  Unpacks a boolean value from the _payload.
- uint16_t UnpackUInt16T (const std::vector< uint8_t > &_payload, const uint16_t &_index) const

  Unpacks a short unsigned integer from the _payload.
- uint64_t UnpackUInt64T (const std::vector< uint8_t > &_payload, const uint16_t &_index) const

  Unpacks a long unsigned integer from the payload.
- int UnpackInteger (const std::vector< uint8_t > &_payload, const uint16_t &_index) const

  Unpacks an integer from the payload.
- double UnpackDouble (const std::vector< uint8_t > &_payload, const uint16_t &_index) const

  Unpacks a double from the payload.

Protected Attributes

- std::vector< uint8_t > payload_

  Data contents (a vector of bytes)

2.30.1 Detailed Description

An abstract container for information.

2.30.2 Constructor & Destructor Documentation

2.30.2.1 Packet()

```
osse::collaborate::Packet::Packet (
            const std::vector< uint8_t > & _payload )  [explicit], [protected]
```

Default Constructor.

Parameters

in	*_payload*	The payload

2.30.3 Member Function Documentation

2.30.3.1 Pack()

```
template<class T >
void osse::collaborate::Packet::Pack (
            const T & _value,
            std::vector< uint8_t > * _payload ) const  [inline], [protected]
```

Inserts a generic value into the payload.

Parameters

in	*_value*	Value
in	*_payload*	Payload

2.30.3.2 payload()

```
const std::vector<uint8_t>& osse::collaborate::Packet::payload ( ) const  [inline]
```

Get payload.

Returns

payload_ Payload

2.30.3.3 UnpackBool()

```
bool osse::collaborate::Packet::UnpackBool (
            const std::vector< uint8_t > & _payload,
            const uint16_t & _index ) const  [protected]
```

Unpacks a boolean value from the _payload.

Parameters

in	*_payload*	Payload
in	*_index*	Index in the payload vector

Returns

Unpacked boolean value

2.30.3.4 UnpackDouble()

```
double osse::collaborate::Packet::UnpackDouble (
            const std::vector< uint8_t > & _payload,
            const uint16_t & _index ) const  [protected]
```

Unpacks a double from the payload.

Parameters

in	*_payload*	Payload
in	*_index*	Index in the payload vector

Returns

Unpacked double

2.30.3.5 UnpackInteger()

```
int osse::collaborate::Packet::UnpackInteger (
            const std::vector< uint8_t > & _payload,
            const uint16_t & _index ) const  [protected]
```

Unpacks an integer from the payload.

Parameters

in	*_payload*	Payload
in	*_index*	Index in the payload vector

Returns

Unpacked integer

2.30.3.6 UnpackUInt16T()

```
uint16_t osse::collaborate::Packet::UnpackUInt16T (
            const std::vector< uint8_t > & _payload,
            const uint16_t & _index ) const  [protected]
```

Unpacks a short unsigned integer from the _payload.

Parameters

in	*_payload*	Payload
in	*_index*	Index in the payload vector

Returns

Unpacked short unsigned integer

2.30.3.7 UnpackUInt64T()

```
uint64_t osse::collaborate::Packet::UnpackUInt64T (
            const std::vector< uint8_t > & _payload,
            const uint16_t & _index ) const  [protected]
```

Unpacks a long unsigned integer from the payload.

Parameters

in	*_payload*	Payload
in	*_index*	Index in the payload vector

Returns

Unpacked long unsigned integer

The documentation for this class was generated from the following files:

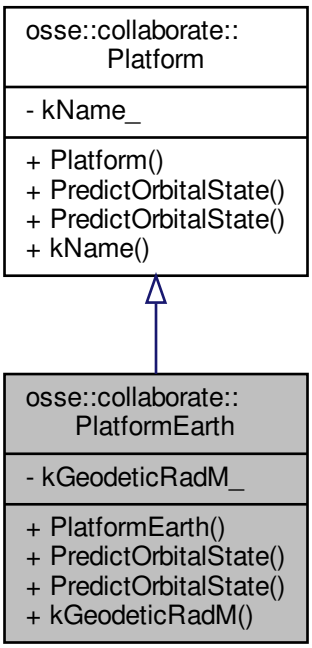
- libs/collaborate/include/collaborate/packet.h
- libs/collaborate/src/packet.cpp

2.31 osse::collaborate::PlatformEarth Class Reference

Propogates the position of a stationary object on Earth's surface.

`#include <platform_earth.h>`

Inheritance diagram for osse::collaborate::PlatformEarth:



Public Member Functions

- PlatformEarth (const std::string &_name, const double &_latitude_rad, const double &_↩ longitude_rad, const double &_altitude_m)

  Constructor From LLH.
- OrbitalState PredictOrbitalState (const SimulationClock &_clock, const uint64_t &_time_s) const

  Get the orbital_state of an object using the current clock value.
- void PredictOrbitalState (const SimulationClock &_clock, const uint64_t &_time_s, Orbital↩ State *_orbital_state) const

  Update the orbital_state of an object the current clock time.
- const Geodetic & kGeodeticRadM () const

  Get geodetic position.

Private Attributes

- const Geodetic kGeodeticRadM_

  Geodetic position.

2.31.1 Detailed Description

Propogates the position of a stationary object on Earth's surface.

2.31.2 Constructor & Destructor Documentation

2.31.2.1 PlatformEarth()

```
osse::collaborate::PlatformEarth::PlatformEarth (
            const std::string & _name,
            const double & _latitude_rad,
            const double & _longitude_rad,
            const double & _altitude_m )
```

Constructor From LLH.

Parameters

in	*_name*	Name of the platform
in	*_latitude_rad*	Latitude (radians)
in	*_longitude_rad*	Longitude (radians)
in	*_altitude_m*	Altitude (meters)

2.31.3 Member Function Documentation

2.31.3.1 kGeodeticRadM()

```
const Geodetic& osse::collaborate::PlatformEarth::kGeodeticRadM ( ) const  [inline]
```

Get geodetic position.

Returns

geodetic_ Geodetic position

2.31.3.2 PredictOrbitalState() [1/2]

```
OrbitalState osse::collaborate::PlatformEarth::PredictOrbitalState (
            const SimulationClock & _clock,
            const uint64_t & _time_s ) const  [virtual]
```

Get the orbital_state of an object using the current clock value.

Parameters

in	*_clock*	Simulation clock
in	*_time_s*	Time offset from the current time (seconds)

Returns

orbital_state OrbitalState of an obect at the specified time

Implements osse::collaborate::Platform.

2.31.3.3 PredictOrbitalState() [2/2]

```
void osse::collaborate::PlatformEarth::PredictOrbitalState (
            const SimulationClock & _clock,
            const uint64_t & _time_s,
            OrbitalState * _orbital_state ) const  [virtual]
```

Update the orbital_state of an object the current clock time.

Parameters

in	*_clock*	Simulation clock
in	*_time_s*	Time offset from the current time (seconds)
in	*_orbital_state*	OrbitalState

Implements osse::collaborate::Platform.

The documentation for this class was generated from the following files:

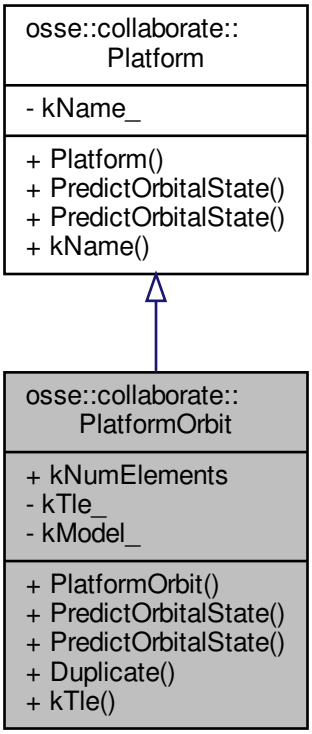
- libs/collaborate/include/collaborate/platform_earth.h
- libs/collaborate/src/platform_earth.cpp

2.32 osse::collaborate::PlatformOrbit Class Reference

Propogates the position of a satellite.

`#include <platform_orbit.h>`

Inheritance diagram for osse::collaborate::PlatformOrbit:



Public Types

- typedef std::array< std::string, kNumElements > TwoLineElementSet

  A two-line element set.

Public Member Functions

- PlatformOrbit (const TwoLineElementSet &_tle)

  Constructor from TLE.
- OrbitalState PredictOrbitalState (const SimulationClock &_simulation_clock, const uint64_t &_time_s) const

Get the orbital_state of an object the current clock time.

- void PredictOrbitalState (const SimulationClock &_simulation_clock, const uint64_t &_↩ time_s, OrbitalState *_orbital_state) const

  Update the orbital_state of an object the current clock time.
- std::vector< PlatformOrbit > Duplicate (const uint16_t &_orbit_planes, const uint16_t &↩ _groups_per_plane, const uint16_t &_sats_in_train, const uint16_t &_sats_in_tandem, const uint16_t &_train_angle, const uint16_t &_tandem_angle) const

  Makes a pattern of orbits based on this.
- const TwoLineElementSet & kTle () const

  Get Array of TLE strings.

Static Public Attributes

- static constexpr int kNumElements = 3

  Number of strings in a two-line element set.

Private Attributes

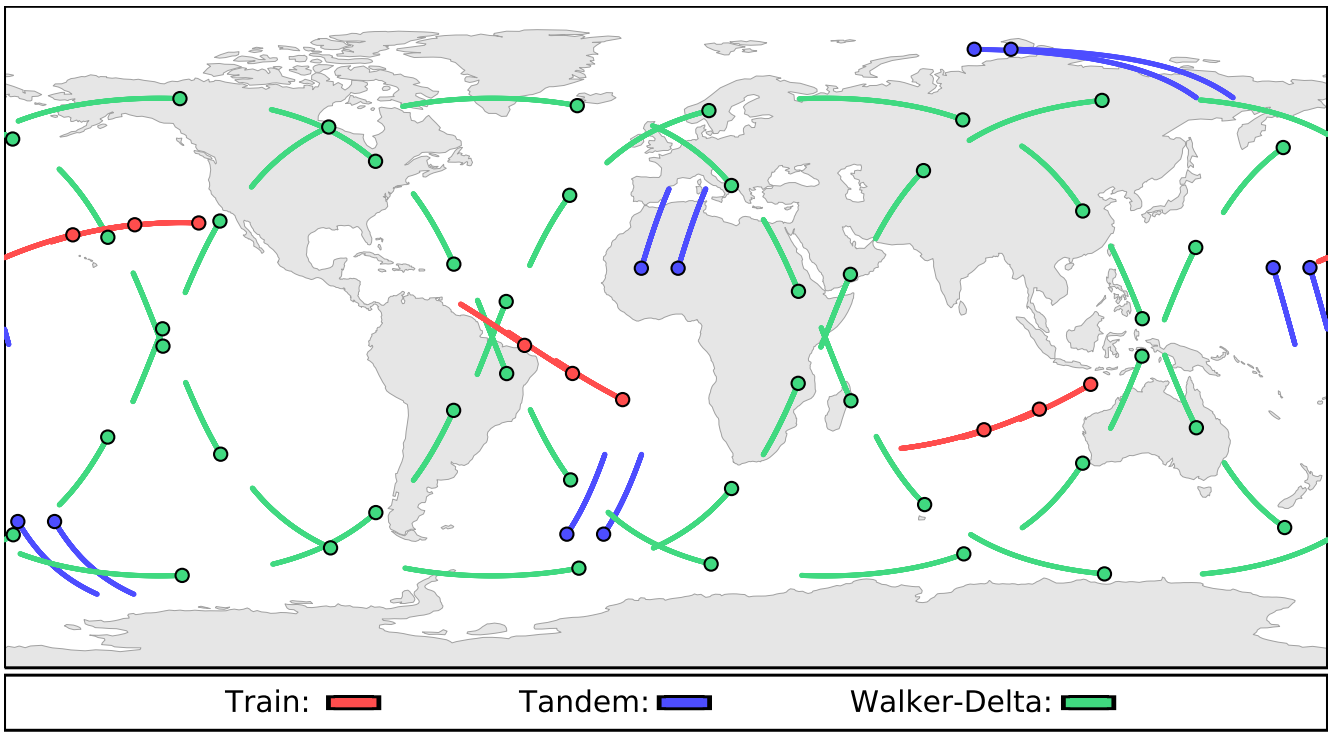
- const TwoLineElementSet kTle_

  Array of TLE strings.
- const sgp4::SGP4 kModel_

  SGP4 orbital model.

2.32.1 Detailed Description

Propogates the position of a satellite.



2.32.2 Constructor & Destructor Documentation

2.32.2.1 PlatformOrbit()

```
osse::collaborate::PlatformOrbit::PlatformOrbit (
            const TwoLineElementSet & _tle ) [explicit]
```

Constructor from TLE.

Parameters

in	*_tle*	Two-line element set

2.32.3 Member Function Documentation

2.32.3.1 Duplicate()

```
std::vector< PlatformOrbit > osse::collaborate::PlatformOrbit::Duplicate (
            const uint16_t & _orbit_planes,
            const uint16_t & _groups_per_plane,
            const uint16_t & _sats_in_train,
            const uint16_t & _sats_in_tandem,
            const uint16_t & _train_angle,
            const uint16_t & _tandem_angle ) const
```

Makes a pattern of orbits based on this.

Parameters

in	*_orbit_planes*	Number of orbit planes
in	*_groups_per_plane*	Number of groups per plane
in	*_sats_in_train*	Number of satellites in a train
in	*_sats_in_tandem*	Number of satellites in tandem
in	*_train_angle*	Angle between train satellites
in	*_tandem_angle*	Angle between tandem satellites

Returns

pattern_ A list of orbits

2.32.3.2 kTle()

```
const TwoLineElementSet& osse::collaborate::PlatformOrbit::kTle ( ) const  [inline]
```

Get Array of TLE strings.

Returns

kTle_ Array of TLE strings

2.32.3.3 PredictOrbitalState() [1/2]

```
OrbitalState osse::collaborate::PlatformOrbit::PredictOrbitalState (
            const SimulationClock & _simulation_clock,
            const uint64_t & _time_s ) const  [virtual]
```

Get the orbital_state of an object the current clock time.

Parameters

in	*_simulation_clock*	Simulation clock
in	*_time_s*	Time offset from the current time (seconds)

Returns

OrbitalState of an object the current clock time

Implements osse::collaborate::Platform.

2.32.3.4 PredictOrbitalState() [2/2]

```
void osse::collaborate::PlatformOrbit::PredictOrbitalState (
            const SimulationClock & _simulation_clock,
            const uint64_t & _time_s,
            OrbitalState * _orbital_state ) const  [virtual]
```

Update the orbital_state of an object the current clock time.

Parameters

in	*_simulation_clock*	Simulation clock
in	*_time_s*	Time offset from the current time (seconds)
in	*_orbital_state*	OrbitalState

Implements osse::collaborate::Platform.

The documentation for this class was generated from the following files:

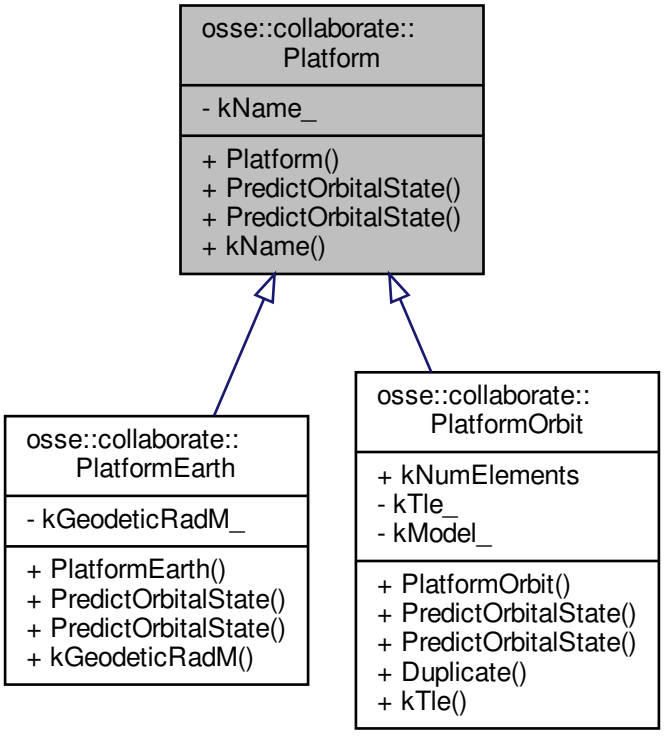
- libs/collaborate/include/collaborate/platform_orbit.h
- libs/collaborate/src/platform_orbit.cpp

2.33 osse::collaborate::Platform Class Reference

Propogates a node's position.

`#include <platform.h>`

Inheritance diagram for osse::collaborate::Platform:



Public Member Functions

- Platform (const std::string &_name)

  Constructor.
- virtual OrbitalState PredictOrbitalState (const SimulationClock &_clock, const uint64_t &↩ _time_s) const =0

  Get the orbital_state of an object the current clock time.
- virtual void PredictOrbitalState (const SimulationClock &_clock, const uint64_t &_time_s, OrbitalState *_orbital_state) const =0

  Update the orbital_state of an object the current clock time.
- std::string kName () const

  Gets kName.

Private Attributes

- const std::string kName_

  Name of the platform.

2.33.1 Detailed Description

Propogates a node's position.

2.33.2 Constructor & Destructor Documentation

2.33.2.1 Platform()

```
osse::collaborate::Platform::Platform (
            const std::string & _name ) [explicit]
```

Constructor.

Parameters

in	*_name*	The name of the platform

2.33.3 Member Function Documentation

2.33.3.1 kName()

```
std::string osse::collaborate::Platform::kName ( ) const  [inline]
```

Gets kName.

Returns

kName_ Name of the platform

2.33.3.2 PredictOrbitalState() [1/2]

```
virtual OrbitalState osse::collaborate::Platform::PredictOrbitalState (
            const SimulationClock & _clock,
            const uint64_t & _time_s ) const  [pure virtual]
```

Get the orbital_state of an object the current clock time.

Parameters

in	*_clock*	Simulation clock
in	*_time_s*	Time offset from the current time (seconds)

Returns

OrbitalState of an object at the current clock time

Implemented in osse::collaborate::PlatformOrbit, and osse::collaborate::PlatformEarth.

2.33.3.3 PredictOrbitalState() [2/2]

```
virtual void osse::collaborate::Platform::PredictOrbitalState (
            const SimulationClock & _clock,
            const uint64_t & _time_s,
            OrbitalState * _orbital_state ) const  [pure virtual]
```

Update the orbital_state of an object the current clock time.

Parameters

in	*_clock*	Simulation clock
in	*_time_s*	Time offset from the current time (seconds)
in	*_orbital_state*	OrbitalState of an object

Implemented in osse::collaborate::PlatformOrbit, and osse::collaborate::PlatformEarth.

The documentation for this class was generated from the following files:

- libs/collaborate/include/collaborate/platform.h
- libs/collaborate/src/platform.cpp

2.34 osse::collaborate::ReferenceFrame Class Reference

An attitude reference frame.

`#include <reference_frame.h>`

Public Member Functions

- ReferenceFrame (const Vector &_x_axis, const Vector &_y_axis, const Vector &_z_axis)

  Constructor from axes.
- ReferenceFrame (const double &_roll_rad, const double &_pitch_rad, const double &_yaw_rad)

  Constructor from angles.
- ReferenceFrame (const ReferenceFrame &_frame, const double &_roll_rad, const double &_pitch_rad, const double &_yaw_rad)

  Constructor from reference frame and rotation angles.
- ReferenceFrame (const ReferenceFrame &_frame_1, const ReferenceFrame &_frame_2, const double &_roll_rad, const double &_pitch_rad, const double &_yaw_rad)

  Constructor from two reference frames and rotation angles.
- void Update (const ReferenceFrame &_frame)

  Updates axes relative to a single reference frame.
- void Update (const ReferenceFrame &_frame_1, const ReferenceFrame &_frame_2)

  Updates axes relative to a single reference frame.
- std::vector< double > ObtainLog () const

  Logs the ReferenceFrame.
- void set_z_axis (const Vector &_z_axis)

  Set z-axis.
- void set_y_axis (const Vector &_y_axis)

Set y-axis.

- void set_x_axis (const Vector &_x_axis)

  Set x-axis.
- const AttitudeMatrix & attitude () const

  Get attitude attitude_matrix.
- const Vector & z_axis () const

  Get z-axis.
- const Vector & y_axis () const

  Get y-axis.
- const Vector & x_axis () const

  Get x-axis.
- std::string ToString () const

  Outputs ReferenceFrame to a string.

Private Member Functions

- Vector TransformXAxis (const ReferenceFrame &_frame) const

  Transforms Earth's x-axis through single reference.
- Vector TransformYAxis (const ReferenceFrame &_frame) const

  Transforms Earth's y-axis through single reference.
- Vector TransformZAxis (const ReferenceFrame &_frame) const

  Transforms Earth's z-axis through single reference.
- Vector TransformXAxis (const ReferenceFrame &_frame_1, const ReferenceFrame &_frame_2) const

  Transforms Earth's x-axis through two reference frames.
- Vector TransformYAxis (const ReferenceFrame &_frame_1, const ReferenceFrame &_frame_2) const

  Transforms Earth's y-axis through two reference frames.
- Vector TransformZAxis (const ReferenceFrame &_frame_1, const ReferenceFrame &_frame_2) const

  Transforms Earth's z-axis through two reference frames.

Private Attributes

- AttitudeMatrix attitude_

  Attitude attitude_matrix.
- Vector x_axis_

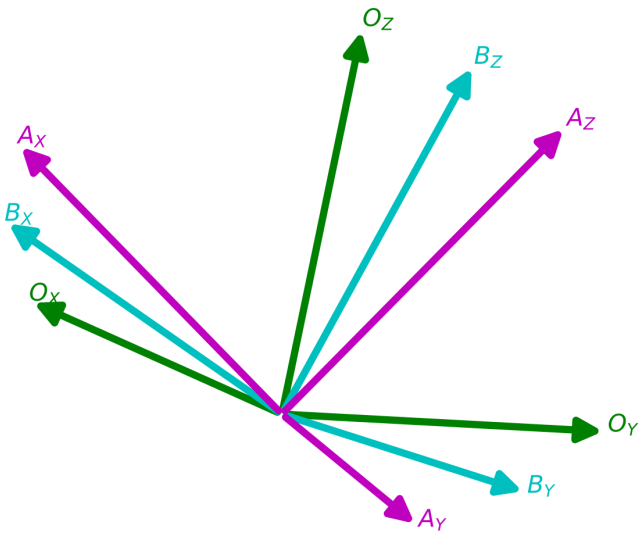
  X-axis.
- Vector y_axis_

  Y-axis.
- Vector z_axis_

  Z-axis.

2.34.1 Detailed Description

An attitude reference frame.



$$\vec{x}_{axis}, \vec{y}_{axis}, \vec{z}_{axis} = \begin{bmatrix} x_x \\ x_y \\ x_z \end{bmatrix}, \begin{bmatrix} y_x \\ y_y \\ y_z \end{bmatrix}, \begin{bmatrix} z_x \\ z_y \\ z_z \end{bmatrix} \quad (unit\ vectors)$$

$$\mathbf{A} = Attitude\ AttitudeMatrix = \begin{bmatrix} a_{0,0} & a_{0,1} & a_{0,2} \\ a_{1,0} & a_{1,1} & a_{1,2} \\ a_{2,0} & a_{2,1} & a_{2,2} \end{bmatrix}$$

2.34.2 Constructor & Destructor Documentation

2.34.2.1 ReferenceFrame() [1/4]

```
osse::collaborate::ReferenceFrame::ReferenceFrame (
            const Vector & _x_axis,
            const Vector & _y_axis,
            const Vector & _z_axis )
```

Constructor from axes.

Parameters

in	*_x_axis*	X-axis (unit)
in	*_y_axis*	Y-axis (unit)
in	*_z_axis*	Z-axis (unit)

2.34.2.2 ReferenceFrame() [2/4]

```
osse::collaborate::ReferenceFrame::ReferenceFrame (
            const double & _roll_rad,
            const double & _pitch_rad,
            const double & _yaw_rad )
```

Constructor from angles.

Parameters

in	*_roll_rad*	Rotation about the x-axis
in	*_pitch_rad*	Rotation about the y-axis
in	*_yaw_rad*	Rotation about the z-axis

2.34.2.3 ReferenceFrame() [3/4]

```
osse::collaborate::ReferenceFrame::ReferenceFrame (
            const ReferenceFrame & _frame,
            const double & _roll_rad,
            const double & _pitch_rad,
            const double & _yaw_rad )
```

Constructor from reference frame and rotation angles.

Parameters

in	*_frame*	Reference frame
in	*_roll_rad*	Rotation about the x-axis
in	*_pitch_rad*	Rotation about the y-axis
in	*_yaw_rad*	Rotation about the z-axis

2.34.2.4 ReferenceFrame() [4/4]

```
osse::collaborate::ReferenceFrame::ReferenceFrame (
            const ReferenceFrame & _frame_1,
            const ReferenceFrame & _frame_2,
            const double & _roll_rad,
            const double & _pitch_rad,
            const double & _yaw_rad )
```

Constructor from two reference frames and rotation angles.

Parameters

in	*_frame_1*	First reference frame
in	*_frame_2*	Second reference frame
in	*_roll_rad*	Rotation about the x-axis
in	*_pitch_rad*	Rotation about the y-axis
in	*_yaw_rad*	Rotation about the z-axis

2.34.3 Member Function Documentation

2.34.3.1 attitude()

```
const AttitudeMatrix& osse::collaborate::ReferenceFrame::attitude ( ) const  [inline]
```

Get attitude attitude_matrix.

Returns

attitude_ Attitude attitude_matrix

2.34.3.2 ObtainLog()

```
std::vector< double > osse::collaborate::ReferenceFrame::ObtainLog ( ) const
```

Logs the ReferenceFrame.

Returns

A vector of doubles

2.34.3.3 set_x_axis()

```
void osse::collaborate::ReferenceFrame::set_x_axis (
            const Vector & _x_axis )  [inline]
```

Set x-axis.

Parameters

in	*_x_axis*	X-axis

2.34.3.4 set_y_axis()

```
void osse::collaborate::ReferenceFrame::set_y_axis (
            const Vector & _y_axis ) [inline]
```

Set y-axis.

Parameters

in	*_y_axis*	Y-axis

2.34.3.5 set_z_axis()

```
void osse::collaborate::ReferenceFrame::set_z_axis (
            const Vector & _z_axis ) [inline]
```

Set z-axis.

Parameters

in	*_z_axis*	Z-axis

2.34.3.6 ToString()

```
std::string osse::collaborate::ReferenceFrame::ToString ( ) const
```

Outputs ReferenceFrame to a string.

Returns

ReferenceFrame as a string

2.34.3.7 TransformXAxis() [1/2]

```
Vector osse::collaborate::ReferenceFrame::TransformXAxis (
            const ReferenceFrame & _frame ) const  [private]
```

Transforms Earth's x-axis through single reference.

Parameters

in	*_frame*	Reference frame

Returns

Earth's transformed x-axis

$$\mathbf{M}_{this} = This\ Attitude\ AttitudeMatrix$$
$$\mathbf{M}_{F} = Reference\ ReferenceFrame\ Attitude\ AttitudeMatrix$$
$$\vec{x}_{earth} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$
$$\vec{x} = ((\vec{x}_{earth}\mathbf{M}_{this})\,\mathbf{M}_{F})$$

2.34.3.8 TransformXAxis() [2/2]

```
Vector osse::collaborate::ReferenceFrame::TransformXAxis (
            const ReferenceFrame & _frame_1,
            const ReferenceFrame & _frame_2 ) const  [private]
```

Transforms Earth's x-axis through two reference frames.

Parameters

in	*_frame_1*	First reference frame
in	*_frame_2*	Second reference frame

Returns

Earth's transformed x-axis

$$\mathbf{M}_{this} = This\ Attitude\ AttitudeMatrix$$

$$\mathbf{M}_{F1} = Reference\ ReferenceFrame\ 1\ Attitude\ AttitudeMatrix$$
$$\mathbf{M}_{F2} = Reference\ ReferenceFrame\ 2\ Attitude\ AttitudeMatrix$$
$$\vec{x}_{earth} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$
$$\vec{x} = (((\vec{x}_{earth}\mathbf{M}_{this})\,\mathbf{M}_{F2})\,\mathbf{M}_{F1})$$

2.34.3.9 TransformYAxis() [1/2]

```
Vector osse::collaborate::ReferenceFrame::TransformYAxis (
            const ReferenceFrame & _frame ) const  [private]
```

Transforms Earth's y-axis through single reference.

Parameters

in	*_frame*	Reference frame

Returns

Earth's transformed y-axis

$$\mathbf{M}_{this} = This\ Attitude\ AttitudeMatrix$$
$$\mathbf{M}_{F} = Reference\ ReferenceFrame\ Attitude\ AttitudeMatrix$$
$$\vec{y}_{earth} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$
$$\vec{y} = ((\vec{y}_{earth}\mathbf{M}_{this})\,\mathbf{M}_{F})$$

2.34.3.10 TransformYAxis() [2/2]

```
Vector osse::collaborate::ReferenceFrame::TransformYAxis (
            const ReferenceFrame & _frame_1,
            const ReferenceFrame & _frame_2 ) const  [private]
```

Transforms Earth's y-axis through two reference frames.

Parameters

in	*_frame_1*	First reference frame
in	*_frame_2*	Second reference frame

Returns

Earth's transformed y-axis

$$\mathbf{M}_{this} = This\ Attitude\ AttitudeMatrix$$
$$\mathbf{M}_{F1} = Reference\ ReferenceFrame\ 1\ Attitude\ AttitudeMatrix$$
$$\mathbf{M}_{F2} = Reference\ ReferenceFrame\ 2\ Attitude\ AttitudeMatrix$$
$$\vec{y}_{earth} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$
$$\vec{y} = (((\vec{y}_{earth}\mathbf{M}_{this})\,\mathbf{M}_{F2})\,\mathbf{M}_{F1})$$

2.34.3.11 TransformZAxis() [1/2]

```
Vector osse::collaborate::ReferenceFrame::TransformZAxis (
            const ReferenceFrame & _frame ) const  [private]
```

Transforms Earth's z-axis through single reference.

Parameters

in	*_frame*	Reference frame

Returns

Earth's transformed z-axis

$$\mathbf{M}_{this} = This\ Attitude\ AttitudeMatrix$$
$$\mathbf{M}_{F} = Reference\ ReferenceFrame\ Attitude\ AttitudeMatrix$$
$$\vec{z}_{earth} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$
$$\vec{z} = ((\vec{z}_{earth}\mathbf{M}_{this})\,\mathbf{M}_{F})$$

2.34.3.12 TransformZAxis() [2/2]

```
Vector osse::collaborate::ReferenceFrame::TransformZAxis (
            const ReferenceFrame & _frame_1,
            const ReferenceFrame & _frame_2 ) const  [private]
```

Transforms Earth's z-axis through two reference frames.

Parameters

in	*_frame_1*	First reference frame
in	*_frame_2*	Second reference frame

Returns

Earth's transformed z-axis

$$\mathbf{M}_{this} = This\ Attitude\ AttitudeMatrix$$
$$\mathbf{M}_{F1} = Reference\ ReferenceFrame\ 1\ Attitude\ AttitudeMatrix$$
$$\mathbf{M}_{F2} = Reference\ ReferenceFrame\ 2\ Attitude\ AttitudeMatrix$$
$$\vec{z}_{earth} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$
$$\vec{z} = (((\vec{z}_{earth}\mathbf{M}_{this})\,\mathbf{M}_{F2})\,\mathbf{M}_{F1})$$

2.34.3.13 Update() [1/2]

```
void osse::collaborate::ReferenceFrame::Update (
            const ReferenceFrame & _frame )
```

Updates axes relative to a single reference frame.

Parameters

in	*_frame*	Reference frame

$$\mathbf{M}_{this} = This\ Attitude\ AttitudeMatrix$$
$$\mathbf{M}_{F} = Reference\ ReferenceFrame\ Attitude\ AttitudeMatrix$$
$$\vec{x}_{earth} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad \vec{y}_{earth} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \quad \vec{z}_{earth} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$
$$\vec{x} = (\vec{x}_{earth}\mathbf{M}_{this})\,\mathbf{M}_{F})$$
$$\vec{y} = (\vec{y}_{earth}\mathbf{M}_{this})\,\mathbf{M}_{F})$$
$$\vec{y} = (\vec{y}_{earth}\mathbf{M}_{this})\,\mathbf{M}_{F})$$

2.34.3.14 Update() [2/2]

```
void osse::collaborate::ReferenceFrame::Update (
            const ReferenceFrame & _frame_1,
            const ReferenceFrame & _frame_2 )
```

Updates axes relative to a single reference frame.

Parameters

in	*_frame_1*	First reference frame
in	*_frame_2*	Second reference frame

$$\mathbf{M}_{this} = This\ Attitude\ AttitudeMatrix$$
$$\mathbf{M}_{F1} = Reference\ ReferenceFrame\ 1\ Attitude\ AttitudeMatrix$$
$$\mathbf{M}_{F2} = Reference\ ReferenceFrame\ 2\ Attitude\ AttitudeMatrix$$
$$\vec{x}_{earth} = \begin{bmatrix}1\\0\\0\end{bmatrix}, \quad \vec{y}_{earth} = \begin{bmatrix}0\\1\\0\end{bmatrix}, \quad \vec{z}_{earth} = \begin{bmatrix}0\\0\\1\end{bmatrix}$$
$$\vec{x} = (((\vec{x}_{earth}\mathbf{M}_{this})\,\mathbf{M}_{F2})\,\mathbf{M}_{F1})$$
$$\vec{y} = (((\vec{y}_{earth}\mathbf{M}_{this})\,\mathbf{M}_{F2})\,\mathbf{M}_{F1})$$
$$\vec{z} = (((\vec{z}_{earth}\mathbf{M}_{this})\,\mathbf{M}_{F2})\,\mathbf{M}_{F1})$$

2.34.3.15 x_axis()

```
const Vector& osse::collaborate::ReferenceFrame::x_axis ( ) const  [inline]
```

Get x-axis.

Returns

x_axis_ X-axis

2.34.3.16 y_axis()

```
const Vector& osse::collaborate::ReferenceFrame::y_axis ( ) const  [inline]
```

Get y-axis.

Returns

y_axis_ Y-axis

2.34.3.17 z_axis()

```
const Vector& osse::collaborate::ReferenceFrame::z_axis ( ) const  [inline]
```

Get z-axis.

Returns

z_axis_ Z-axis

The documentation for this class was generated from the following files:

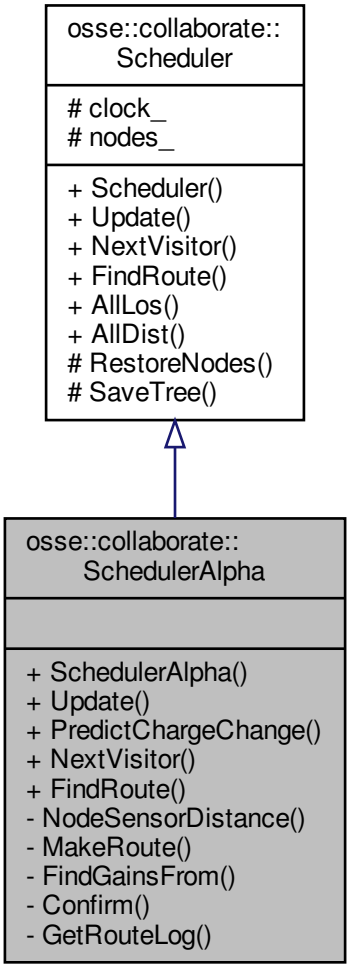
- libs/collaborate/include/collaborate/reference_frame.h
- libs/collaborate/src/reference_frame.cpp

2.35 osse::collaborate::SchedulerAlpha Class Reference

Concrete internal simulation processor.

```
#include <scheduler_alpha.h>
```

Inheritance diagram for osse::collaborate::SchedulerAlpha:



Public Member Functions

- SchedulerAlpha (SimulationClock *_clock)

  Constructor.
- void Update (const std::vector< Node *> &_nodes, EventLogger *_event_log)

  Updates the list of nodes.
- void PredictChargeChange (Sun *_sun, Node *_node, const uint64_t &_limit_s)

  Predicts the times at which charge status will change.
- bool NextVisitor (const std::vector< Geodetic > &_destinations_rad_m, const uint16_t &_↩ sink_constellation, Node **next_visitor_, uint64_t *prediction_s_)

  Find the next node to visit a location, at future time (seconds)

- std::vector< std::pair< uint16_t, uint64_t > > FindRoute (const uint16_t &_start, const uint16_t &_end, const uint64_t &_contact_s, const uint64_t &_limit_s)

  Constructs the most efficient time-dynamic route available.

Private Member Functions

- double NodeSensorDistance (Node *_node, const Geodetic &_destination_rad_m, const uint64_t &_offset_s)

  Determines the distance from a sensor's current reading position.
- std::vector< std::pair< uint16_t, uint64_t > > MakeRoute (Tree *_tree, const uint16_t &_end, const uint64_t &_contact_s)

  Makes the route from the tree and adds it to the list.
- std::vector< uint16_t > FindGainsFrom (const uint16_t &_tx_index, const uint64_t &_offset_s, const std::vector< uint16_t > &_rxs)

  Finds the current open channels with neighbors.
- uint64_t Confirm (Node *_tx_node, Node *_rx_node, const uint64_t &_duration_s, const uint64_t &_original_s, const uint64_t &_lower_limit_s)

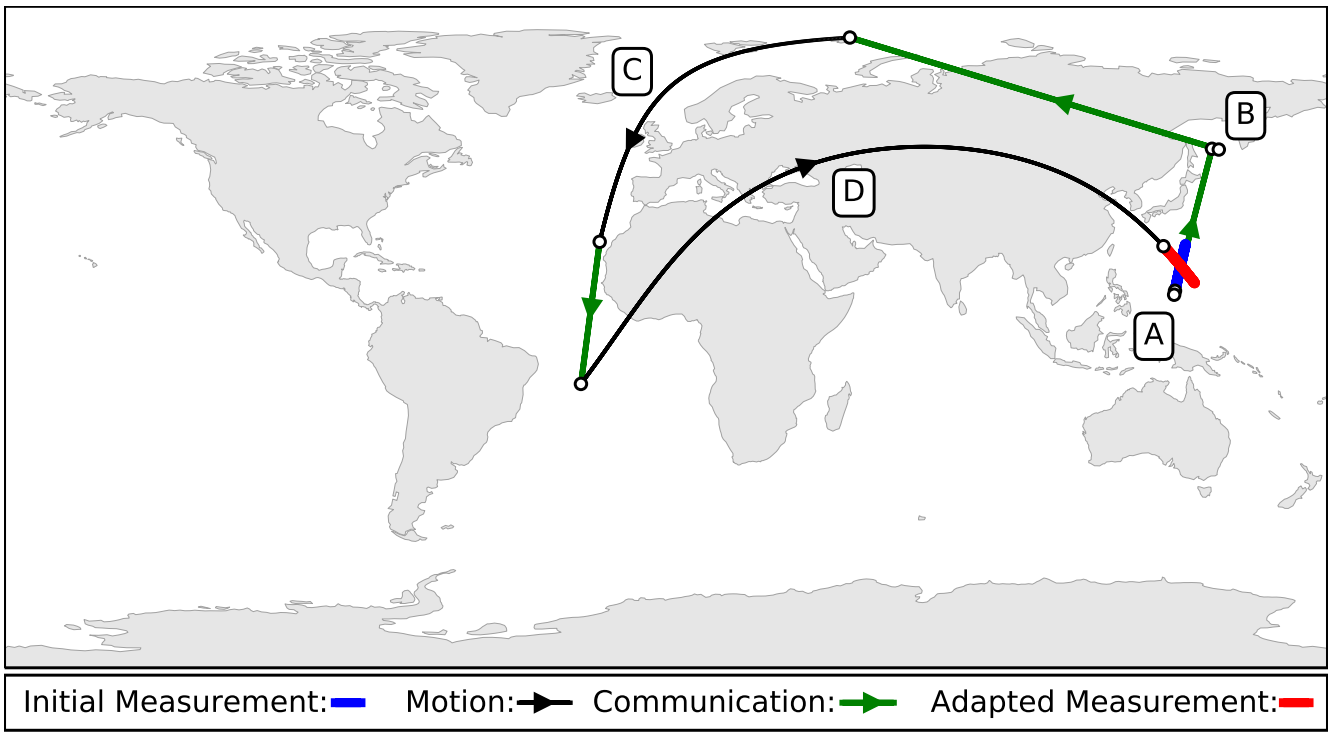
  Confirms that the contact lasts as long as the specified duration.
- std::string GetRouteLog (const uint16_t &_start_index, const std::vector< std::pair< uint16_t, uint64_t >> &_route)

  Obtain the string log for the route.

Additional Inherited Members

2.35.1 Detailed Description

Concrete internal simulation processor.



2.35.2 Constructor & Destructor Documentation

2.35.2.1 SchedulerAlpha()

```
osse::collaborate::SchedulerAlpha::SchedulerAlpha (
            SimulationClock * _clock ) [explicit]
```

Constructor.

Parameters

in	*_clock*	Simulation clock

2.35.3 Member Function Documentation

2.35.3.1 Confirm()

```
uint64_t osse::collaborate::SchedulerAlpha::Confirm (
            Node * _tx_node,
            Node * _rx_node,
            const uint64_t & _duration_s,
            const uint64_t & _original_s,
            const uint64_t & _lower_limit_s ) [private]
```

Confirms that the contact lasts as long as the specified duration.

Parameters

in	*_tx_node*	The transmitter
in	*_rx_node*	The receiver
in	*_duration_s*	The duration of contact (seconds)
in	*_original_s*	The starting offset (seconds)
in	*_lower_↩ limit_s*	The lower limit (seconds)

Returns

The starting time of contact, or infinity if none found

2.35.3.2 FindGainsFrom()

```
std::vector< uint16_t > osse::collaborate::SchedulerAlpha::FindGainsFrom (
            const uint16_t & _tx_index,
            const uint64_t & _offset_s,
            const std::vector< uint16_t > & _rxs ) [private]
```

Finds the current open channels with neighbors.

Parameters

in	*_tx_index*	The index of the transmitter
in	*_offset_s*	The time offset from current time (seconds)
in	*_rxs*	The receivers to consider

Returns

The possible receivers

2.35.3.3 FindRoute()

```
std::vector< std::pair< uint16_t, uint64_t > > osse::collaborate::SchedulerAlpha::FindRoute (
            const uint16_t & _start,
            const uint16_t & _end,
            const uint64_t & _contact_s,
            const uint64_t & _limit_s ) [virtual]
```

Constructs the most efficient time-dynamic route available.

Parameters

in	*_start*	Start node index
in	*_end*	End node index
in	*_contact_s*	Duration of contact (seconds)
in	*_limit_s*	Expiration time limit (seconds)

Returns

Most efficient time-dynamic route available

Implements osse::collaborate::Scheduler.

2.35.3.4 GetRouteLog()

```
std::string osse::collaborate::SchedulerAlpha::GetRouteLog (
            const uint16_t & _start_index,
            const std::vector< std::pair< uint16_t, uint64_t >> & _route ) [private]
```

Obtain the string log for the route.

Parameters

in	*_start_index*	Starting node index
in	*_route*	List of individual transfers

Returns

String log for the route

2.35.3.5 MakeRoute()

```
std::vector< std::pair< uint16_t, uint64_t > > osse::collaborate::SchedulerAlpha::MakeRoute (
            Tree * _tree,
            const uint16_t & _end,
            const uint64_t & _contact_s ) [private]
```

Makes the route from the tree and adds it to the list.

Parameters

in	*_tree*	Tree
in	*_end*	End node index
in	*_contact_s*	Duration of contact (seconds)

Returns

Route

2.35.3.6 NextVisitor()

```
bool osse::collaborate::SchedulerAlpha::NextVisitor (
            const std::vector< Geodetic > & _destinations_rad_m,
            const uint16_t & _sink_constellation,
            Node ** next_visitor_,
            uint64_t * prediction_s_ )  [virtual]
```

Find the next node to visit a location, at future time (seconds)

Parameters

in	*_destinations_rad_m*	Destinations
in	*_sink_constellation*	Number of the sink constellation
out	*next_visitor_*	Next visitor
out	*prediction_s_*	Time of the next visit (seconds)

Returns

Whether a visitor has been found (if outputs are valid)

Implements osse::collaborate::Scheduler.

2.35.3.7 NodeSensorDistance()

```
double osse::collaborate::SchedulerAlpha::NodeSensorDistance (
            Node * _node,
            const Geodetic & _destination_rad_m,
            const uint64_t & _offset_s )  [private]
```

Determines the distance from a sensor's current reading position.

Parameters

in	*_node*	Node
in	*_destination_rad_m*	Geodetic destination
in	*_offset_s*	Time offset from present (seconds)

Returns

Distance from a sensor's current reading position

2.35.3.8 PredictChargeChange()

```
void osse::collaborate::SchedulerAlpha::PredictChargeChange (
            Sun * _sun,
            Node * _node,
            const uint64_t & _limit_s )
```

Predicts the times at which charge status will change.

Parameters

in	*_sun*	Sun
in	*_node*	Subject node
in	*_limit_s*	Time limit (seconds)

2.35.3.9 Update()

```
void osse::collaborate::SchedulerAlpha::Update (
            const std::vector< Node *> & _nodes,
            EventLogger * _event_log ) [virtual]
```

Updates the list of nodes.

Parameters

in	*_nodes*	Nodes
in	*_event_log*	Event logger

Implements osse::collaborate::Scheduler.

The documentation for this class was generated from the following files:

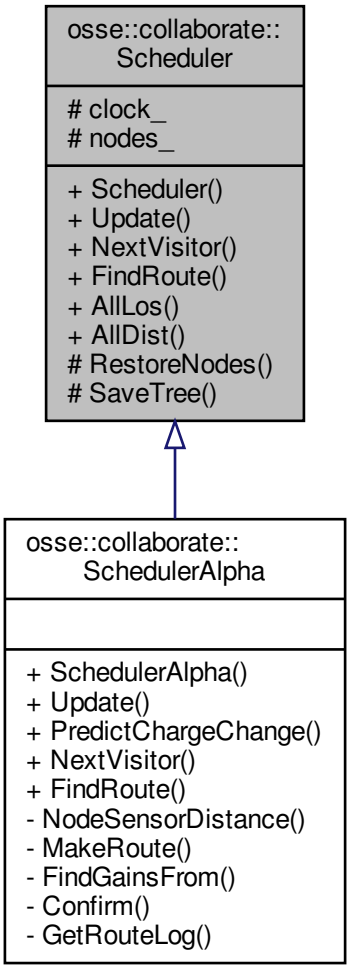
- libs/collaborate/include/collaborate/scheduler_alpha.h
- libs/collaborate/src/scheduler_alpha.cpp

2.36 osse::collaborate::Scheduler Class Reference

Abstract internal simulation processor.

`#include <scheduler.h>`

Inheritance diagram for osse::collaborate::Scheduler:



Public Member Functions

- Scheduler (SimulationClock *_clock)

  Constructor.
- virtual void Update (const std::vector< Node *> &_nodes, EventLogger *_event_log)=0

  Runs the scheduling algorithm.
- virtual bool NextVisitor (const std::vector< Geodetic > &_destinations_rad_m, const uint16_t &_sink_constellation, Node **next_visitor_, uint64_t *prediction_s_)=0

  Find the next node to visit a location, at future time (seconds)

- virtual std::vector< std::pair< uint16_t, uint64_t > > FindRoute (const uint16_t &_start, const uint16_t &_end, const uint64_t &_contact_s, const uint64_t &_limit_s)=0

  Constructs the most efficient time-dynamic route available.
- void AllLos (GraphUnweighted *_unweighted)

  Fills the unweighted graph with LOS connections.
- void AllDist (GraphWeighted *_weighted)

  Fills the weighted graph with distances.

Protected Member Functions

- void RestoreNodes ()

  Restores all nodes to original orbital_state.
- void SaveTree (const uint16_t &_start, const uint16_t &_end, const Tree &_tree)

  Saves a tree to a text file.

Protected Attributes

- SimulationClock * clock_

  Simulation clock.
- std::vector< Node * > nodes_

  Current nodes from sensor network.

2.36.1 Detailed Description

Abstract internal simulation processor.

2.36.2 Constructor & Destructor Documentation

2.36.2.1 Scheduler()

```
osse::collaborate::Scheduler::Scheduler (
            SimulationClock * _clock ) [explicit]
```

Constructor.

Parameters

in	*_clock*	Simulation clock

2.36.3 Member Function Documentation

2.36.3.1 AllDist()

```
void osse::collaborate::Scheduler::AllDist (
            GraphWeighted * _weighted )
```

Fills the weighted graph with distances.

Parameters

in	*_weighted*	Weighted graph

2.36.3.2 AllLos()

```
void osse::collaborate::Scheduler::AllLos (
            GraphUnweighted * _unweighted )
```

Fills the unweighted graph with LOS connections.

Parameters

in	*_unweighted*	Unweighted graph

2.36.3.3 FindRoute()

```
virtual std::vector<std::pair<uint16_t, uint64_t> > osse::collaborate::Scheduler::FindRoute (
            const uint16_t & _start,
            const uint16_t & _end,
            const uint64_t & _contact_s,
            const uint64_t & _limit_s )  [pure virtual]
```

Constructs the most efficient time-dynamic route available.

Parameters

in	*_start*	Start node index
in	*_end*	End node index

Parameters

in	_↩ *contact*↩ _s	Duration of contact (seconds)
in	*_limit_s*	Expiration time limit (seconds)

Returns

Most efficient time-dynamic route available

Implemented in osse::collaborate::SchedulerAlpha.

2.36.3.4 NextVisitor()

```
virtual bool osse::collaborate::Scheduler::NextVisitor (
            const std::vector< Geodetic > & _destinations_rad_m,
            const uint16_t & _sink_constellation,
            Node ** next_visitor_,
            uint64_t * prediction_s_ )  [pure virtual]
```

Find the next node to visit a location, at future time (seconds)

Parameters

in	*_destinations_rad_m*	Destinations (meters and radians)
in	*_sink_constellation*	Which constellation to track
out	*next_visitor_*	Next visitor
out	*prediction_s_*	Time of the next visit (seconds)

Returns

Whether a visitor has been found (if outputs are valid)

Implemented in osse::collaborate::SchedulerAlpha.

2.36.3.5 SaveTree()

```
void osse::collaborate::Scheduler::SaveTree (
            const uint16_t & _start,
            const uint16_t & _end,
            const Tree & _tree ) [protected]
```

Saves a tree to a text file.

Parameters

in	*_tree*	Tree
in	*_start*	Start node index
in	*_end*	End node index

2.36.3.6 Update()

```
virtual void osse::collaborate::Scheduler::Update (
            const std::vector< Node *> & _nodes,
            EventLogger * _event_log ) [pure virtual]
```

Runs the scheduling algorithm.

Parameters

in	*_nodes*	Nodes
in	*_event_log*	Event logger

Implemented in osse::collaborate::SchedulerAlpha.

The documentation for this class was generated from the following files:

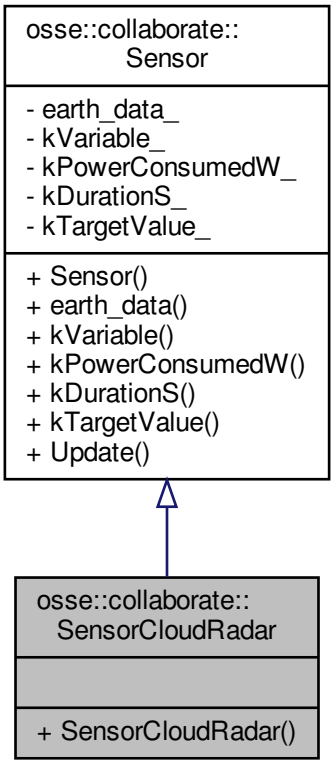
- libs/collaborate/include/collaborate/scheduler.h
- libs/collaborate/src/scheduler.cpp

2.37 osse::collaborate::SensorCloudRadar Class Reference

Measures total cloud optical depth.

```
#include <sensor_cloud_radar.h>
```

Inheritance diagram for osse::collaborate::SensorCloudRadar:



Public Member Functions

- SensorCloudRadar (const std::string &_path, const uint64_t &_duration_s)
  Constructor.

2.37.1 Detailed Description

Measures total cloud optical depth.

2.37.2 Constructor & Destructor Documentation

2.37.2.1 SensorCloudRadar()

```
osse::collaborate::SensorCloudRadar::SensorCloudRadar (
            const std::string & _path,
            const uint64_t & _duration_s )
```

Constructor.

Parameters

in	*_path*	Parent directory path
in	*_duration_s*	Duration of measurements (seconds)

The documentation for this class was generated from the following files:

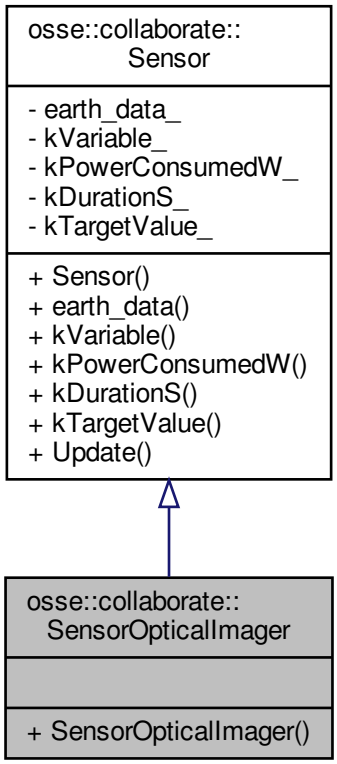
- libs/collaborate/include/collaborate/sensor_cloud_radar.h
- libs/collaborate/src/sensor_cloud_radar.cpp

2.38 osse::collaborate::SensorOpticalImager Class Reference

Obtains optical images of Earth's surface.

`#include <sensor_optical_imager.h>`

Inheritance diagram for osse::collaborate::SensorOpticalImager:



Public Member Functions

- SensorOpticalImager (const std::string &_path, const uint64_t &_duration_s)
  Constructor.

2.38.1 Detailed Description

Obtains optical images of Earth's surface.

2.38.2 Constructor & Destructor Documentation

2.38.2.1 SensorOpticalImager()

```
osse::collaborate::SensorOpticalImager::SensorOpticalImager (
            const std::string & _path,
            const uint64_t & _duration_s )
```

Constructor.

Parameters

in	*_path*	Parent directory path
in	*_duration_s*	Duration of measurements (seconds)

The documentation for this class was generated from the following files:

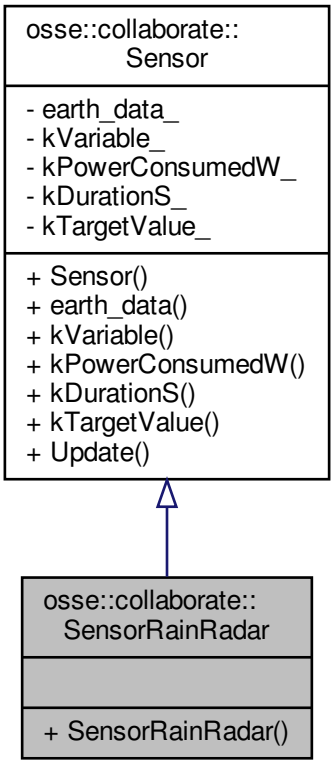
- libs/collaborate/include/collaborate/sensor_optical_imager.h
- libs/collaborate/src/sensor_optical_imager.cpp

2.39 osse::collaborate::SensorRainRadar Class Reference

Measures total precipitation.

```
#include <sensor_rain_radar.h>
```

Inheritance diagram for osse::collaborate::SensorRainRadar:



Public Member Functions

- SensorRainRadar (const std::string &_path, const uint64_t &_duration_s)
  Constructor.

2.39.1 Detailed Description

Measures total precipitation.

2.39.2 Constructor & Destructor Documentation

2.39.2.1 SensorRainRadar()

```
osse::collaborate::SensorRainRadar::SensorRainRadar (
            const std::string & _path,
            const uint64_t & _duration_s )
```

Constructor.

Parameters

in	*_path*	Parent directory path
in	*_↩ duration↩ _s*	Duration of measurements (seconds)

The documentation for this class was generated from the following files:

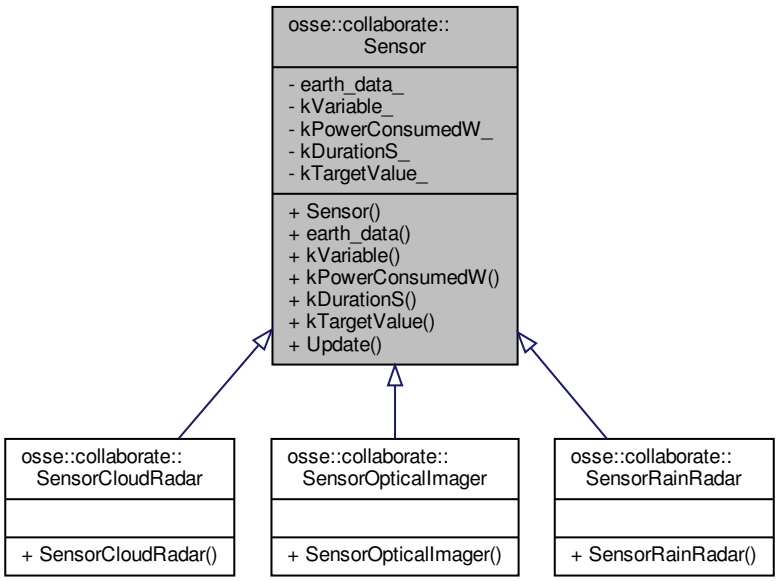
- libs/collaborate/include/collaborate/sensor_rain_radar.h
- libs/collaborate/src/sensor_rain_radar.cpp

2.40 osse::collaborate::Sensor Class Reference

Measures scientific data.

`#include <sensor.h>`

Inheritance diagram for osse::collaborate::Sensor:



Public Member Functions

- Sensor (EarthData *_earth_data, const std::string &_variable, const double &_power_↩ consumed_w, const uint64_t &_duration_s, const double &_target_value)

  Constructor from earth_data.
- EarthData * earth_data () const

  Get The earth_data of scientific data.
- const std::string & kVariable () const

  Get variable name.
- const double & kPowerConsumedW () const

  Get power consumed (Watts)
- const uint64_t & kDurationS () const

  Get duration of measurements (seconds)
- const double & kTargetValue () const

  Get target value for measurements.
- void Update (const SimulationClock &_clock) const

  Updates the earth_data.

Private Attributes

- std::unique_ptr< EarthData > earth_data_

  Data earth_data.
- const std::string kVariable_

  Variable name.
- const double kPowerConsumedW_

  Power consumed during operation (Watts)
- const uint64_t kDurationS_

  Duration of measurements (seconds)
- const double kTargetValue_

  Target value for measurements.

2.40.1 Detailed Description

Measures scientific data.

2.40.2 Constructor & Destructor Documentation

2.40.2.1 Sensor()

```
osse::collaborate::Sensor::Sensor (
            EarthData * _earth_data,
            const std::string & _variable,
            const double & _power_consumed_w,
            const uint64_t & _duration_s,
            const double & _target_value )
```

Constructor from earth_data.

Parameters

in	*_earth_data*	EarthData
in	*_variable*	Variable name
in	*_power_consumed_w*	Power consumed (Watts)
in	*_duration_s*	Duration of measurements (seconds)
in	*_target_value*	Target value for measurements

2.40.3 Member Function Documentation

2.40.3.1 earth_data()

```
EarthData* osse::collaborate::Sensor::earth_data ( ) const  [inline]
```

Get The earth_data of scientific data.

Returns

earth_data_ The earth_data of scientific data

2.40.3.2 kDurationS()

```
const uint64_t& osse::collaborate::Sensor::kDurationS ( ) const  [inline]
```

Get duration of measurements (seconds)

Returns

kDurationS_ Duration of measurements (seconds)

2.40.3.3 kPowerConsumedW()

```
const double& osse::collaborate::Sensor::kPowerConsumedW ( ) const  [inline]
```

Get power consumed (Watts)

Returns

kPowerConsumedW_ Power consumed (Watts)

2.40.3.4 kTargetValue()

```
const double& osse::collaborate::Sensor::kTargetValue ( ) const  [inline]
```

Get target value for measurements.

Returns

kTargetValue_ Target value for measurements

2.40.3.5 kVariable()

```
const std::string& osse::collaborate::Sensor::kVariable ( ) const  [inline]
```

Get variable name.

Returns

kVariable_ Variable name

2.40.3.6 Update()

```
void osse::collaborate::Sensor::Update (
            const SimulationClock & _clock ) const
```

Updates the earth_data.

Parameters

in	*_clock*	Simulation clock

The documentation for this class was generated from the following files:

- libs/collaborate/include/collaborate/sensor.h
- libs/collaborate/src/sensor.cpp

2.41 osse::collaborate::SimulationClock Class Reference

A clock for maintaining the current simulation time.

`#include <simulation_clock.h>`

Data Structures

- struct LogBuffer

  Log buffer.

Public Types

- typedef struct osse::collaborate::SimulationClock::LogBuffer LogBuffer

  Log buffer.

Public Member Functions

- SimulationClock (DataLogger *_data_log)

  Constructor.
- SimulationClock (DataLogger *_data_log, const int &_year, const int &_month, const int &_day)

  Constructor from Year, Month, and Day.
- SimulationClock (DataLogger *_data_log, const int &_year, const int &_month, const int &_day, const int &_hour, const int &_minute, const int &_second)

  Constructor from YMDHMS.
- void Tick (const uint64_t &_seconds)

  Increments time.
- std::string ToString () const

  Converts current CPU and SGP4 date_time to string.
- const sgp4::DateTime & date_time () const

Get current date and time.

- const uint64_t & last_increment_s () const

  Get previous time increment (seconds)
- const uint64_t & elapsed_s () const

  Get total time elapsed (seconds)
- const uint64_t & ticks () const

  Get total ticks.
- void Buffer ()

  Buffer clock values.
- void Flush ()

  Write buffers to a log file.

Static Public Attributes

- static constexpr int kLogBufferSize = 1000

  Size of a log buffer.

Private Attributes

- sgp4::DateTime date_time_

  Current date and time.
- uint64_t last_increment_s_

  Previous time increment (seconds)
- uint64_t elapsed_s_

  Total time elapsed (seconds)
- uint64_t ticks_

  Total ticks.
- DataLogger * data_log_

  Data logger.
- LogBuffer log_buffer_

  Log buffer.

2.41.1 Detailed Description

A clock for maintaining the current simulation time.

2.41.2 Constructor & Destructor Documentation

2.41.2.1 SimulationClock() [1/3]

```
osse::collaborate::SimulationClock::SimulationClock (
            DataLogger * _data_log ) [explicit]
```

Constructor.

Parameters

in	*_data_log*	Data logger

2.41.2.2 SimulationClock() [2/3]

```
osse::collaborate::SimulationClock::SimulationClock (
            DataLogger * _data_log,
            const int & _year,
            const int & _month,
            const int & _day )
```

Constructor from Year, Month, and Day.

Parameters

in	*_data_log*	Data logger
in	*_year*	Year
in	*_month*	Month
in	*_day*	Day

2.41.2.3 SimulationClock() [3/3]

```
osse::collaborate::SimulationClock::SimulationClock (
            DataLogger * _data_log,
            const int & _year,
            const int & _month,
            const int & _day,
            const int & _hour,
            const int & _minute,
            const int & _second )
```

Constructor from YMDHMS.

Parameters

in	*_data_log*	Data logger
in	*_year*	Year
in	*_month*	Month
in	*_day*	Day
in	*_hour*	Hour
in	*_minute*	Minute
in	*_second*	Second

2.41.3 Member Function Documentation

2.41.3.1 date_time()

```
const sgp4::DateTime& osse::collaborate::SimulationClock::date_time ( ) const  [inline]
```

Get current date and time.

Returns

date_time_ Current date and time

2.41.3.2 elapsed_s()

```
const uint64_t& osse::collaborate::SimulationClock::elapsed_s ( ) const  [inline]
```

Get total time elapsed (seconds)

Returns

elapsed_s_ Total time elapsed (seconds)

2.41.3.3 last_increment_s()

```
const uint64_t& osse::collaborate::SimulationClock::last_increment_s ( ) const  [inline]
```

Get previous time increment (seconds)

Returns

last_increment_s_ Previous time increment (seconds)

2.41.3.4 Tick()

```
void osse::collaborate::SimulationClock::Tick (
            const uint64_t & _seconds )
```

Increments time.

Parameters

in	*_seconds*	Number of seconds to increment

2.41.3.5 ticks()

```
const uint64_t& osse::collaborate::SimulationClock::ticks ( ) const  [inline]
```

Get total ticks.

Returns

ticks_ Total ticks

2.41.3.6 ToString()

```
std::string osse::collaborate::SimulationClock::ToString ( ) const
```

Converts current CPU and SGP4 date_time to string.

Returns

Current CPU and SGP4 date_time as a string

The documentation for this class was generated from the following files:

- libs/collaborate/include/collaborate/simulation_clock.h
- libs/collaborate/src/simulation_clock.cpp

2.42 osse::collaborate::SolarPanel Class Reference

A solar panel.

```
#include <solar_panel.h>
```

Public Member Functions

- SolarPanel (const double &_efficiency_percent, const double &_surface_area_m2, const double &_roll_rad, const double &_pitch_rad, const double &_yaw_rad, Sun *_sun)

  Constructor from attitude angles.
- double RxPowerW () const

  Obtain the received power in a direction (Watts)
- void Update (const ReferenceFrame &_body_frame, const ReferenceFrame &_orbit_frame, const Vector &_position_m_rad)

  Updates the attitude frame orientation and effective area.
- const double & effective_area_m2 () const

  Get Effective area (meters squared)

Static Public Attributes

- static constexpr double kSolarIrradianceWPerM2 = 1332

  The sun's irradiance (Watts per meter squared)

Private Attributes

- const double kEfficiencyPercent_

  Efficiency (percent)
- const double kSurfaceAreaM2_

  Surface area (meters squared)
- const double kRollRad_

  Roll angle to host body frame (radians)
- const double kPitchRad_

  Pitch angle to host body frame (radians)
- const double kYawRad_

  Yaw angle to host body frame (radians)
- const Sun * kSun_

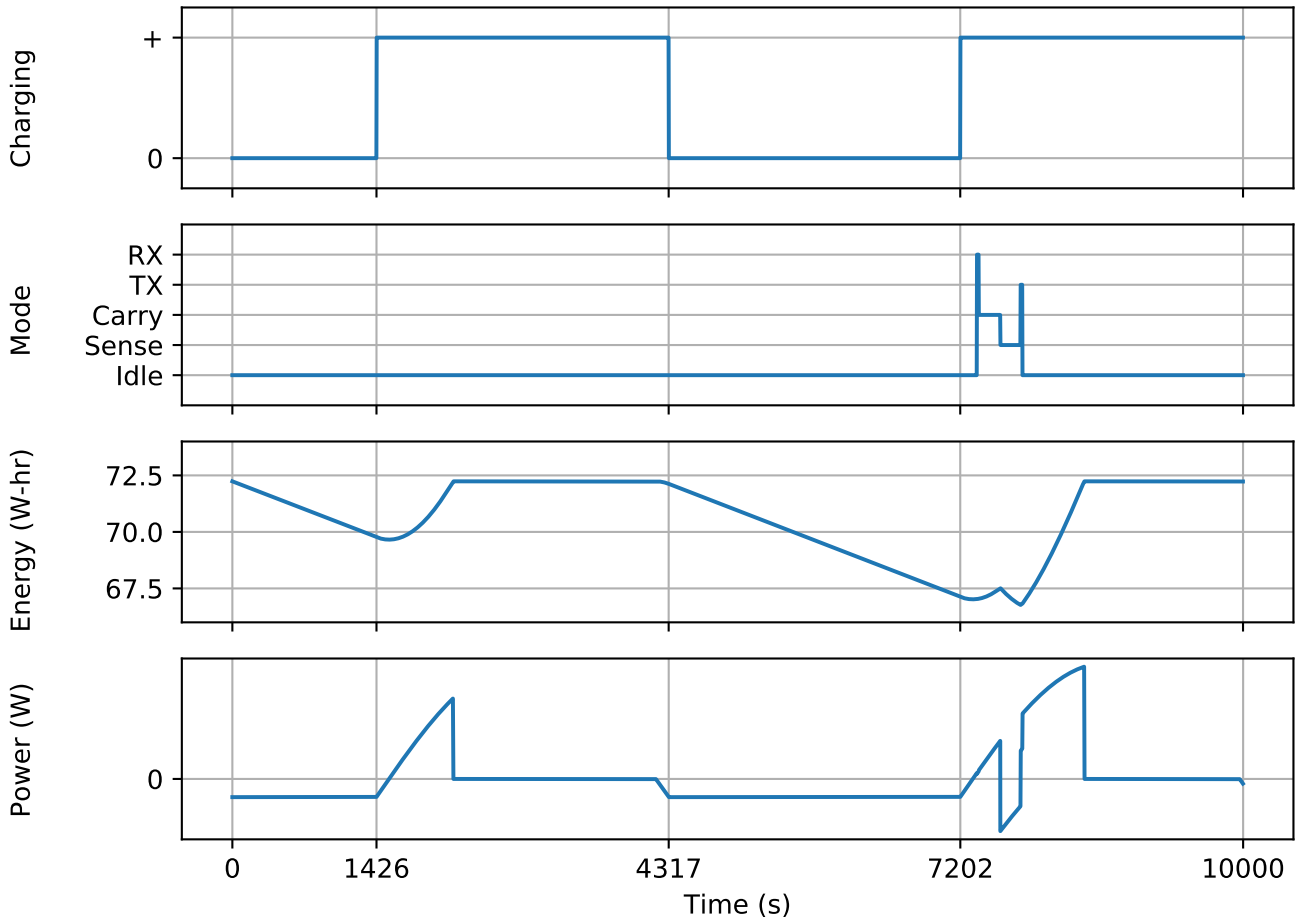
  Sun.
- double effective_area_m2_

  Effective area (meters squared)
- ReferenceFrame attitude_

  Attitude frame.

2.42.1 Detailed Description

A solar panel.



2.42.2 Constructor & Destructor Documentation

2.42.2.1 SolarPanel()

```
osse::collaborate::SolarPanel::SolarPanel (
            const double & _efficiency_percent,
            const double & _surface_area_m2,
            const double & _roll_rad,
            const double & _pitch_rad,
            const double & _yaw_rad,
            Sun * _sun )
```

Constructor from attitude angles.

Parameters

in	*_efficiency_percent*	Efficiency (percent)
in	*_surface_area_m2*	Surface area (meters squared)
in	*_roll_rad*	Roll angle to host body frame (radians)
in	*_pitch_rad*	Pitch angle to host body frame (radians)
in	*_yaw_rad*	Yaw angle to host body frame (radians)
in	*_sun*	Sun

2.42.3 Member Function Documentation

2.42.3.1 effective_area_m2()

```
const double& osse::collaborate::SolarPanel::effective_area_m2 ( ) const  [inline]
```

Get Effective area (meters squared)

Returns

effective_area_m2_ Effective area (meters squared)

2.42.3.2 RxPowerW()

```
double osse::collaborate::SolarPanel::RxPowerW ( ) const
```

Obtain the received power in a direction (Watts)

Returns

Power received in a specified direction (Watts)

$$I = Solar\ Irradiance \quad (Watt\ per\ square\ meter)$$
$$\sigma = Effective\ Area \quad (meters\ squared)$$
$$\eta = Efficiency \quad (percent)$$
$$P = I\eta\sigma \quad (Watts)$$

2.42.3.3 Update()

```
void osse::collaborate::SolarPanel::Update (
            const ReferenceFrame & _body_frame,
            const ReferenceFrame & _orbit_frame,
            const Vector & _position_m_rad )
```

Updates the attitude frame orientation and effective area.

Parameters

in	*_body_frame*	Host's body frame
in	*_orbit_frame*	Host's orbit frame
in	*_position_m_rad*	Host's position (meters and radians)

$$\vec{p}_{sun} = Solar\ Position \quad (meters)$$
$$\vec{p} = Current\ Position \quad (meters)$$
$$A = Total\ Surface\ Area \quad (meters\ squared)$$
$$\hat{n} = Unit\ Normal\ Vector,\ Attitude\ Z - Axis$$
$$\sigma = A\cos\left(\angle(\hat{n}, \vec{r}_{p,sun})\right) \quad (meters)$$

The documentation for this class was generated from the following files:

- libs/collaborate/include/collaborate/solar_panel.h
- libs/collaborate/src/solar_panel.cpp

2.43 osse::collaborate::SubsystemComm Class Reference

An interface for RF communication.

`#include <subsystem_comm.h>`

Data Structures

- struct CommunicationEvent

  A plan to transfer a packet to another node.
- struct FeedbackEvent

  A plan to feedback a packet to the original node.

Public Types

- enum kMode { **Free**, **Transmitting**, **Receiving** }

  Possible modes of operation.
- typedef struct osse::collaborate::SubsystemComm::CommunicationEvent CommunicationEvent

  A plan to transfer a packet to another node.
- typedef struct osse::collaborate::SubsystemComm::FeedbackEvent FeedbackEvent

  A plan to feedback a packet to the original node.

Public Member Functions

- SubsystemComm (const Antenna *_antenna, const Modem *_modem)

  Constructor.
- uint16_t Update (const SimulationClock &_clock)

  Update the communication interface.
- uint64_t RequiredTransferDurationS () const

  Calculate the required transfer duration (seconds)
- uint64_t RequiredTransferDurationS (const uint64_t &_buffer_size_bytes) const

  Calculate the required transfer duration (seconds)
- double CalculatePowerDrainW ()

  Calculate the power consumed during operation.
- void AddToStorage (CommunicationEvent _event)

  Adds a planned event to storage for later.
- void AddToStorage (FeedbackEvent _feedback_event)

  Adds a planned event to storage for later.
- void OrientAntenna (const ReferenceFrame &_orbit_frame, const ReferenceFrame &_body↩ _frame)

  Calculates a new attitude frame for the antenna.
- void LoadData (const std::vector< uint8_t > &_payload)

  Adds data to the data buffer.
- void EraseDataBuffer ()

  Empties data buffer.
- void set_data_buffer (const std::vector< uint8_t > &_data_buffer)

  Set data buffer.
- const Antenna * kAntenna () const

  Get Antenna.
- ReferenceFrame antenna_frame () const

  Get Antenna reference frame.
- const std::vector< uint8_t > & data_buffer () const

  Get Data buffer.
- const bool & active () const

  Activity status.
- const uint64_t & elapsed_s () const

  Get time counter (seconds)
- void set_mode (const kMode &_mode)

  Set mode of operation.
- const kMode & mode () const

  Get mode of operation.
- const Modem * kModem () const

  Get kModem.

- const std::vector< CommunicationEvent > & storage ()

  Get list of control packets.
- const std::vector< FeedbackEvent > & feedback_storage ()

  Get list of return packets.

Private Attributes

- const Modem * kModem_

  The modem.
- std::vector< CommunicationEvent > storage_

  List of control packets.
- kMode mode_

  The mode of operation.
- std::vector< FeedbackEvent > feedback_storage_

  List of return packets.
- const Antenna * kAntenna_

  Antenna.
- ReferenceFrame antenna_frame_

  Antenna reference frame.
- std::vector< uint8_t > data_buffer_

  Data buffer.
- bool active_

  Activity status.
- uint64_t elapsed_s_

  Get time counter (seconds)

2.43.1 Detailed Description

An interface for RF communication.

2.43.2 Constructor & Destructor Documentation

2.43.2.1 SubsystemComm()

```
osse::collaborate::SubsystemComm::SubsystemComm (
            const Antenna * _antenna,
            const Modem * _modem )
```

Constructor.

Parameters

in	*_antenna*	Antenna
in	*_modem*	Modem

2.43.3 Member Function Documentation

2.43.3.1 active()

```
const bool& osse::collaborate::SubsystemComm::active ( ) const  [inline]
```

Activity status.

Returns

active_ Activity status

2.43.3.2 AddToStorage() [1/2]

```
void osse::collaborate::SubsystemComm::AddToStorage (
            SubsystemComm::CommunicationEvent _event )
```

Adds a planned event to storage for later.

Parameters

in	*_event*	Planned commmunication event

2.43.3.3 AddToStorage() [2/2]

```
void osse::collaborate::SubsystemComm::AddToStorage (
            SubsystemComm::FeedbackEvent _feedback_event )
```

Adds a planned event to storage for later.

Parameters

in	*_feedback_event*	Planned feedback event

2.43.3.4 antenna_frame()

```
ReferenceFrame osse::collaborate::SubsystemComm::antenna_frame ( ) const  [inline]
```

Get Antenna reference frame.

Returns

antenna_frame_ Antenna reference frame

2.43.3.5 CalculatePowerDrainW()

```
double osse::collaborate::SubsystemComm::CalculatePowerDrainW ( )
```

Calculate the power consumed during operation.

Returns

Power consumed during operation

2.43.3.6 data_buffer()

```
const std::vector<uint8_t>& osse::collaborate::SubsystemComm::data_buffer ( ) const  [inline]
```

Get Data buffer.

Returns

data_buffer_ Data buffer

2.43.3.7 elapsed_s()

```
const uint64_t& osse::collaborate::SubsystemComm::elapsed_s ( ) const  [inline]
```

Get time counter (seconds)

Returns

elapsed_s_ Time counter (seconds)

2.43.3.8 feedback_storage()

```
const std::vector<FeedbackEvent>& osse::collaborate::SubsystemComm::feedback_storage ( )
[inline]
```

Get list of return packets.

Returns

feedback_storage_ List of return packets

2.43.3.9 kAntenna()

```
const Antenna* osse::collaborate::SubsystemComm::kAntenna ( ) const  [inline]
```

Get Antenna.

Returns

kAntenna_ Antenna

2.43.3.10 kModem()

```
const Modem* osse::collaborate::SubsystemComm::kModem ( ) const  [inline]
```

Get kModem.

Returns

kModem_ Modem

2.43.3.11 LoadData()

```
void osse::collaborate::SubsystemComm::LoadData (
            const std::vector< uint8_t > & _payload )
```

Adds data to the data buffer.

Parameters

in	*_payload*	Payload

2.43.3.12 mode()

```
const kMode& osse::collaborate::SubsystemComm::mode ( ) const  [inline]
```

Get mode of operation.

Returns

mode_ Mode of operation

2.43.3.13 OrientAntenna()

```
void osse::collaborate::SubsystemComm::OrientAntenna (
            const ReferenceFrame & _orbit_frame,
            const ReferenceFrame & _body_frame )
```

Calculates a new attitude frame for the antenna.

Parameters

_orbit_frame	Satellite orbit frame
_body_frame	Satellite body frame

2.43.3.14 RequiredTransferDurationS() [1/2]

```
uint64_t osse::collaborate::SubsystemComm::RequiredTransferDurationS ( ) const
```

Calculate the required transfer duration (seconds)

Returns

Required transfer duration (seconds)

2.43.3.15 RequiredTransferDurationS() [2/2]

```
uint64_t osse::collaborate::SubsystemComm::RequiredTransferDurationS (
            const uint64_t & _buffer_size_bytes ) const
```

Calculate the required transfer duration (seconds)

Returns

Required transfer duration (seconds)

Parameters

in	*_buffer_size_bytes*	Number of bytes in a predicted buffer

2.43.3.16 set_data_buffer()

```
void osse::collaborate::SubsystemComm::set_data_buffer (
            const std::vector< uint8_t > & _data_buffer )
```

Set data buffer.

Parameters

in	*_data_buffer*	Data buffer

2.43.3.17 set_mode()

```
void osse::collaborate::SubsystemComm::set_mode (
            const kMode & _mode )
```

Set mode of operation.

Parameters

in	*_mode*	Mode of operation

2.43.3.18 storage()

```
const std::vector<CommunicationEvent>& osse::collaborate::SubsystemComm::storage ( )  [inline]
```

Get list of control packets.

Returns

storage_ List of control packets

2.43.3.19 Update()

```
uint16_t osse::collaborate::SubsystemComm::Update (
            const SimulationClock & _clock )
```

Update the communication interface.

Parameters

in	*_clock*	Simulation clock

Returns

Target receiver index if one has been selected

The documentation for this class was generated from the following files:

- libs/collaborate/include/collaborate/subsystem_comm.h
- libs/collaborate/src/subsystem_comm.cpp

2.44 osse::collaborate::SubsystemPower Class Reference

```
#include <subsystem_power.h>
```

Public Member Functions

- SubsystemPower (const Battery &_battery, const std::vector< SolarPanel > _solar_panels, const double &_idle_power_w)

  Constructor.
- void Update (const bool &_charge, const SimulationClock &_simulation_clock, const ReferenceFrame &_body_frame, const ReferenceFrame &_orbit_frame, const double &_power_drain_w, const Vector &_position_m_rad)

  Updates the system's net energy stored.
- const Battery & battery () const

  Get Battery.
- const std::vector< SolarPanel > & solar_panels () const

  Get Panels.
- const bool & charging () const

  Get whether the system is charging.

Static Public Attributes

- static constexpr int kSecsPerHr = 3600

  Number of seconds per hour.

Private Attributes

- bool charging_

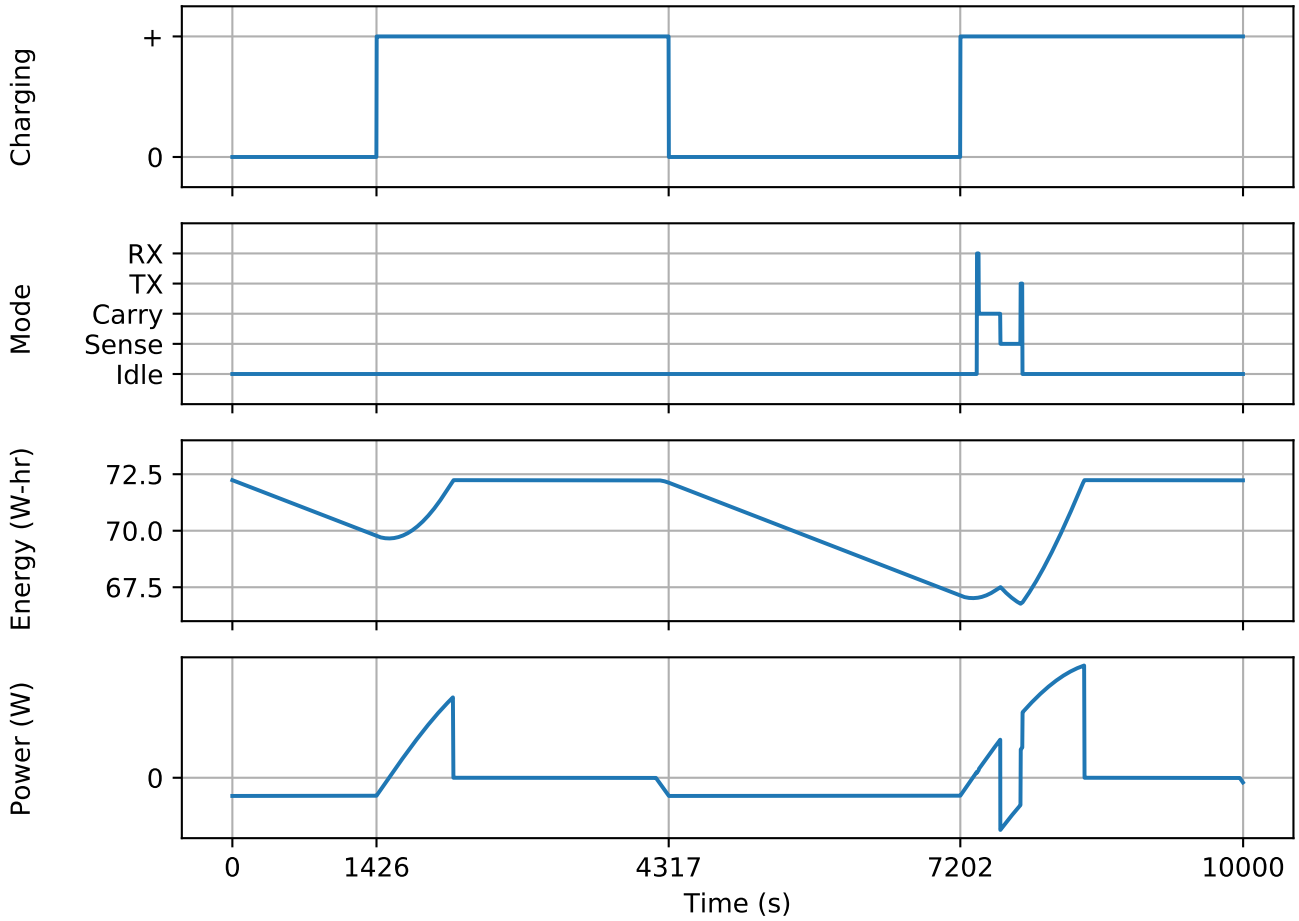
  Whether the system is charging.
- Battery battery_

  Battery.
- std::vector< SolarPanel > solar_panels_

  List of solar panels.
- double idle_power_w_

  Idle consumed power.

2.44.1 Detailed Description



2.44.2 Constructor & Destructor Documentation

2.44.2.1 SubsystemPower()

```
osse::collaborate::SubsystemPower::SubsystemPower (
            const Battery & _battery,
            const std::vector< SolarPanel > _solar_panels,
            const double & _idle_power_w )
```

Constructor.

Parameters

in	*_battery*	Battery
in	*_solar_panels*	Solar panels
in	*_idle_power_w*	Idle power consumed

2.44.3 Member Function Documentation

2.44.3.1 battery()

```
const Battery& osse::collaborate::SubsystemPower::battery ( ) const  [inline]
```

Get Battery.

Returns

battery_ Battery

2.44.3.2 charging()

```
const bool& osse::collaborate::SubsystemPower::charging ( ) const  [inline]
```

Get whether the system is charging.

Returns

charging_ Whether the system is charging

2.44.3.3 solar_panels()

```
const std::vector<SolarPanel>& osse::collaborate::SubsystemPower::solar_panels ( ) const
[inline]
```

Get Panels.

Returns

solar_panels_ Panels

2.44.3.4 Update()

```
void osse::collaborate::SubsystemPower::Update (
            const bool & _charge,
            const SimulationClock & _simulation_clock,
            const ReferenceFrame & _body_frame,
            const ReferenceFrame & _orbit_frame,
            const double & _power_drain_w,
            const Vector & _position_m_rad )
```

Updates the system's net energy stored.

Parameters

in	*_charge*	Whether to charge the battery
in	*_simulation_clock*	Simulation simulation_clock
in	*_body_frame*	Host body reference_frame
in	*_orbit_frame*	Host orbit reference_frame
in	*_power_drain_w*	Power drained since last update
in	*_position_m_rad*	Host's position (meters and radians)

The documentation for this class was generated from the following files:

- libs/collaborate/include/collaborate/subsystem_power.h
- libs/collaborate/src/subsystem_power.cpp

2.45 osse::collaborate::SubsystemSensing Class Reference

An interface for RF sensing.

`#include <subsystem_sensing.h>`

Data Structures

- struct LogBuffer

  A buffer for logged node data.

Public Types

- typedef struct osse::collaborate::SubsystemSensing::LogBuffer LogBuffer

  A buffer for logged node data.

Public Member Functions

- SubsystemSensing (const Antenna *_antenna, Sensor *_sensor)

  Constructor.
- void Measure (const uint16_t &_informer_index)

  Measures for a specified time.
- bool Update (const SimulationClock &_clock, const Vector &_position_m_rad)

  Obtain a measurement from the earth_data.
- void OrientAntenna (const ReferenceFrame &_orbit_frame, const ReferenceFrame &_body↩ _frame)

Calculates a new attitude frame for the antenna.

- void LoadData (const std::vector< uint8_t > &_payload)

  Adds data to the data buffer.
- void EraseDataBuffer ()

  Empties data buffer.
- void set_data_buffer (const std::vector< uint8_t > &_data_buffer)

  Set data buffer.
- const Antenna * kAntenna () const

  Get Antenna.
- ReferenceFrame antenna_frame () const

  Get Antenna reference frame.
- const std::vector< uint8_t > & data_buffer () const

  Get Data buffer.
- const bool & active () const

  Activity status.
- const uint64_t & elapsed_s () const

  Get time counter (seconds)
- void set_complete (const bool &_complete)

  Sets the complete flag.
- Sensor * sensor () const

  Get The sensor.
- const uint64_t & expiration_s () const

  Get The time limit for activity.
- const bool & complete () const

  Get whether or not the measurement is complete.

Private Member Functions

- void Flush (const SimulationClock &_clock)

  Output the logged data to a netcdf file.

Private Attributes

- Sensor * sensor_

  The sensor.
- uint64_t expiration_s_

  The time limit for activity.
- bool complete_

  Whether or not the measurement is complete.
- uint16_t informer_index_

Index of the informer node.

- LogBuffer buffer_

  Buffer for logged data.

- const Antenna * kAntenna_

  Antenna.

- ReferenceFrame antenna_frame_

  Antenna reference frame.

- std::vector< uint8_t > data_buffer_

  Data buffer.

- bool active_

  Activity status.

- uint64_t elapsed_s_

  Get time counter (seconds)

2.45.1 Detailed Description

An interface for RF sensing.

2.45.2 Constructor & Destructor Documentation

2.45.2.1 SubsystemSensing()

```
osse::collaborate::SubsystemSensing::SubsystemSensing (
            const Antenna * _antenna,
            Sensor * _sensor )
```

Constructor.

Parameters

in	*_antenna*	The antenna
in	*_sensor*	The sensor

2.45.3 Member Function Documentation

2.45.3.1 active()

```
const bool& osse::collaborate::SubsystemSensing::active ( ) const  [inline]
```

Activity status.

Returns

active_ Activity status

2.45.3.2 antenna_frame()

```
ReferenceFrame osse::collaborate::SubsystemSensing::antenna_frame ( ) const  [inline]
```

Get Antenna reference frame.

Returns

antenna_frame_ Antenna reference frame

2.45.3.3 complete()

```
const bool& osse::collaborate::SubsystemSensing::complete ( ) const  [inline]
```

Get whether or not the measurement is complete.

Returns

complete_ Whether or not the measurement is complete

2.45.3.4 data_buffer()

```
const std::vector<uint8_t>& osse::collaborate::SubsystemSensing::data_buffer ( ) const
[inline]
```

Get Data buffer.

Returns

data_buffer_ Data buffer

2.45.3.5 elapsed_s()

```
const uint64_t& osse::collaborate::SubsystemSensing::elapsed_s ( ) const  [inline]
```

Get time counter (seconds)

Returns

elapsed_s_ Time counter (seconds)

2.45.3.6 expiration_s()

```
const uint64_t& osse::collaborate::SubsystemSensing::expiration_s ( ) const  [inline]
```

Get The time limit for activity.

Returns

expiration_s_ The time limit for activity

2.45.3.7 Flush()

```
void osse::collaborate::SubsystemSensing::Flush (
            const SimulationClock & _clock )  [private]
```

Output the logged data to a netcdf file.

Parameters

in	*_clock*	The simulation clock

2.45.3.8 kAntenna()

```
const Antenna* osse::collaborate::SubsystemSensing::kAntenna ( ) const  [inline]
```

Get Antenna.

Returns

kAntenna_ Antenna

2.45.3.9 LoadData()

```
void osse::collaborate::SubsystemSensing::LoadData (
            const std::vector< uint8_t > & _payload )
```

Adds data to the data buffer.

Parameters

in	*_payload*	Payload

2.45.3.10 Measure()

```
void osse::collaborate::SubsystemSensing::Measure (
            const uint16_t & _informer_index )
```

Measures for a specified time.

Parameters

in	*_informer_index*	Index of the informer node

2.45.3.11 OrientAntenna()

```
void osse::collaborate::SubsystemSensing::OrientAntenna (
            const ReferenceFrame & _orbit_frame,
            const ReferenceFrame & _body_frame )
```

Calculates a new attitude frame for the antenna.

Parameters

_orbit_frame	Satellite orbit frame
_body_frame	Satellite body frame

2.45.3.12 sensor()

```
Sensor* osse::collaborate::SubsystemSensing::sensor ( ) const  [inline]
```

Get The sensor.

Returns

sensor_ The sensor

2.45.3.13 set_complete()

```
void osse::collaborate::SubsystemSensing::set_complete (
            const bool & _complete )  [inline]
```

Sets the complete flag.

Parameters

in	*_complete*	

2.45.3.14 set_data_buffer()

```
void osse::collaborate::SubsystemSensing::set_data_buffer (
            const std::vector< uint8_t > & _data_buffer )
```

Set data buffer.

Parameters

in	*_data_buffer*	Data buffer

2.45.3.15 Update()

```
bool osse::collaborate::SubsystemSensing::Update (
            const SimulationClock & _clock,
            const Vector & _position_m_rad )
```

Obtain a measurement from the earth_data.

Parameters

in	*_clock*	The simulation clock
in	*_position_m_rad*	The position of the node

Returns

active_

The documentation for this class was generated from the following files:

- libs/collaborate/include/collaborate/subsystem_sensing.h
- libs/collaborate/src/subsystem_sensing.cpp

2.46 osse::collaborate::Sun Class Reference

The star at the center of the solar system.

`#include <sun.h>`

Public Member Functions

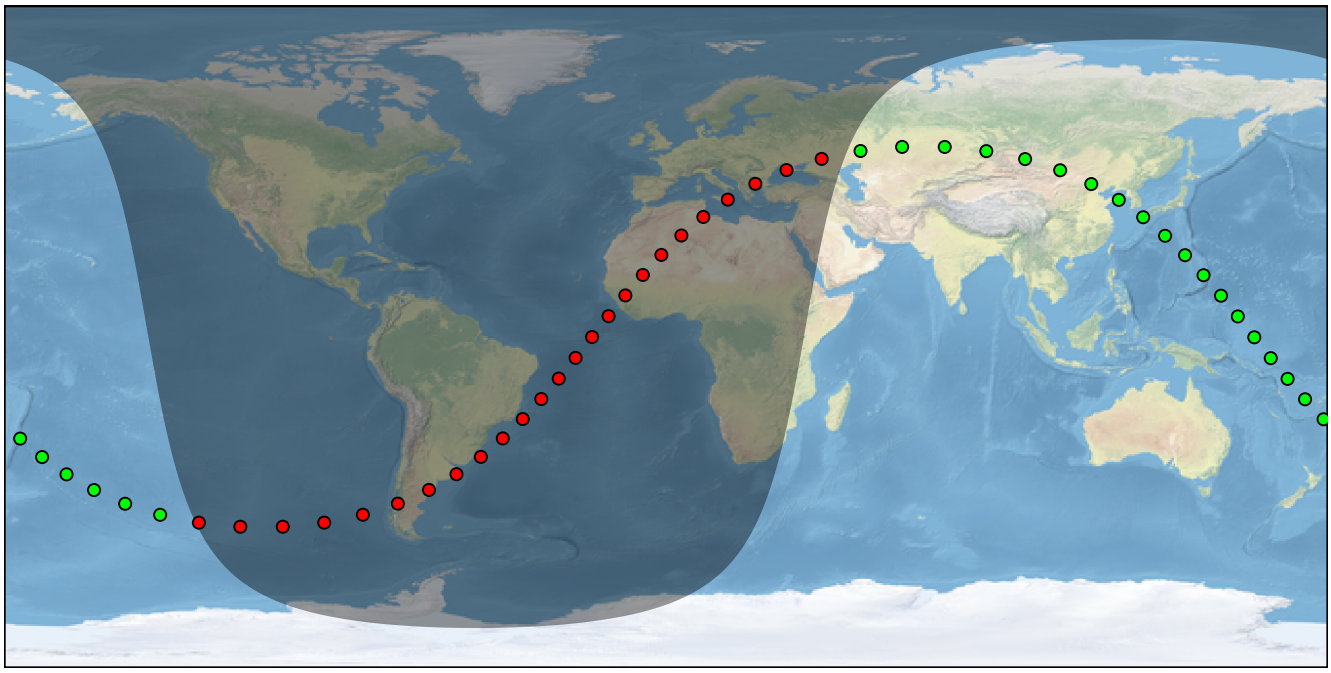
- Sun (SimulationClock *_clock)
  Constructor.
- void Update (const uint64_t &_offset_s)
  Update the position.
- Vector PositionMRad () const
  Obtain position (meters and radians)

Private Attributes

- SimulationClock * clock_
  Simulation clock.
- Vector position_m_rad_
  Position (meters and radians)

2.46.1 Detailed Description

The star at the center of the solar system.



2.46.2 Constructor & Destructor Documentation

2.46.2.1 Sun()

```
osse::collaborate::Sun::Sun (
            SimulationClock * _clock ) [explicit]
```

Constructor.

Parameters

in	*_clock*	Simulation clock

2.46.3 Member Function Documentation

2.46.3.1 PositionMRad()

```
Vector osse::collaborate::Sun::PositionMRad ( ) const  [inline]
```

Obtain position (meters and radians)

Returns

Position (meters and radians)

2.46.3.2 Update()

```
void osse::collaborate::Sun::Update (
            const uint64_t & _offset_s )
```

Update the position.

Parameters

in	_↩ *offset*↩ _s	Offset in time (seconds)

The documentation for this class was generated from the following files:

- libs/collaborate/include/collaborate/sun.h
- libs/collaborate/src/sun.cpp

2.47 osse::collaborate::Tree Class Reference

A tree.

```
#include <tree.h>
```

Data Structures

- struct Branch

  A node of the tree.

Public Types

- typedef struct osse::collaborate::Tree::Branch Branch

  A node of the tree.

Public Member Functions

- Tree (Node *_root, const uint16_t &_height, Node *_target)

  Constructor.
- ~Tree ()

  Destructor.
- void DeleteSubtree (Branch *_branch)

  Deletes a subtree.
- Branch * CreateRoot (Node *_root_identity)

  Creates a root node.
- Branch * AddChild (Branch *_parent, Node *_identity, const uint64_t &_rx_time_s)

  Add a child to the tree.
- std::vector< Branch * > Ancestry (Branch *_branch) const

  Obtain list of direct parents (back to root node)
- Branch * SearchSpecific (Node *_identity, const uint64_t &_rx_time_s) const

  Perform a BFS on the tree to find a node with specified rx time.
- Branch * BreadthFirstSearch (Node *_identity) const

  Perform a BFS on the tree to find a node.
- bool HasChild (Branch *_branch, Node *_identity) const

  Determine if a node is a current child of a node.
- bool IsLeaf (Branch *_branch) const

  Determine if a node has no children.
- void Log (const std::string &_path) const

  Write the tree to a LaTeX Tikz file.
- void set_root (Branch *_root)

  Set the root tree node.
- void set_height (const uint16_t &_height)

  Set the height.
- void set_target (Node *_target)

  Set the target node.
- Branch * root () const

  Get the root tree node.
- const uint64_t & size () const

  Get the number of tree nodes in the tree.
- const uint16_t & height () const

  Get the height.
- Node * target () const

  Get the target node.

Private Attributes

- Branch * root_

  Root tree node.
- uint64_t size_

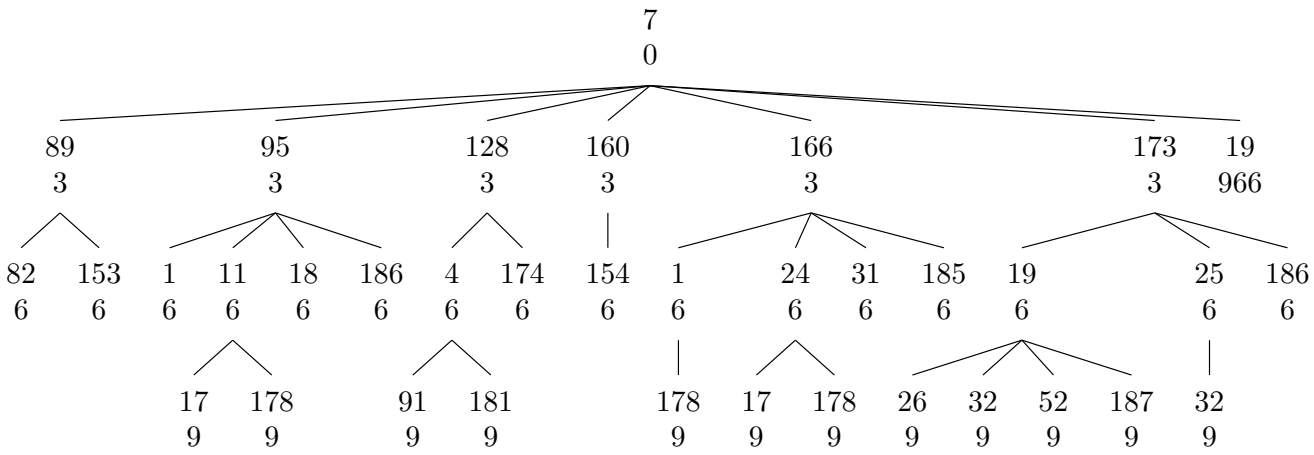
  Number of tree nodes in the tree.
- uint16_t height_

  Height.
- Node * target_

  Target node;.

2.47.1 Detailed Description

A tree.



2.47.2 Constructor & Destructor Documentation

2.47.2.1 Tree()

```
osse::collaborate::Tree::Tree (
            Node * _root,
            const uint16_t & _height,
            Node * _target )
```

Constructor.

Parameters

in	*_root*	Root node
in	*_height*	Maximum levels in the tree
in	*_target*	Target node

2.47.3 Member Function Documentation

2.47.3.1 AddChild()

```
Tree::Branch * osse::collaborate::Tree::AddChild (
            Branch * _parent,
            Node * _identity,
            const uint64_t & _rx_time_s )
```

Add a child to the tree.

Parameters

in	*_parent*	Parent branch
in	*_identity*	New satellite node to add
in	*_rx_time_s*	Message receive time of the new node (seconds)

Returns

New branch child

2.47.3.2 Ancestry()

```
std::vector< Tree::Branch * > osse::collaborate::Tree::Ancestry (
            Tree::Branch * _branch ) const
```

Obtain list of direct parents (back to root node)

Parameters

in	*_branch*	Lowest node in the ancestry

Returns

List of direct parents, back to the root

2.47.3.3 BreadthFirstSearch()

```
Tree::Branch * osse::collaborate::Tree::BreadthFirstSearch (
            Node * _identity ) const
```

Perform a BFS on the tree to find a node.

Parameters

in	*_identity*	Node to search for

Returns

First branch found from top down and left to right

2.47.3.4 CreateRoot()

```
Tree::Branch * osse::collaborate::Tree::CreateRoot (
            Node * _root_identity )
```

Creates a root node.

Parameters

in	*_root_identity*	The root identity

Returns

A root node

2.47.3.5 DeleteSubtree()

```
void osse::collaborate::Tree::DeleteSubtree (
            Tree::Branch * _branch )
```

Deletes a subtree.

Parameters

in	*_branch*	Root of the substree

2.47.3.6 HasChild()

```
bool osse::collaborate::Tree::HasChild (
            Tree::Branch * _branch,
            Node * _identity ) const
```

Determine if a node is a current child of a node.

Parameters

in	*_branch*	Tree node parent
in	*_identity*	Node child

Returns

Whether the child exists

2.47.3.7 height()

```
const uint16_t& osse::collaborate::Tree::height ( ) const  [inline]
```

Get the height.

Returns

height_ Height

2.47.3.8 IsLeaf()

```
bool osse::collaborate::Tree::IsLeaf (
            Branch * _branch ) const
```

Determine if a node has no children.

Parameters

in	*_branch*	Tree node

Returns

Whether it is a leaf

2.47.3.9 Log()

```
void osse::collaborate::Tree::Log (
            const std::string & _path ) const
```

Write the tree to a LaTeX Tikz file.

Parameters

in	*_path*	Path to the log file

2.47.3.10 root()

```
Branch* osse::collaborate::Tree::root ( ) const  [inline]
```

Get the root tree node.

Returns

root_ Root tree node

2.47.3.11 SearchSpecific()

```
Tree::Branch * osse::collaborate::Tree::SearchSpecific (
            Node * _identity,
            const uint64_t & _rx_time_s ) const
```

Perform a BFS on the tree to find a node with specified rx time.

Parameters

in	*_identity*	Node to search for
in	*_rx_↩ time_s*	Message receive time of the new node (seconds)

Returns

Specific branch found

2.47.3.12 set_height()

```
void osse::collaborate::Tree::set_height (
            const uint16_t & _height ) [inline]
```

Set the height.

Parameters

in	*_height*	Height

2.47.3.13 set_root()

```
void osse::collaborate::Tree::set_root (
            Branch * _root ) [inline]
```

Set the root tree node.

Parameters

_root	Root tree node

2.47.3.14 set_target()

```
void osse::collaborate::Tree::set_target (
            Node * _target ) [inline]
```

Set the target node.

Parameters

_target	Target node

2.47.3.15 size()

```
const uint64_t& osse::collaborate::Tree::size ( ) const  [inline]
```

Get the number of tree nodes in the tree.

Returns

size_ Number of tree nodes in the tree

2.47.3.16 target()

```
Node* osse::collaborate::Tree::target ( ) const  [inline]
```

Get the target node.

Returns

target_ Target node

The documentation for this class was generated from the following files:

- libs/collaborate/include/collaborate/tree.h
- libs/collaborate/src/tree.cpp

2.48 osse::collaborate::Vector Class Reference

An element of the real coordinate space (cartesian and spherical)

```
#include <vector.h>
```

Public Member Functions

- Vector ()

  Default Constructor.
- Vector (double _x_m, double _y_m, double _z_m)

  3D Cartesian Constructor - Calculates the 4th (magnitude)
- void CompleteCoordinates ()

  Updates the vector's spherical and cylindrical coordinates.
- double CalculateThetaRad () const

  Calculates a theta value between 0 and pi (radians)
- double CalculatePhiRad () const

  Calculates a phi value between 0 and 2∗pi (radians)
- std::vector< double > ObtainLog () const

  Logs the Vector.
- Vector operator- () const

  Negative operator.
- Vector operator+ (const Vector &_other) const

  Addition operator.
- Vector operator- (const Vector &_other) const

  Subtraction operator.
- Vector Unit () const

  Produces a unit vector in the current direction.
- double Dot (const Vector &_other) const

  Calculates the dot product.
- Vector Cross (const Vector &_other) const

  Calculates the cross product between two vectors.
- Vector operator∗ (const double &_scalar) const

  Multiplication operator.
- Vector operator/ (const double &_scalar) const

  Division operator.
- operator bool () const

  Boolean operator.
- double AngleBetween (const Vector &_other) const

  Calculate the angle between two coordinates.
- Vector OrthoNormal (const Vector &_other) const

  Calculate the unit vector orthonormal to other vector.
- Vector ConstraintToPlane (const Vector &_other, const Vector &_reference) const

  Constrain to plane.
- std::string ToString () const

  Outputs the vector to a string.
- const double & x_m () const

Get X value (meters)

- const double & y_m () const

  Get Y value (meters)
- const double & z_m () const

  Get z_value (meters)
- const double & r_m () const

  Get maginitude (meters)
- const double & rho_m () const

  Get the distance from z-axis in the x/y-plane (meters)
- const double & theta_rad () const

  Get angle from the positive z-axis (radians)
- const double & phi_rad () const

  Get angle from the positive x-axis (radians)

Private Attributes

- double x_m_

  X value (meters)
- double y_m_

  Y value (meters)
- double z_m_

  Z value (meters)
- double r_m_

  Magnitude (meters)
- double rho_m_

  Distance from z-axis in the x/y-plane (meters)
- double theta_rad_

  Angle from the positive z-axis.
- double phi_rad_

  Angle from the positive x-axis.

2.48.1 Detailed Description

An element of the real coordinate space (cartesian and spherical)

$$\vec{v} = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} \quad (meters)$$

2.48.2 Constructor & Destructor Documentation

2.48.2.1 Vector() [1/2]

```
osse::collaborate::Vector::Vector ( )
```

Default Constructor.

$$\vec{s} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad (meters)$$

2.48.2.2 Vector() [2/2]

```
osse::collaborate::Vector::Vector (
            double _x_m,
            double _y_m,
            double _z_m )
```

3D Cartesian Constructor - Calculates the 4th (magnitude)

Parameters

in	*_x_m*	X value (meters)
in	*_y_m*	Y value (meters)
in	*_z_m*	Z value (meters)

$$\vec{s} = \begin{bmatrix} s_x \\ s_y \\ s_z \end{bmatrix} \quad (meters)$$

2.48.3 Member Function Documentation

2.48.3.1 AngleBetween()

```
double osse::collaborate::Vector::AngleBetween (
            const Vector & _other ) const
```

Calculate the angle between two coordinates.

Parameters

in	*_other*	Other coordinates

Returns

Angle between the coordinates (radians)

$$\angle(s,t) = \arccos\left(\frac{s \cdot t}{|s||t|}\right)$$

2.48.3.2 CalculatePhiRad()

```
double osse::collaborate::Vector::CalculatePhiRad ( ) const
```

Calculates a phi value between 0 and 2∗pi (radians)

Returns

A phi value between 0 and 2∗pi (radians)

$$\phi = \arctan\left(\frac{y}{x}\right) (\text{mod } 2\pi) \quad (radians)$$

2.48.3.3 CalculateThetaRad()

```
double osse::collaborate::Vector::CalculateThetaRad ( ) const
```

Calculates a theta value between 0 and pi (radians)

Returns

A theta value between 0 and pi (radians)

$$\theta = \left|\arctan\left(\frac{\rho}{z}\right)\right| \quad (radians)$$

2.48.3.4 CompleteCoordinates()

```
void osse::collaborate::Vector::CompleteCoordinates ( )
```

Updates the vector's spherical and cylindrical coordinates.

$$r = \sqrt{x^2 + y^2 + z^2} \quad (meters)$$
$$\rho = \sqrt{x^2 + y^2} \quad (meters)$$
$$\theta = \left|\arctan\left(\frac{\rho}{z}\right)\right| \quad (radians)$$
$$\phi = \arctan\frac{y}{x} (\text{mod } 2\pi) \quad (radians)$$

2.48.3.5 ConstraintToPlane()

```
Vector osse::collaborate::Vector::ConstraintToPlane (
            const Vector & _other,
            const Vector & _reference ) const
```

Constrain to plane.

Parameters

in	*_other*	Other vector
in	*_reference*	Reference vector

Returns

Vector in the same plane as other and reference

2.48.3.6 Cross()

```
Vector osse::collaborate::Vector::Cross (
            const Vector & _other ) const
```

Calculates the cross product between two vectors.

Parameters

in	*_other*	Other vector

Returns

Cross product

$$\vec{s} \times \vec{t} = \begin{bmatrix} s_y t_z - s_z t_y \\ s_z t_x - s_x t_z \\ s_x t_y - s_y t_x \end{bmatrix} \quad (meters)$$

2.48.3.7 Dot()

```
double osse::collaborate::Vector::Dot (
            const Vector & _other ) const
```

Calculates the dot product.

Parameters

in	*_other*	Other vector

Returns

Dot product

$$\vec{s} \cdot \vec{t} = s_x t_x + s_y t_y + s_z t_z$$

2.48.3.8 ObtainLog()

```
std::vector< double > osse::collaborate::Vector::ObtainLog ( ) const
```

Logs the Vector.

Returns

A vector of doubles

2.48.3.9 operator bool()

```
osse::collaborate::Vector::operator bool ( ) const  [inline], [explicit]
```

Boolean operator.

Returns

True if vector is non-zero

2.48.3.10 operator∗()

```
Vector osse::collaborate::Vector::operator∗ (
            const double & _scalar ) const  [inline]
```

Multiplication operator.

Parameters

in	*_scalar*	Scaling value

Returns

Scaled vector

$$c\vec{s} = \begin{bmatrix} cs_x \\ cs_y \\ cs_z \end{bmatrix} \quad (meters)$$

2.48.3.11 operator+()

```
Vector osse::collaborate::Vector::operator+ (
            const Vector & _other ) const  [inline]
```

Addition operator.

Parameters

in	*_other*	Other vector

Returns

Sum of the vectors

$$\vec{s} + \vec{t} = \begin{bmatrix} s_x + t_x \\ s_y + t_y \\ s_z + t_z \end{bmatrix} \quad (meters)$$

2.48.3.12 operator-() [1/2]

```
Vector osse::collaborate::Vector::operator- ( ) const  [inline]
```

Negative operator.

Returns

Negative of the vector

$$-\vec{s} = \begin{bmatrix} -x \\ -y \\ -z \end{bmatrix} \quad (meters)$$

2.48.3.13 operator-() [2/2]

```
Vector osse::collaborate::Vector::operator- (
            const Vector & _other ) const  [inline]
```

Subtraction operator.

Parameters

in	*_other*	Other vector

Returns

Difference between the vectors

$$\vec{s} - \vec{t} = \begin{bmatrix} s_x - t_x \\ s_y - t_y \\ s_z - t_z \end{bmatrix} \quad (meters)$$

2.48.3.14 operator/()

```
Vector osse::collaborate::Vector::operator/ (
            const double & _scalar ) const  [inline]
```

Division operator.

Parameters

in	*_scalar*	Scaling value

Returns

Scaled vector

$$c\vec{s} = \begin{bmatrix} \frac{s_x}{c} \\ \frac{s_y}{c} \\ \frac{s_z}{c} \end{bmatrix} \quad (meters)$$

2.48.3.15 OrthoNormal()

```
Vector osse::collaborate::Vector::OrthoNormal (
            const Vector & _other ) const
```

Calculate the unit vector orthonormal to other vector.

Parameters

in	*_other*	Other coordinates

Returns

Unit vector orthonormal to other vector

2.48.3.16 phi_rad()

```
const double& osse::collaborate::Vector::phi_rad ( ) const  [inline]
```

Get angle from the positive x-axis (radians)

Returns

_phi_rad Angle from the positive x-axis (radians)

2.48.3.17 r_m()

```
const double& osse::collaborate::Vector::r_m ( ) const  [inline]
```

Get maginitude (meters)

Returns

_r_m Magnutude (meters)

2.48.3.18 rho_m()

```
const double& osse::collaborate::Vector::rho_m ( ) const  [inline]
```

Get the distance from z-axis in the x/y-plane (meters)

Returns

_rho_rad Distance from z-axis in the x/y-plane (meters)

2.48.3.19 theta_rad()

```
const double& osse::collaborate::Vector::theta_rad ( ) const  [inline]
```

Get angle from the positive z-axis (radians)

Returns

_theta_rad Angle from the positive z-axis (radians)

2.48.3.20 ToString()

```
std::string osse::collaborate::Vector::ToString ( ) const
```

Outputs the vector to a string.

Returns

Vector as a string

2.48.3.21 Unit()

```
Vector osse::collaborate::Vector::Unit ( ) const
```

Produces a unit vector in the current direction.

Returns

Unit vector in the current direction

$$\hat{s} = \frac{\vec{s}}{|s|}$$

2.48.3.22 x_m()

```
const double& osse::collaborate::Vector::x_m ( ) const  [inline]
```

Get X value (meters)

Returns

_x_m X value (meters)

2.48.3.23 y_m()

```
const double& osse::collaborate::Vector::y_m ( ) const  [inline]
```

Get Y value (meters)

Returns

_y_m Y value (meters)

2.48.3.24 z_m()

```
const double& osse::collaborate::Vector::z_m ( ) const  [inline]
```

Get z_value (meters)

Returns

_z_m Z value (meters)

The documentation for this class was generated from the following files:

- libs/collaborate/include/collaborate/vector.h
- libs/collaborate/src/vector.cpp

2.49 osse::collaborate::Channel::LogBuffer Struct Reference

A buffer for logged node data.

`#include <channel.h>`

Data Fields

- std::vector< uint64_t > ticks

  Elapsed ticks.
- std::vector< int > year

  Year.
- std::vector< int > month

  Month.
- std::vector< int > day

  Day.
- std::vector< int > hour

  Hour.
- std::vector< int > minute

  Minute.
- std::vector< int > second

  Second.
- std::vector< int > microsecond

  Microsecond.
- std::vector< double > los_speed

  Line of sight speed.
- std::vector< double > omega

  Frequency.
- std::vector< double > distance

  Distance.
- std::vector< double > delay

  Delay.
- std::vector< double > data_rate

  Data Rate.
- std::vector< double > tx_idx

  TX Index.
- std::vector< uint64_t > tx_buffer

  TX Buffer.
- std::vector< double > tx_lon

  TX Longitude.
- std::vector< double > tx_lat

  TX Latitude.
- std::vector< double > tx_alt

  TX Altitude.
- std::vector< double > tx_gain

  TX Gain.
- std::vector< double > tx_power

TX Power.

- std::vector< double > rx_idx

  RX Index.
- std::vector< uint64_t > rx_buffer

  RX Buffer.
- std::vector< double > rx_lon

  RX Longitude.
- std::vector< double > rx_lat

  RX Latitude.
- std::vector< double > rx_alt

  RX Altitude.
- std::vector< double > rx_gain

  RX Gain.
- std::vector< double > rx_power

  RX Power.

2.49.1 Detailed Description

A buffer for logged node data.

The documentation for this struct was generated from the following file:

- libs/collaborate/include/collaborate/channel.h

2.50 osse::collaborate::Node::LogBuffer Struct Reference

A buffer for logged node data.

`#include <node.h>`

Data Fields

- int counter

  Index.
- uint16_t index [kLogBufferSize]

  Index.
- uint16_t constellation [kLogBufferSize]

  Constellation.
- uint64_t mode [kLogBufferSize]

  Operation mode.

- double latitude [kLogBufferSize]
  Latitude (degrees)
- double longitude [kLogBufferSize]
  Longitude (degrees)
- double energy [kLogBufferSize]
  Battery energy stored.
- bool charging [kLogBufferSize]
  Charging status.
- double area [kLogBufferSize]
  Solar panel effective area.
- uint16_t num_neighbors [kLogBufferSize]
  Number of neighbors.

2.50.1 Detailed Description

A buffer for logged node data.

The documentation for this struct was generated from the following file:

- libs/collaborate/include/collaborate/node.h

2.51 osse::collaborate::Node::PartialLog Struct Reference

A geodetic log for a node.

`#include <node.h>`

Data Fields

- uint16_t index_
  Index.
- uint16_t constellation_
  Constellation.
- uint64_t mode_
  Operation mode.
- double latitude_
  Latitude (degrees)
- double longitude_
  Longitude (degrees)
- double energy_

Battery energy stored.

- bool charging_

  Charging status.

- double area_

  Solar panel effective area.

- uint16_t num_neighbors_

  Number of neighbors.

2.51.1 Detailed Description

A geodetic log for a node.

The documentation for this struct was generated from the following file:

- libs/collaborate/include/collaborate/node.h

2.52 osse::collaborate::SimulationClock::LogBuffer Struct Reference

Log buffer.

`#include <simulation_clock.h>`

Data Fields

- int counter

  Counter.

- int year [kLogBufferSize]

  Year.

- int month [kLogBufferSize]

  Month.

- int day [kLogBufferSize]

  Day.

- int hour [kLogBufferSize]

  Hour.

- int minute [kLogBufferSize]

  Minute.

- int second [kLogBufferSize]

  Second.

- int microsecond [kLogBufferSize]

  Microsecond.

2.52.1 Detailed Description

Log buffer.

The documentation for this struct was generated from the following file:

- libs/collaborate/include/collaborate/simulation_clock.h

2.53 osse::collaborate::SubsystemComm::CommunicationEvent Struct Reference

A plan to transfer a packet to another node.

`#include <subsystem_comm.h>`

Data Fields

- uint16_t index_

  Index of the receiver node.
- uint64_t elapsed_s_

  Total earliest elapsed time when transfer should begin (seconds)
- PacketForward packet_

  Packet to transfer.

2.53.1 Detailed Description

A plan to transfer a packet to another node.

The documentation for this struct was generated from the following file:

- libs/collaborate/include/collaborate/subsystem_comm.h

2.54 osse::collaborate::SubsystemComm::FeedbackEvent Struct Reference

A plan to feedback a packet to the original node.

`#include <subsystem_comm.h>`

Data Fields

- uint16_t index_
  Index of the receiver node.
- uint64_t elapsed_s_
  Total earliest elapsed time when transfer should begin (seconds)
- PacketReturn packet_
  Packet to transfer.

2.54.1 Detailed Description

A plan to feedback a packet to the original node.

The documentation for this struct was generated from the following file:

- libs/collaborate/include/collaborate/subsystem_comm.h

2.55 osse::collaborate::SubsystemSensing::LogBuffer Struct Reference

A buffer for logged node data.

`#include <subsystem_sensing.h>`

Data Fields

- std::vector< uint64_t > elapsed_s
  Elapsed time (seconds)
- std::vector< int > year
  Year.
- std::vector< int > month
  Month.
- std::vector< int > day
  Day.
- std::vector< int > hour
  Hour.
- std::vector< int > minute
  Minute.
- std::vector< int > second
  Second.

- std::vector< int > microsecond
  Microsecond.
- std::vector< double > latitude_rad
  Latitude (radians)
- std::vector< double > longitude_rad
  Longitude (radians)
- std::vector< double > altitude_m
  Altitude (meters)
- std::vector< double > measurement
  Meaurement.
- std::vector< double > resolution_m
  Measurement resolution (m)
- std::vector< uint16_t > index
  Node index.

2.55.1 Detailed Description

A buffer for logged node data.

The documentation for this struct was generated from the following file:

- libs/collaborate/include/collaborate/subsystem_sensing.h

2.56 osse::collaborate::Tree::Branch Struct Reference

A node of the tree.

`#include <tree.h>`

Data Fields

- Branch * parent
  Branch directly above this one.
- std::vector< Branch * > children
  A list of branches originating from this one.
- uint16_t level
  Level of the current branch in the tree.
- Node * identity
  Corresponding satellite node.
- uint64_t rx_time_s
  Message received time (seconds)

2.56.1 Detailed Description

A node of the tree.

The documentation for this struct was generated from the following file:

- libs/collaborate/include/collaborate/tree.h