IS4S

# PNT Integrity Documentation

v1.16.1

Generated by Doxygen: August 17, 2021

# Contents

# Chapter 1

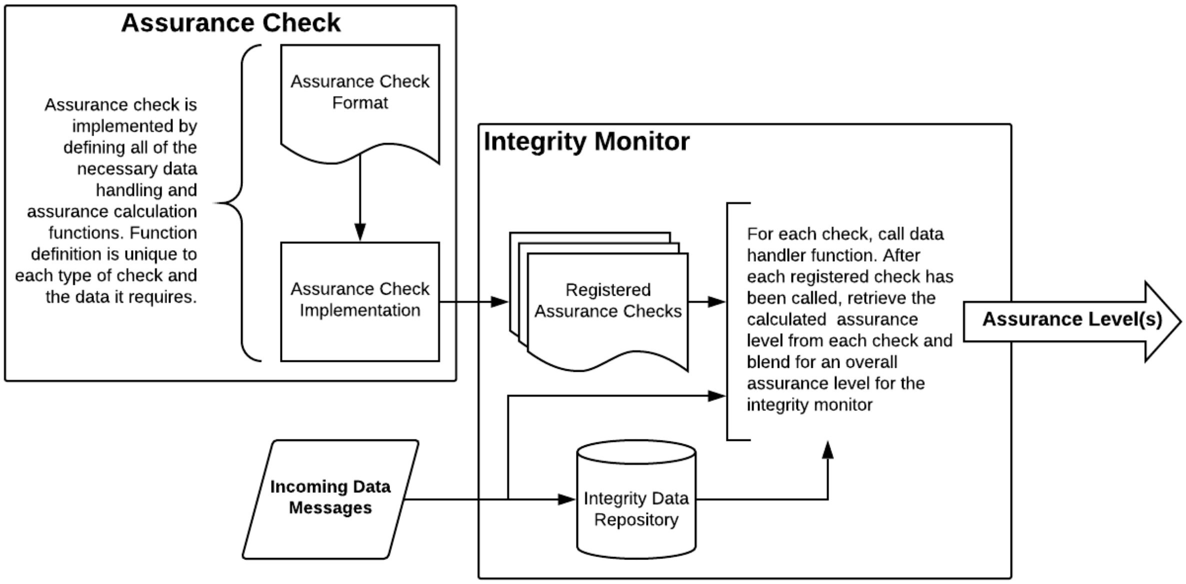
# The PNT-Integrity Library

The PNT Integrity Library provides a scalable framework for GNSS-based PNT manipulation detection that offers varying levels of protection based on the available data. The software is to be provided to GNSS receiver and GNSS-based timing server OEMs for use in future development or integration into existing products and platforms.

The modular nature of the application allows additional checks to be added as new threats arise. It also allows for the future addition of network-based data to further improve integrity.

## Framework Overview

The PNT Integrity Library can be used out-of-the-box with existing, built-in integrity checks. The framework also allows additional, user-defined integrity check algorithms to be incorporated into the application. The figure below gives a high-level description of the framework and how user-defined modules can be included.


Assurance Check
Assurance check is implemented by defining all of the necessary data handling and assurance calculation functions. Function definition is unique to each type of check and the data it requires.
Assurance Check Format
Assurance Check Implementation
Integrity Monitor
Registered Assurance Checks
For each check, call data handler function. After each registered check has been called, retrieve the calculated assurance level from each check and blend for an overall assurance level for the integrity monitor
Assurance Level(s)
Incoming Data Messages
Integrity Data Repository

The initial release of this Library has the following built-in assurance / integrity checks:

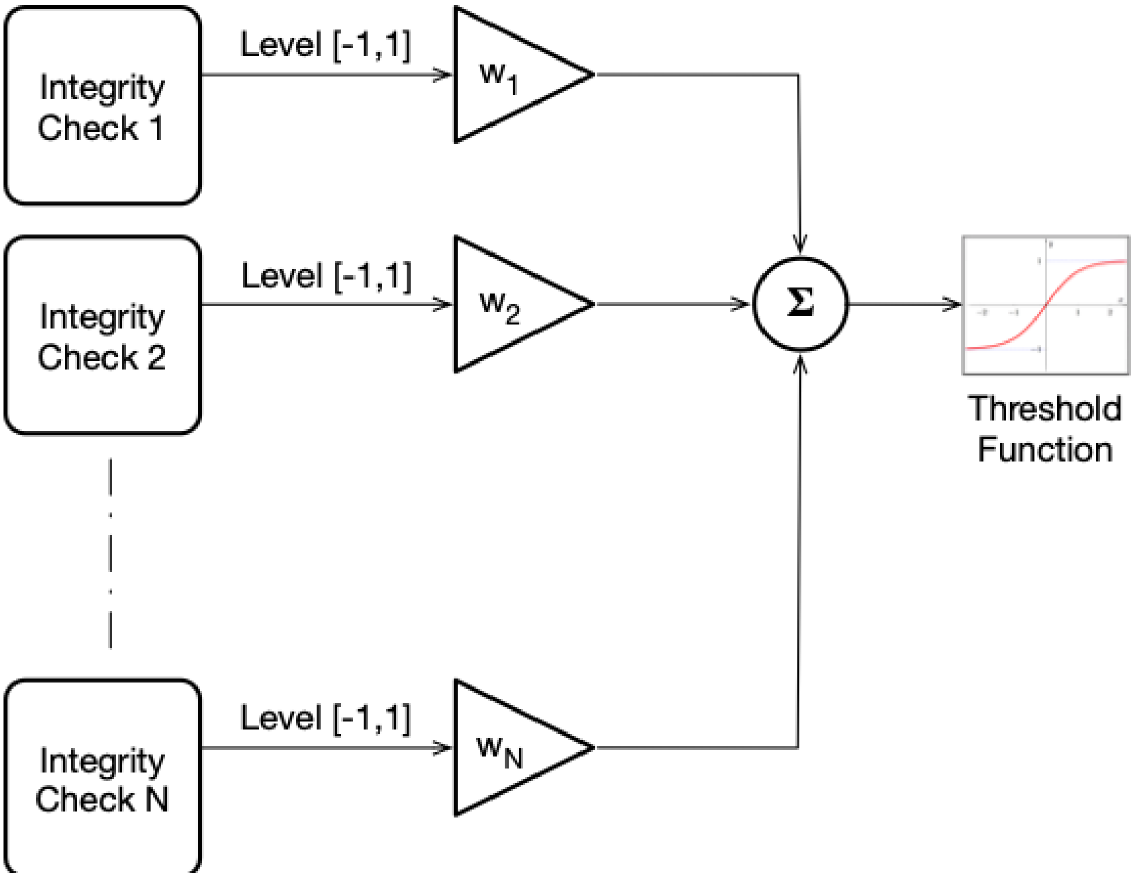
- Multi-Antenna / Receiver Detection Methods
  - Angle of Arrival (AOA)
  - Range / Position Verification (for networked receivers)
- Signal-Power and Vestigial-Peak Detection Methods
  - Spoofer signature detection in code / doppler 2D search
  - Automatic Gain Control (AGC) monitoring
- Model-based Consistency Checks
  - PNT discontinuities (time / position jumps)
  - Consistency checks (Carrier-to-noise, position/velocity)

## System Components

Each of the following sections describe each component of the framework in more detail.

### Blending Functions

The PNT-Integrity Library provides functionality for monitoring the integrity of single or multiple receiver outputs. It provides a confidence level on the receiver PVT solution(s) using a modular set of integrity checks based on receiver outputs and observables. The output of each individual integrity check is weighted based on the effectiveness of that check. The weighted outputs of all checks are then summed, and a threshold applied to determine a confidence level in the PVT solution as shown below.



Each individual integrity check returns an assurance level with a n associated numeric value for blending with other checks for a total level. This value, $l_i$, will be either -1, 0, or 1. The table below defines the levels and their values. Note that the first assurance level, 'Unavailable', does not provide a value and therefore will not be used when calculating a total assurance level.

| Level | Description | Enumeration | Value |
|---|---|---|---|
| Unavailable | Assurance level is unavailable (insufficient data or has not yet been run) | 0 | N/A |
| Unassured | Indicates a high likelihood that the measurement / source cannot be trusted | 1 | -1 |
| Inconsistent | Cannot reliably determine the validity of the measurement / source | 2 | 0 |
| Assured | Indicates a high likelihood that measurement / source can be trusted | 3 | 1 |

A weight, $w_i$, is assigned to each check to indicate the relative accuracy in determining the integrity level. The weighted sum (L') of these N individual levels $l_i$ is then calculated:

$$L' = \frac{\sum_{i=1}^{N} w_i l_i}{\sum_{i=1}^{N} w_i}$$

*Note: Weights must be positive and cannot all be 0.*

Weights are normalized for faster implementation:

$$w'_i = \frac{w_i}{\sum_{j=1}^{N} w_j}$$

$$L' = \sum_{i=1}^{N} w'_i l_i$$

The resulting level, L', is then thresholded to determine the overall integrity output L, which is then mapped to an overall assurance level using the values given in the table above.

$$L = \begin{cases} -1, & L' < threshold_{low} \\ 0, & threshold_{low} \leq L' < threshold_{hi} \\ 1, & L' \geq threshold_{hi} \end{cases}$$

For a simple rounding scheme, use the following thresholds:

$$threshold_{low} = -0.5$$
$$threshold_{hi} = 0.5$$

#### Positive Weighting Exception

In some situations it may be desirable to only weight certain assurance checks in the negative direction (i.e. when the check is attempting to lower the overall assurance level). For example, if a certain check should only be used to lower the overall assurance level and not keep it raised. Each assurance check has an internal flag used to indicate whether or not it allows positive weighting. Currently, this flag is hard-coded for each check Future versions of the software will allow this to be an input parameter. The table below shows which checks allow positive weighting and which do not.

#### Assurance Check

The "AssuranceCheck" module is a virtual object within the framework, meaning that it cannot be directly used, but rather a specific implementation must inherent its interface in order to be incorporated into the integrity monitor. As previously mentioned, several existing assurance check definitions are included out-of-the-box with the framework. An integrated application must either define a new/custom assurance check(s) or use existing, pre-defined check(s). Refer to the included software documentation for details on how to implement a user-defined check. The included example application can be referenced on how to incorporate, initialize, and use the built-in checks.

The Assurance Check base class contains all data and functions that are common to every assurance check child derivative class. As an example, the parent class contains a setting known as the assurance level period. This setting defines how long each check must hold a lowered assurance level before it can be raised again. If a child check detects that the level should be lowered to indicate an attack, this level change is allowed immediately. However, if the child check decides that the attack condition is no longer present, the check must hold previously lowered value for this pre-determined amount of time. This single-sided hysteresis is intended to prevent level "flickering."

### Integrity Monitor

The "IntegrityMonitor" module is the primary component that the user application will interface with. All assurance checks, both user-defined and built-in, must be registered with the integrity monitor, which will keep a vector of all registered checks. The enclosing application will then pass all received data messages to the integrity monitor for processing as they are received. The monitor will cycle through all registered checks, calling the appropriate data processing function in each check. After all checks have been called, the monitor will then extract the calculated assurance level from each check and blend them for an overall result.

### Integrity Data Repository

A time history of received integrity data is available for use by the built-in and user-defined assurance checks. The repository has a time-history length that can be controlled with a setting. Repository use is not required for user-defined checks, but it is accessible if desired. See the attached documentation on how to utilize the repository.

### Data Structures

Refer to the software documentation in later chapters for details on the data structures used in the framework.

## Included Assurance Checks

The table below lists the out-of-the-box assurance checks that are packaged with the library. The table shows which checks allow positive weighting, the class of check, and the resilience level associated with each check. A definition of the resilience levels and how they apply to a PNT system can be found [here](link).

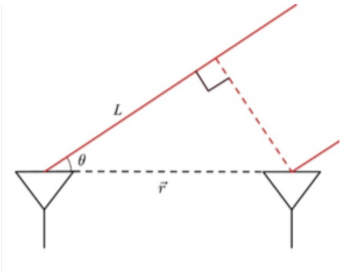
| Check Name | Allows Positive Weighting | Assurance Check Class | Resilience Level |
|---|---|---|---|
| Angle-of-Arrival (AOA) Check | Yes | Multi-Antenna / Node | 2 |
| Range-Position Check | Yes | Multi-Antenna / Node | 1 |
| Automatic Gain Control (A↩GC) Check | Yes | Signal Power and Peak Detection (CAF) | 2 |
| Acquisition Check | Yes | Signal Power and Peak Detection (CAF) | 3 |
| Static-Position Check | Yes | Model-Based Consistency | 1 |
| Position-Jump Check | Yes | Model-Based Consistency | 1 |
| Position-Velocity Consistency Check | Yes | Model-Based Consistency | 1 |
| Clock-Jump Check | Yes | Model-Based Consistency | 2 |
| Carrier-to-Noise (CNo) Check | No | Model-Based Consistency | 2 |

## Multi-Antenna Detection Algorithms

Multiple receivers / antennas connected a system can be leveraged to form power assurance checks. Two such checks included with the PNT Integrity Library are described in the following sub-sections.

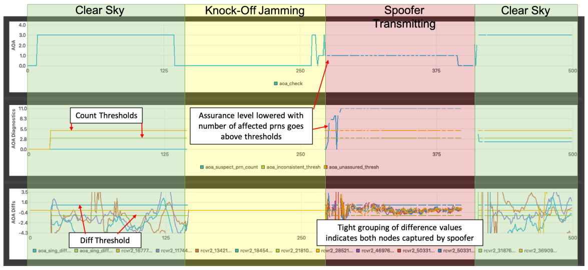
### The Angle of Arrival Check

The most effective method for detecting GPS spoofing is using angle of arrival (AOA) with multiple receiver antennas. This method can be implemented using a single receiver with multiple antennas or multiple independent receivers with an available network connection for sharing data. Relative pseudorange or carrier phase from each antenna in an array is a function of the angle of the arriving signal, as shown in the figure below.



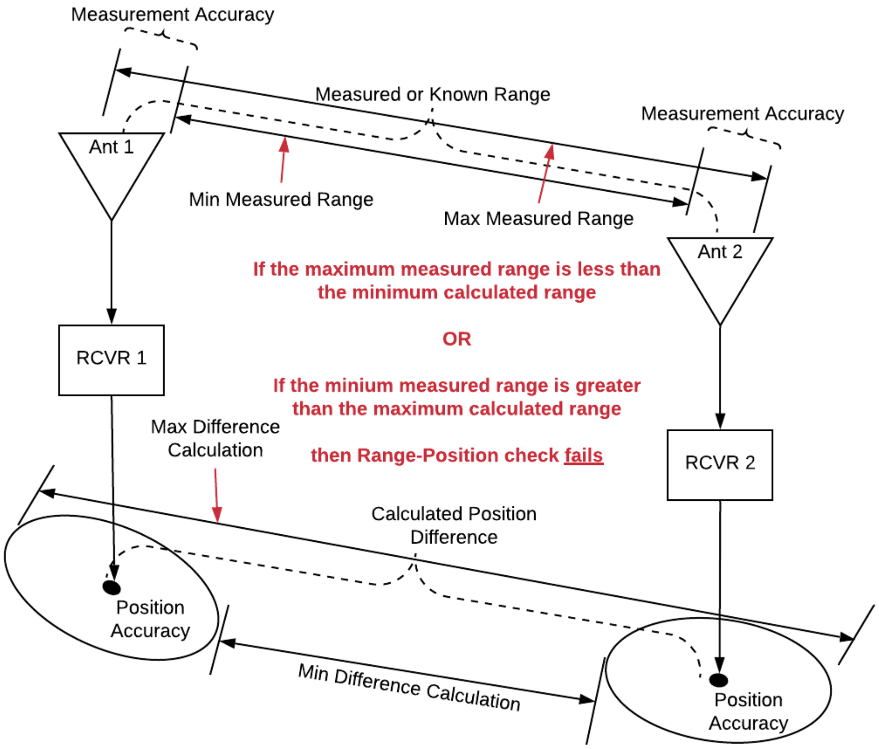
To determine the AOA, pseudorange or carrier phase measurements from two antennas are subtracted to produce a single differenced observable. The AOA is dependent on the satellite position in the sky (and platform attitude). As a result, the phase differences will vary from channel to channel. However, during a spoofing attack, the AOA of every captured channel will converge to the same value as all signals will be propagating from the same source.

The figure below demonstrates the calculations and logic used inside the AOA check. Single differences are computed between common prn psuedoranges (or carrier-phases) between two communicating nodes (node = receiver + antenna). Under clear-sky conditions, the difference values are separated (providing that the receiving antennas are separated sufficiently). When both nodes become captured by the spoofer, the difference values collapse to a tight grouping. The algorithm counts the number of prns that are within the difference thresholds and sets the level according to a separate set of thresholds (count thresholds).



### The Range-Position Check

If the distance between two antennas is known (or measured), a simple but effective assurance check can be achieved by comparing the measured range to the differenced position of the two antennas. A differenced position is computed by taking the absolute value of the position of receiver A minus the position of receiver B. This computed range is then compared to the measured range between the two antennas. Taking position and range measurement variances into account, the two are compared to see if the difference is within reasonable tolerances. If either receiver (or both) is captured by the spoofer, then this check can be a reliable indicator of a position-based spoofing attack. Obviously, this check will not be reliable when both antennas are close together, as the range measurement will not be large enough to invalidate a position difference, regardless of spoofer effectiveness.


Measurement Accuracy
Measured or Known Range
Measurement Accuracy
Ant 1
Min Measured Range
Max Measured Range
Ant 2
If the maximum measured range is less than the minimum calculated range
OR
If the minium measured range is greater than the maximum calculated range
then Range-Position check fails
RCVR 1
RCVR 2
Max Difference Calculation
Calculated Position Difference
Position Accuracy
Min Difference Calculation
Position Accuracy

## Signal Power and Vestigial Peak Detection Algorithms

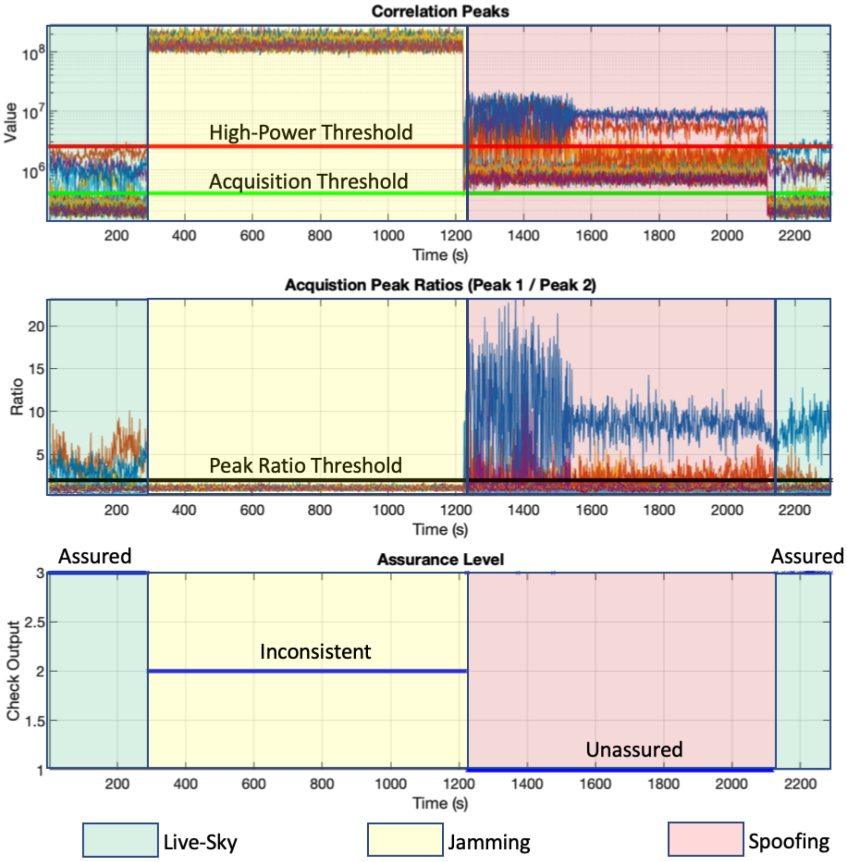
### Acquisition Check

Often a jammer will be used in conjunction with a spoofer in order to raise the noise floor and hide the authentic signal. Other times the spoofer will transmit at much higher power levels than the live-sky signal in order to cover a large area or due to mis-calibration (a very difficult process). One straightforward way of detecting spoofers is by closely monitoring the power levels of the signals in the acquisition process. Despite the fact that the GPS signal transmission was designed to provide roughly constant power from horizon to horizon, there are still power level differences of three to six decibels from horizon to zenith. By knowing the possible power change across the satellites path, the amount of amplification coming from the antenna, amplification from low noise amplifiers, and using previously known correlation values, a range of possible correlation values can be determined and set as a threshold. Should any correlation value be higher than the threshold, it is likely from an attack and can be quickly detected.

The acquisition check currently implemented in this library takes IF data snippets from a receiver front end, runs a signal acquisition process, and then monitors the correlator outputs for suspect power levels . Each tracking channel (up to 32 for GPS L1) is assessed independently and then aggregated to form a composite assurance level for the check. For each channel, the highest two peaks in the signal correlation plane are selected. If the highest peak is above a threshold, then this channel is considered to be suspect. If PRN is considered to be acquired in this high-power state (i.e. when the ratio of peak1 to peak 2 is above the threshold), then it is flagged as "unassured" to indicate a possible attack. If it is not acquired in this high-power state, then it is flagged as "inconsistent" to indicate possible jamming. If the peak 1 value is below the high-power threshold, but still above the acquisition threshold, then that PRN is considered to be acquired in safe state and is acquired as "assured". Otherwise, the PRN is flagged as "unavailable" (i.e. the satellite is not in view). The number of PRNs in each state are summed and passed through the following logic to determine the assurance level for any given time.

```
if (unassuredCount >= assuranceUnassuredThresh_)
{
    level = UNASSURED
}
else if (inconsistentCount >= assuranceInconsistentThresh_)
{
    level = INCONSISTENT
}
else if (assuredCount >= assuredThresh)
{
    level = ASSURED
}
else
{
    level = UNAVAILABLE
}
```

The figure below shows output data from the Acquisition Check during a "knock-off-then-spoof" attack. The check is tuned to produce an overall level of "Inconsistent" in a pure-jamming environment and "Unassured" when it is being spoofed.



### Automatic-Gain Control (AGC) Check

A relatively simple check for a jamming / spoofing attack can be achieved by monitoring the output of the receiver's automatic gain control (AGC). An AGC is a common component in any radio device, and attempts to regulate the incoming signal to a desired level to optimizes the downstream signal processing. If the incoming power level is high, the AGC will lower its gain so that the incoming signal will not saturate. Conversely, if the incoming power level is low, the AGC will increase its gain to boost the signal for better signal processing and data demodulation. By simply monitoring the current AGC setting of the receiver (provided that it is available), a user (or detection software) can gain a good sense of what might be happening in the signal environment.

In this library, the AGC check implementation simply monitors the AGC setting (in all available bands), normalizes its current value, and compares it to a threshold to indicate attack. To operate this check, the minimum and maximum setting values for the AGC must be known. In addition, this check is currently tuned so that the assurance level of the check is only raised to "Inconsistent." On its own, the AGC check cannot discern the difference between a jamming or spoofing attack. In future versions of this library, the AGC check and acquisition checks will be integrated together to form a complete picture of what is happening in the signal environment.

## Model-based Consistency Checks

Another effective approach in developing assurance checks is to analyze output data from the receiver (solution(s), measurements, and raw observables) and compare to known behavior. This grouping of checks is labeled "model-based" checks, as they aim to perform sanity reference checks of available data to known models.

### Static-Position Check

For PNT applications where system receiver remains static (cell towers, power stations, financial centers, etc), a static position check be employed for a simple check against attacks. The check can be provided with a surveyed position of the receiver's antenna to compare with the solution that is being published by the receiver. If the difference is greater than a configurable threshold, then the check will attempt to lower the assurance level. The check also has the capability to perform an initial survey at startup, with the assumption that things are started in a "safe" environment.

### Position-Jump Check

The position jump check is an advanced extension of the static-position check. The receiver's position solution is monitored and compared to a secondary source of the platform position and covariance. If the receiver's position travels outside of the bounds of the secondary source, then the assurance level is lowered. Additionally, for systems that do not have a separate position measurement available, a maximum platform velocity can be used to propagate the error bounds by using a "last known good position" and a maximum distance traveled since that time.

### Position-Velocity Consistency Check

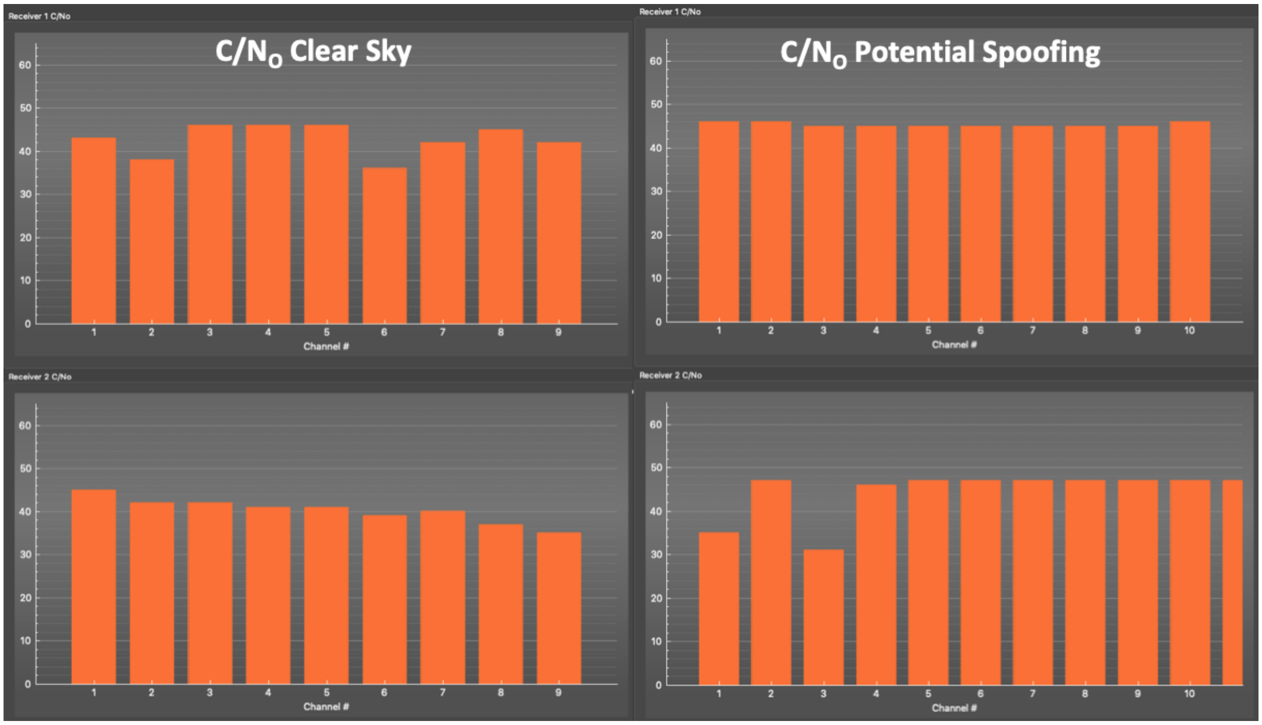
Another model-based check is available by comparing the consistency between the position and velocity measurements out of a receiver. A pseudo-velocity measurement can be created by differencing the position measurements over time. If these measurements are not in agreement with the velocity measurement (within a threshold / bound), then the assurance level is lowered.

### Clock-Jump Check

Another model-based check examines the clock bias and drift for normal behavior. The Clock Bias Check calculates the expectation and variance of the clock drift for the most recent set of clock samples, minus the most recent sample. The expectation is used to propagate the clock forward to the most recent single sample's arrival time and check if it is within reasonable bounds. The variance is used to check for zero-bias disruption.

### Carrier-to-noise (CNo) Consistency Check

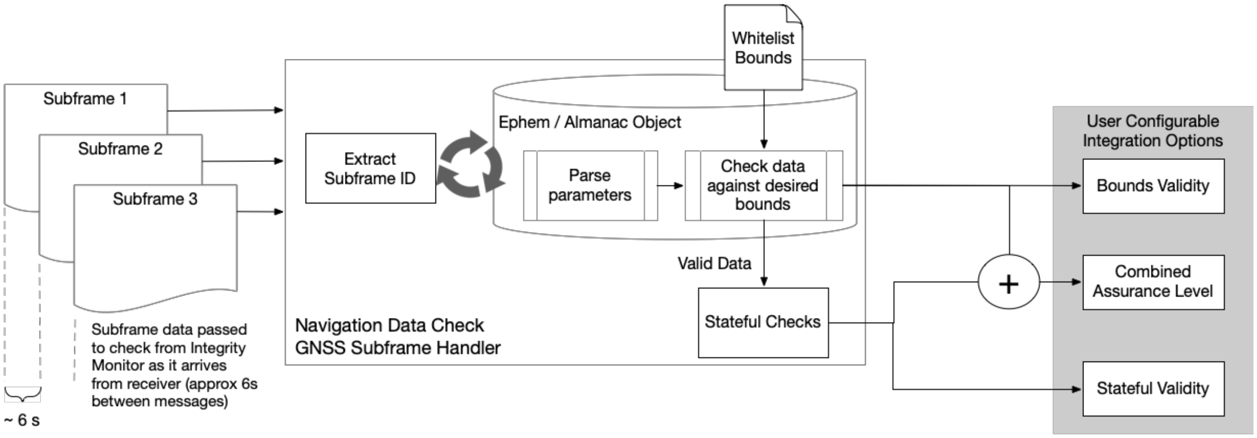
This check is often effective in detecting a code-generating spoofing attack. In live sky signals, observed C/No values have significant variation due to differences in SV elevation, signal obstructions, multi-path, etc. During simulator-based spoofing attack, all spoofed signals may be transmitted at the same C/No level. This check detects this artifact by monitoring the distribution of observed signal C/No's.



## Navigation Data Monitoring

The PNT Integrity Library includes monitoring of navigation data output available from a GPS receiver. Many COTS receivers typically output raw subframe data received from each satellite vehicle. The library has the ability to monitor these subframes for malformed data. By default, the library uses constraints codified in a whitelist defined by the IS-GPS-200H specification. Source code can be modified to adjust these constraints to end-users' desires.

The diagram below shows the navigation data monitoring process as implemented in the PNT Integrity Library. The Navigation data check is structured like the other checks in the library, with common inheritance from the base AssuranceCheck class and handlers for all observables. The Nav-Data Check only responds to GNSS subframe messages. Subframes should arrive the in the receive on 6 second intervals. The subframe handler function in the nav-data check extracts the subframe ID from the received raw data and parses accordingly. An ephemeris (or almanac object) is populated as subframes arrive. Each parsing call checks the data for validity bounds. When valid data is received, it is pass along to "stateful" checks that check consistency of certain parameters over time. Validity outputs from both the bounds checks and the stateful checks are available for downstream applications.


Subframe 1
Subframe 2
Subframe 3
Subframe data passed to check from Integrity Monitor as it arrives from receiver (approx 6s between messages)
~ 6 s
Extract Subframe ID
Whitelist Bounds
Ephem / Almanac Object
Parse parameters
Check data against desired bounds
Valid Data
Stateful Checks
Navigation Data Check GNSS Subframe Handler
+
User Configurable Integration Options
Bounds Validity
Combined Assurance Level
Stateful Validity

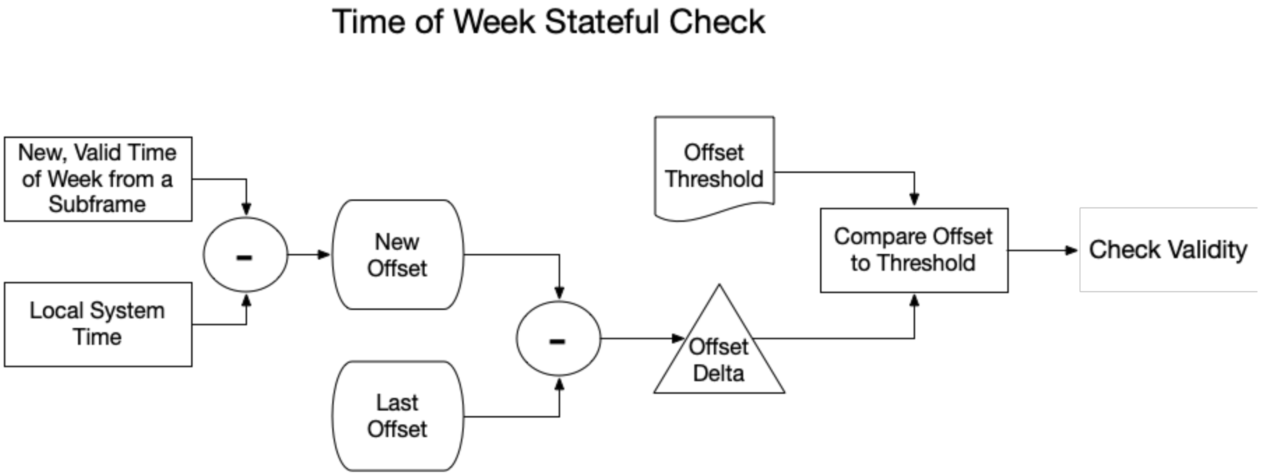
The check also produces a combined assurance level based on the combined output of the bounds validity checks and the stateful checks. In the "out-of-the-box" configuration, the Integrity Monitor object simply logical OR's all other validity flags together. A logic high results in assurance level elevation to the WARNING state.

Two stateful checks are currently implemented in the library. A time of week consistency check and a week number consistency check are described in the following sections.

### Time of Week Stateful Check

The time of week stateful check is responsible for verifying the proper behavior of the GPS time of week parameter. The time of week parameter from GPS subframes is an integer that increments by the amount of time between the arrival between each subframe (6 seconds), starting at 0 and advancing through the week to a value of 604794, then rolls over to 0 on the start of a new GPS week. The check must take into account that a receiver may eventually miss a subframe, meaning that the check cannot simply check for an increase of 6 for each subframe.

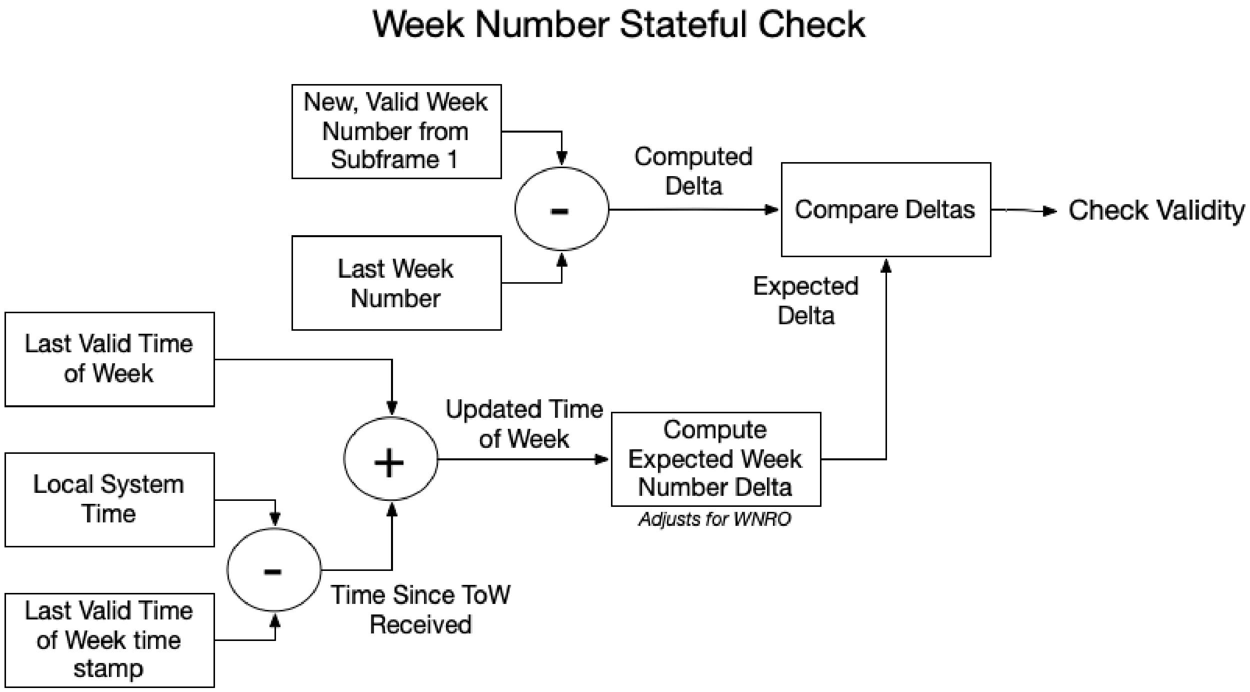
The check accomplishes by calculating an offset between the week number and the local system time (PC clock). This offset should change very little over the course of time. A newly computed offset is compared to the last offset calculated with valid data. If the difference is greater than a threshold, the time of week is considered to be invalid.


Time of Week Stateful Check
New, Valid Time of Week from a Subframe
Local System Time
-
New Offset
Last Offset
-
Offset Threshold
Offset Delta
Compare Offset to Threshold
Check Validity

The implementation of this check assumes that the local system time will drift vary slowly over time. This is true particularly if the local system is synchronized to a stable reference source (NTP, PTP, GPS, etc).

### Week Number Stateful Check

The library also includes a stateful check of the GPS week number. When a new week number is received, the algorithm computes a delta between the new value and the last valid week number received. This delta is compared to an expected delta, which is computed based on the elapsed system time since the last valid time of week was received. If the deltas match, then the new week number is considered to be valid. The expected week number computation adjusts for potential week number rollover (WNRO). The week number check algorithm is shown in the figure below.



# Chapter 2

# License

This library is licensed under the BSD 3-Clause License. The library contains source code developed by IS4S and third parties. Refer to the invidiual source files for applicable copyright information.

Copyright (c) 2020 Integrated Solutions for Systems (IS4S), Inc Copyright (c) 2017, ETHZ ASL (geodetic converter)

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of IS4S or ETHZ ASL nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

# Chapter 3

# Namespace Index

## 3.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

# Chapter 4

# Hierarchical Index

## 4.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

# Chapter 5

# Class Index

## 5.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

| | | |
|---|---|---|
| pnt_integrity::data::AccumulatedDistranceTraveled | A structure that represents a distance traveled over a time period | 51 |
| pnt_integrity::AcqCheckDiagnostics | Structure for publishing Acquisition Check diagnostics | 52 |
| pnt_integrity::AcquisitionCheck | Class implementation for the acquisition check | 52 |
| pnt_integrity::AgcCheck | Class implementation for the AGC check | 57 |
| pnt_integrity::AgcCheckDiagnostics | Diagnostic data for AGC check | 60 |
| pnt_integrity::data::AgcValue | A structure to represent an AGC measurement | 60 |
| pnt_integrity::AlmanacParameters | | 61 |
| pnt_integrity::AlmanacSubframeFaults | | 62 |
| pnt_integrity::AngleOfArrivalCheck | Class implementation for the angle of arrival check | 62 |
| pnt_integrity::AoaCheckDiagnostics | Structure used to publish diagnostic data | 68 |
| pnt_integrity::AssuranceCheck | Parent class for all integrity checks | 69 |
| pnt_integrity::data::AssuranceReport | A structure to hold a single assurance report | 83 |
| pnt_integrity::data::AssuranceReports | A structure to hold assurance data for all registered checks | 84 |
| pnt_integrity::data::AssuranceState | A structure to hold an AssuranceLevel and value | 85 |
| pnt_integrity::ClockBiasCheck | Class implementation for the position velocity check | 88 |
| pnt_integrity::ClockBiasCheckDiagnostics | Structure used to publish diagnostic data | 91 |
| pnt_integrity::data::ClockOffset | A structure for measuring the offset between two clocks | 92 |

# Chapter 6

# File Index

## 6.1 File List

Here is a list of all documented files with brief descriptions:

# Chapter 7

# Namespace Documentation

## 7.1 geodetic_converter Namespace Reference

Third-party. Downloaded from: `https://github.com/ethz-asl/geodetic_utils` //.

### Classes

- class GeodeticConverter

  *Class to implement gedetic conversions for the pnt_integrity library.*

### Variables

- const double kSemimajorAxis = 6378137

  *Equatorial radius (a), in meters.*
- const double kSemiminorAxis = 6356752.3142

  *Semi-minor radius (b), in meters.*
- const double kFirstEccentricitySquared = 6.69437999014 ∗ 0.001

  *First eccentricity squared (e2), dimensionless e2 = (a^2 - b^2) / a^2 = f ∗ (2 - f)*
- const double kSecondEccentricitySquared = 6.73949674228 ∗ 0.001

  *Second eccentricity squared (e'2), dimensionless e'2 = (a^2 - b^2) / b^2 = e^2 / (1 - e^2) = e2 / (1 - e2)*
- const double kFlattening = 1 / 298.257223563

  *flattening, dimensionless*
- const double PI = 3.14159265358979323846

  *Pi (pi), dimensionless.*

### 7.1.1 Detailed Description

Third-party. Downloaded from: `https://github.com/ethz-asl/geodetic_utils` //.

## 7.2 pnt_integrity Namespace Reference

Namespace for all pnt_integrity applications.

### Namespaces

- data

  *Namespace for all integrity data definitions.*

### Classes

- struct AcqCheckDiagnostics

  *Structure for publishing Acquisition Check diagnostics.*
- class AcquisitionCheck

  *Class implementation for the acquisition check.*
- class AgcCheck

  *Class implementation for the AGC check.*
- struct AgcCheckDiagnostics

  *Diagnostic data for AGC check.*
- struct AlmanacParameters
- union AlmanacSubframeFaults
- class AngleOfArrivalCheck

  *Class implementation for the angle of arrival check.*
- struct AoaCheckDiagnostics

  *Structure used to publish diagnostic data.*
- class AssuranceCheck

  *Parent class for all integrity checks.*
- class ClockBiasCheck

  *Class implementation for the position velocity check.*
- struct ClockBiasCheckDiagnostics

  *Structure used to publish diagnostic data.*
- class CnoCheck

  *Class implementation of the carrier-to-noise (CnO) assurance check. The check analyzes the CnO values for abnormalities.*
- struct CnoCheckDiagnostics

  *Diagnostic data for the check.*
- struct EphemerisParameters
- class GpsAlmanac

  *Class to parse and store almanac data for a GPS Satellite.*
- class GpsEphemeris
- class IntegrityDataRepository

  *Class definition for the history of data at a single PNT node.*
- class IntegrityMonitor

  *Class implementation of integrity monitoring using AssuranceChecks and IntegrityData.*
- struct NavDataCheckDiagnostics

  *Structure for check diagnostics.*

- class NavigationDataCheck

  *Class implementation for the navigation data check.*
- class PositionJumpCheck

  *Class implementation for the position-jump check.*
- class PositionVelocityConsistencyCheck

  *Class implementation for the position velocity check.*
- struct PosJumpCheckDiagnostics

  *Structure for check diagnostics.*
- struct PosVelConsCheckDiagnostics

  *Structure used to publish diagnostic data.*
- class RangePositionCheck

  *Class implementation for the range / position check.*
- class RepositoryEntry

  *Class definition for an entry into the repository.*
- struct RngPosCheckNodeDiagnostic

  *Structure for check diagnostics.*
- struct StaticPosCheckDiagnostics

  *Structure used to publish diagnostic data.*
- class StaticPositionCheck

  *Class implementation for the static-position check.*
- struct Subframe1Fault
- struct Subframe2Fault
- struct Subframe3Fault
- struct SVAlmHealth
- struct SVHealth

  *Structure to hold the SV health status from subframe 1, word 3, bits 17-22.*
- struct TimeEntry

  *Structure for a time entry into the repository.*

## Typedefs

- using CodeMap = std::map< int, std::vector< float > >

  *A map for holding PRN codes, indexed on prn.*
- using CodeMapEntry = std::pair< int, std::vector< float > >

  *A pair for holding a PRN and it's code.*
- using CodeFreqMap = std::map< int, Eigen::ArrayXcf >

  *A map for holding frequency bin values.*
- using CodeFreqMapEntry = std::pair< int, Eigen::ArrayXcf >

  *A pair for holding a frequency bin number its values.*
- using CorrelationResultsMap = std::map< int, Eigen::ArrayXXf >

  *A map that stores the correlation results for a prn.*
- using PeakResultsMap = std::map< int, std::pair< double, double > >
- using PrnList = std::vector< int >

  *A vector type for a list of prns.*
- using SingleDiffMap = std::map< int, double >

  *Defines a type that maps PRN to a calculated difference.*
- using PrnAssuranceEachNode = std::map< int, std::vector< data::AssuranceLevel > >

*Defines a map that holds an assurance level for each prn for each node.*

- using MultiPrnAssuranceMap = std::map< int, data::AssuranceLevel >

  *A map for pairing an assurance level to each PRN.*
- using RemoteRepoEntries = std::map< std::string, RepositoryEntry >

  *A type to map remote entries to their node name / device id.*
- using TimeEntryHistory = std::map< double, TimeEntry >
- using AssuranceChecks = std::map< std::string, AssuranceCheck ∗ >

  *A vector type for a collection of AssuranceChecks.*
- using RngPosCheckDiagnostics = std::map< std::string, RngPosCheckNodeDiagnostic >

  *Defined type for check diagnostics.*

## Enumerations

- enum AoaCheckData { **UsePseudorange** = 0, **UseCarrierPhase**, **UseBoth** }

  *Enumeration to indicate what data field to use for the AOA check.*
- enum SVNavHealth : uint8_t {
  **AllDataOK** = 0, **ParityFailure** = 1, **TlmHowFormatProblem** = 2, **ZCountInHowBad** = 3,
  **Subframe_1_2_or_3_Bad** = 4, **Subframe_4_or_5_Bad** = 5, **AllUploadedDataBad** = 6, **AllDataBad** = 7 }
- enum AntiSpoofFlag { **Off** = 0, **On** = 1 }
- enum L2CodeType { **Reserved** = 0, **PCodeOn** = 1, **CACodeOn** = 2, **CAAndPCodeOn** = 3 }
- enum FitInterval { **FourHrs** = 0, **GreaterThanFourHrs** = 1 }
- enum AlertFlag { **ALERT_OFF** = 0, **ALERT_RAISED** = 1 }
- enum L2NavDataFlag { **On** = 0, **Off** = 1 }
- enum SVSignalHealth : uint8_t {
  **AllSignalsOk** = 0, **AllSignalsWeak** = 1, **AllSignalsDead** = 2, **AllSignalsHaveNoDataModulation** = 3,
  **L1PSignalWeak** = 4, **L1PSignalDead** = 5, **L1PSignalHasNoDataModulation** = 6, **L2PSignalWeak** = 7,
  **L2PSignalDead** = 8, **L2PSignalHasNoDataModulation** = 9, **L1CSignalWeak** = 10, **L1CSignalDead** = 11,
  **L1CSignalHasNoDataModulation** = 12, **L2CSignalWeak** = 13, **L2CSignalDead** = 14, **L2CSignalHasNoDataModulation** = 15,
  **L1AndL2PSignal_Weak** = 16, **L1AndL2PSignal_Dead** = 17, **L1AndL2PSignal_HasNoDataModulation** = 18,
  **L1AndL2CSignal_Weak** = 19,
  **L1AndL2CSignal_Dead** = 20, **L1AndL2CSignal_HasNoDataModulation** = 21, **L1SignalWeak** = 22, **L1SignalDead** = 23,
  **L1SignalHasNoDataModulation** = 24, **L2SignalWeak** = 25, **L2SignalDead** = 26, **L2SignalHasNoDataModulation** = 27,
  **SVIsTemporarilyOutDoNotUse** = 28, **SVWillBeTemporarilyOutUseWithCaution** = 29, **OneOrMoreSignalsDeforedURAStillValid** = 30, **MoreThanOneCombinationNeededToDescribeAnomalies** = 31 }
- enum NavDataTimeOfArrival { **Older**, **Same**, **Newer** }

  *Enumeration to define the relative time between multiple LNAV data sets.*
- enum DataLocaleType { **Local** = 0, **Remote** = 1 }

  *Defines the possible observable types.*

## Functions

- void fromHex (const std::string &in, void ∗const data)

  *Convert hexadecimal string to char array.*
- void toHex (unsigned char ∗const byteData, const size_t dataLength, std::string &dest)

  *Convert char array to hexadecimal string.*

- void convertSubframeFrom10To30Word (const uint32_t(&sfIn)[10], uint8_t(&sfOut)[30])

    *Converts uint32_t[10] subframe to uint8_t[30] array.*
- void convertSubframeFrom30To10Word (const uint8_t(&sfIn)[30], uint32_t(&sfOut)[10])

    *Converts uint8_t[30] subframe to uint32_t[10] array.*
- void removeSubframeParity (const uint32_t(&subframeWordsIn)[10], uint32_t(&subframeWordsOut)[10])

    *Remove parity bits from subframe.*
- uint16_t parseSubframeID (const uint8_t(&subframe)[30])

    *Parse a subframe and return its ID number.*
- void parseSubframeID (const uint8_t(&subframe)[30], uint16_t &subframeID)

    *Parse a subframe and return its ID number.*
- double parseTimeOfWeek (const uint8_t(&subframe)[30])

    *Parse a subframe and return the time of week.*
- void parseTimeOfWeek (const uint8_t(&subframe)[30], double &tow)

    *Parse a subframe and return the time of week.*

## Variables

- const std::string INTEGRITY_ACQ_PEAK_VALS = "INTEGRITY_ACQ_PEAK_VALS"

    *String ID for the ACQ check peak vals.*
- const std::string INTEGRITY_ACQ_PEAK1_KEY = "INT_ACQ_PEAK1_"

    *String ID for the ACQ check peak 1 key.*
- const std::string INTEGRITY_ACQ_PEAK2_KEY = "INT_ACQ_PEAK2_"

    *String ID for the ACQ check peak 2 key.*
- const std::string INTEGRITY_ACQ_DIAGNOSTICS = "INTEGRITY_ACQ_DIAGNOSTICS"

    *String ID for the ACQ check diagnostic data.*
- const std::string INT_ACQ_DIAG_HI_PWR_THRESH = "INT_ACQ_DIAG_HI_PWR_THRESH"

    *String ID for the ACQ check high power threshold.*
- const std::string INT_ACQ_DIAG_PEAK_RATIO_THRESH

    *String ID for the ACQ check peak ratio threshold.*
- const std::string INT_ACQ_DIAG_ACQ_THRESH = "INT_ACQ_DIAG_ACQ_THRESH"

    *String ID for the ACQ check acquisition threshold.*
- const std::string INT_ACQ_DIAG_ITHRESH = "INT_ACQ_DIAG_ITHRESH"

    *String ID for the ACQ check survey inconsistent thresh.*
- const std::string INT_ACQ_DIAG_UTHRESH = "INT_ACQ_DIAG_UTHRESH"

    *String ID for the ACQ check survey unassured thresh.*
- const std::string INT_ACQ_DIAG_ICOUNT = "INT_ACQ_DIAG_ICOUNT"

    *String ID for the ACQ check survey inconsistent count.*
- const std::string INT_ACQ_DIAG_UCOUNT = "INT_ACQ_DIAG_UCOUNT"

    *String ID for the ACQ check survey unassured count.*
- const std::string INT_ACQ_DIAG_PEAK_RATIO_KEY = "INT_ACQ_DIAG_PEAK_RATIO_KEY_"

    *String ID for the ACQ check survey peak ratio key.*
- const std::string INTEGRITY_AGC_DIAGNOSTICS = "INTEGRITY_AGC_DIAGNOSTICS"

    *String ID for the AGC check diagnostic data.*
- const std::string INTEGRITY_AGC_DIAG_ITHRESH = "INTEGRITY_AGC_DIAG_ITHRESH"

    *String ID for the AGC check survey inconsistent thresh.*
- const std::string INTEGRITY_AOA_DIFF_DIAGNOSTICS

*String ID for the AOA check difference diagnostic data.*

- const std::string INTEGRITY_AOA_DIFF_NODE_ID = "INTEGRITY_AOA_DIFF_NODE_ID"

  *String ID for the AOA check diagnostic node id.*
- const std::string INTEGRITY_AOA_DIAGNOSTICS = "INTEGRITY_AOA_DIAGNOSTICS"

  *String ID for the AOA check diagnostic data.*
- const std::string INTEGRITY_AOA_DIAG_DIFF_THRESH

  *String ID for the AOA check diagnostic difference threshold.*
- const std::string INTEGRITY_AOA_DIAG_SUSPECT_PRN_PERCENT

  *String ID for the AOA check diagnostic suspect prn percent.*
- const std::string INTEGRITY_AOA_DIAG_UNAVAILABLE_PRN_PERCENT

  *String ID for the AOA check diagnostic unavailable prn percent.*
- const std::string INTEGRITY_AOA_DIAG_ASSURED_PRN_PERCENT

  *String ID for the AOA check diagnostic assured prn percent.*
- const std::string INTEGRITY_AOA_DIAG_ITHRESH = "INTEGRITY_AOA_DIAG_ITHRESH"

  *String ID for the AOA check survey inconsistent thresh.*
- const std::string INTEGRITY_AOA_DIAG_UTHRESH = "INTEGRITY_AOA_DIAG_UTHRESH"

  *String ID for the AOA check survey unassured thresh.*
- const std::string INTEGRITY_AOA_DIAG_ATHRESH = "INTEGRITY_AOA_DIAG_ATHRESH"

  *String ID for the AOA check survey assured thresh.*
- const std::string INTEGRITY_CLOCK_BIAS_DIAGNOSTICS

  *String ID for the clock-bias check diagnostic data.*
- const std::string INTEGRITY_CLOCK_BIAS_DIAG_EXP_DRIFT

  *String ID for the clock-bias check expected drift.*
- const std::string INTEGRITY_CLOCK_BIAS_DIAG_EXP_DRIFT_VAR

  *String ID for the clock-bias check drift variance.*
- const std::string INTEGRITY_CLOCK_BIAS_DIAG_PROP_OFFSET

  *String ID for the clock-bias check propagation offset.*
- const std::string INTEGRITY_CLOCK_BIAS_DIAG_ACTUAL_OFFSET

  *String ID for the clock-bias check actual offset.*
- const std::string INTEGRITY_CLOCK_BIAS_DIAG_OFFSET_ERROR

  *String ID for the clock-bias check offset error.*
- const std::string INTEGRITY_CLOCK_BIAS_DIAG_DRIFT_RATE_BOUND

  *String ID for the clock-bias check drift rate bound.*
- const std::string INTEGRITY_CLOCK_BIAS_DIAG_DRIFT_RATE_VAR_BOUND

  *String ID for the clock-bias check drift rate var bound.*
- const std::string INTEGRITY_CN0_DIAGNOSTICS = "INTEGRITY_CN0_DIAGNOSTICS"

  *String ID for the CNO check diagnostic data.*
- const std::string INTEGRITY_CN0_DIAG_AVG_COUNT = "INTEGRITY_CN0_DIAG_AVG_COUNT"

  *String ID for the CNO check average count.*
- const std::string INTEGRITY_CN0_DIAG_ITHRESH = "INTEGRITY_CN0_DIAG_ITHRESH"

  *String ID for the CNO check survey inconsistent thresh.*
- const std::string INTEGRITY_CN0_DIAG_UTHRESH = "INTEGRITY_CN0_DIAG_UTHRESH"

  *String ID for the CNO check survey unassured thresh.*
- const double gpsPi = 3.1415926535898

  *PI as defined in IS-GPS-200 (30.3.3.1.3)*
- const double twoGpsPi = 2.0 ∗ gpsPi

  *2 ∗ PI as defined in IS-GPS-200 (convenience constant)*

- const double speedOfLight = 2.99792458e8

  *Speed of light as defined in IS-GPS-200 (20.3.4.3) [m/s].*
- const double gpsGM = 3.986005e14

  *Earth gravitational constant as defined in IS-GPS-200 (Tbl. 30-II) [m^3/s^2].*
- const double gpsF = -4.442807633e-10

  *Flattening constant as defined in IS-GPS-200 (20.3.3.3.3) [sec/meter^0.5].*
- const double gpsEarthRotationRate = 7.2921151467e-5
- const double secondsInWeek = 604800.0

  *Number of GPS seconds in a week.*
- const double secondsInHalfWeek = secondsInWeek / 2.0

  *Number of GPS seconds in a half week.*
- const std::string INTEGRITY_NAV_DATA_DIAGNOSTICS

  *String ID for the nav data check diagnostic data.*
- const std::string INTEGRITY_NAV_DATA_VALID = "INTEGRITY_NAV_DATA_VALID"

  *String ID for the nav data check data valid flag.*
- const std::string INTEGRITY_NAV_DATA_VALID_MSG = "INTEGRITY_NAV_DATA_VALID_MSG"

  *String ID for the nav data check data valid msg.*
- const std::string INTEGRITY_NAV_DATA_TOW_VALID = "INTEGRITY_NAV_DATA_TOW_VALID"

  *String ID for the nav data check tow valid flag.*
- const std::string INTEGRITY_NAV_DATA_TOW_VALID_MSG

  *String ID for the nav data check tow valid flag msg.*
- const std::string INTEGRITY_NAV_DATA_WN_VALID = "INTEGRITY_NAV_DATA_WN_VALID"

  *String ID for the nav data check week number valid flag.*
- const std::string INTEGRITY_NAV_DATA_WN_VALID_MSG

  *String ID for the nav data check week number valid flag msg.*
- const std::string INTEGRITY_POS_JUMP_DIAGNOSTICS

  *String ID for the position-jump check diagnostic data.*
- const std::string INTEGRITY_POS_JUMP_DIAG_BOUND

  *String ID for the position-jump check bound.*
- const std::string INTEGRITY_POS_JUMP_DIAG_DIST = "INTEGRITY_POS_JUMP_DIAG_DIST"

  *String ID for the position-jump check distance.*
- const std::string INTEGRITY_PVC_DIAGNOSTICS = "INTEGRITY_PVC_DIAGNOSTICS"

  *String ID for the position-velocity consistent check diagnostics data.*
- const std::string INTEGRITY_PVC_DIAG_PB = "INTEGRITY_PVC_DIAG_PB"

  *String ID for PVC diagnostic key for the "percent bad" variable.*
- const std::string INTEGRITY_PVC_DIAG_ITHRESH = "INTEGRITY_PVC_DIAG_ITHRESH"

  *String ID for the PVC diagnostic key for the inconsistent threshold.*
- const std::string INTEGRITY_PVC_DIAG_UTHRESH = "INTEGRITY_PVC_DIAG_UTHRESH"

  *String ID for the PVC diagnostic key for the unassured threshold.*
- const std::string INTEGRITY_PVC_DIAG_ERR_VAL = "INTEGRITY_PVC_DIAG_ERR_VAL"

  *String ID for the PVC diagnostic key for error values.*
- const std::string INTEGRITY_PVC_DIAG_ERR_THRESH

  *String ID for the PVC diagnostic key for error thresh values.*
- const std::string INTEGRITY_RNG_POS_DIAGNOSTICS

  *String ID for the range-position check diagnostic data.*
- const std::string INTEGRITY_RNG_POS_DIAG_MAX_CALC

  *String ID for the range-position check max calculated range.*

- const std::string INTEGRITY_RNG_POS_DIAG_MIN_CALC

  *String ID for the range-position check min calculated range.*
- const std::string INTEGRITY_RNG_POS_DIAG_MAX_MEAS

  *String ID for the range-position check max measured range.*
- const std::string INTEGRITY_RNG_POS_DIAG_MIN_MEAS

  *String ID for the range-position check min measured range.*
- const std::string INTEGRITY_STATIC_POS_DIAGNOSTICS

  *String ID for the static position check diagnostic data.*
- const std::string INTEGRITY_STAIC_POS_DIAG_POS_LAT

  *String ID for the static position check survey latitude.*
- const std::string INTEGRITY_STAIC_POS_DIAG_POS_LON

  *String ID for the static position check survey longitdue.*
- const std::string INTEGRITY_STAIC_POS_DIAG_POS_ALT

  *String ID for the static position check survey altitude.*
- const std::string INTEGRITY_STAIC_POS_DIAG_POS_CHNG_THRESH

  *String ID for the static position check change threshold.*
- const std::string INTEGRITY_STAIC_POS_DIAG_PERCENT_OVER

  *String ID for the static position check percentage threshold.*
- const std::string INTEGRITY_STAIC_POS_DIAG_ITHRESH

  *String ID for the static position check survey inconsistent thresh.*
- const std::string INTEGRITY_STAIC_POS_DIAG_UTHRESH

  *String ID for the static position check survey unassured thresh.*

## 7.2.1 Detailed Description

Namespace for all pnt_integrity applications.

## 7.2.2 Typedef Documentation

### 7.2.2.1 PeakResultsMap

```
using pnt_integrity::PeakResultsMap = typedef std::map<int, std::pair<double, double> >
```

A map that holds the first and second peak values in each acquisition plan

Definition at line 92 of file AcquisitionCheck.hpp.

#### 7.2.2.2 TimeEntryHistory

`using pnt_integrity::TimeEntryHistory = typedef std::map<double, TimeEntry>`

Defining a type for a history of time entries, which is realized by an ordered map keyed on time.

Definition at line 85 of file IntegrityDataRepository.hpp.

### 7.2.3 Enumeration Type Documentation

#### 7.2.3.1 AlertFlag

`enum pnt_integrity::AlertFlag [strong]`

Enumeration to define the alert flag given in bit 18 of the HOW If the alert flag is raised, it indicates to the user that the URA may be worse than indicated in subframe 1 and that the SV should be used only at the user's risk Rcef: IS-GPS-200 - 20.3.3.2

Definition at line 90 of file GPSEphemeris.hpp.

#### 7.2.3.2 AntiSpoofFlag

`enum pnt_integrity::AntiSpoofFlag [strong]`

Anti-spoof flag given in bit 19 of the handover word (HOW) If anti-spoof is on, P(Y) code is transmitted If anti-spoof is off, P code is transmitted Ref: IS-GPS-200 - 20.3.3.2

Definition at line 58 of file GPSEphemeris.hpp.

#### 7.2.3.3 FitInterval

`enum pnt_integrity::FitInterval [strong]`

Fit interval for the ephemeris data provided in subframe 2. Indicates if the satellite is under normal operations (fit interval = 4) or extended operations with a fit interval of greater than 4 hours. Ref: IS-GPS-200 - 20.3.3.4.3.1

Definition at line 79 of file GPSEphemeris.hpp.

#### 7.2.3.4 L2CodeType

`enum pnt_integrity::L2CodeType [strong]`

Code on L2. Indicates if C/A code, P code, or both are on on L2 Given in bits 11 and 12 of subframe 1 Ref: IS-GPS-200 - 20.3.3.3.1.2

Definition at line 67 of file GPSEphemeris.hpp.

#### 7.2.3.5 L2NavDataFlag

`enum pnt_integrity::L2NavDataFlag [strong]`

Enumeration to define the data flag for L2 P-code. Indicates if the NAV data stream has been turned on the P-code channel on L2 Ref: IS-GPS-200, 20.3.3.3.1.6

Definition at line 99 of file GPSEphemeris.hpp.

#### 7.2.3.6 SVNavHealth

`enum pnt_integrity::SVNavHealth : uint8_t [strong]`

Enumeration to define the top 3 bits of the 8-bith satellite health field included in Almanac subframes 4 and 5 Defined in paragraph 20.3.3.5.1.3 of IS-GPS-200

Definition at line 52 of file GPSAlmanac.hpp.

#### 7.2.3.7 SVSignalHealth

`enum pnt_integrity::SVSignalHealth : uint8_t [strong]`

Enumeration to define the 5 bit satellite signal health field given in bits 18 to 22 of subframe 1 and in the bottom 5 bits of the 8 bit satellite health field in Almanac subframes 4 and 5 Defined in paragraph 20.3.3.5.1.3 of IS-GPS-200

Definition at line 70 of file GPSNavDataCommon.hpp.

### 7.2.4 Function Documentation

#### 7.2.4.1 fromHex()

```
void pnt_integrity::fromHex (
             const std::string & in,
             void *const data )
```

Convert hexadecimal string to char array.

**Parameters**

| | |
|---|---|
| *in* | Input hex string |

#### 7.2.4.2 toHex()

```
void pnt_integrity::toHex (
             unsigned char *const byteData,
             const size_t dataLength,
             std::string & dest )
```

Convert char array to hexadecimal string.

**Parameters**

| | |
|---|---|
| *byteData* | Data to convert |
| *dataLength* | Length of the data to convert |

### 7.2.5 Variable Documentation

#### 7.2.5.1 gpsEarthRotationRate

```
const double pnt_integrity::gpsEarthRotationRate = 7.2921151467e-5
```

Earth rotation rate about ECEF Z-axis (little omega), as defined in IS-GPS-200 (Table 20-IV) [rad/s]

Definition at line 60 of file GPSNavDataCommon.hpp.

#### 7.2.5.2 INT_ACQ_DIAG_PEAK_RATIO_THRESH

```
const std::string pnt_integrity::INT_ACQ_DIAG_PEAK_RATIO_THRESH
```

**Initial value:**

```
=
  "INT_ACQ_DIAG_PEAK_RATIO_THRESH"
```

String ID for the ACQ check peak ratio threshold.

Definition at line 65 of file AcquisitionCheck.hpp.

#### 7.2.5.3 INTEGRITY_AOA_DIAG_ASSURED_PRN_PERCENT

```
const std::string pnt_integrity::INTEGRITY_AOA_DIAG_ASSURED_PRN_PERCENT
```

**Initial value:**

```
=
  "INTEGRITY_AOA_DIAG_ASSURED_PRN_PERCENT"
```

String ID for the AOA check diagnostic assured prn percent.

Definition at line 66 of file AngleOfArrivalCheck.hpp.

#### 7.2.5.4 INTEGRITY_AOA_DIAG_DIFF_THRESH

```
const std::string pnt_integrity::INTEGRITY_AOA_DIAG_DIFF_THRESH
```

**Initial value:**

```
=
  "INTEGRITY_AOA_DIAG_DIFF_THRESH"
```

String ID for the AOA check diagnostic difference threshold.

Definition at line 57 of file AngleOfArrivalCheck.hpp.

#### 7.2.5.5 INTEGRITY_AOA_DIAG_SUSPECT_PRN_PERCENT

```
const std::string pnt_integrity::INTEGRITY_AOA_DIAG_SUSPECT_PRN_PERCENT
```

**Initial value:**

```
=
  "INTEGRITY_AOA_DIAG_SUSPECT_PRN_PERCENT"
```

String ID for the AOA check diagnostic suspect prn percent.

Definition at line 60 of file AngleOfArrivalCheck.hpp.

#### 7.2.5.6 INTEGRITY_AOA_DIAG_UNAVAILABLE_PRN_PERCENT

`const std::string pnt_integrity::INTEGRITY_AOA_DIAG_UNAVAILABLE_PRN_PERCENT`

**Initial value:**

```
=
  "INTEGRITY_AOA_DIAG_UNAVAILABLE_PRN_PERCENT"
```

String ID for the AOA check diagnostic unavailable prn percent.

Definition at line 63 of file AngleOfArrivalCheck.hpp.

#### 7.2.5.7 INTEGRITY_AOA_DIFF_DIAGNOSTICS

`const std::string pnt_integrity::INTEGRITY_AOA_DIFF_DIAGNOSTICS`

**Initial value:**

```
=
  "INTEGRITY_AOA_DIFF_DIAGNOSTICS"
```

String ID for the AOA check difference diagnostic data.

Definition at line 50 of file AngleOfArrivalCheck.hpp.

#### 7.2.5.8 INTEGRITY_CLOCK_BIAS_DIAG_ACTUAL_OFFSET

`const std::string pnt_integrity::INTEGRITY_CLOCK_BIAS_DIAG_ACTUAL_OFFSET`

**Initial value:**

```
=
  "INTEGRITY_CLOCK_BIAS_DIAG_ACTUAL_OFFSET"
```

String ID for the clock-bias check actual offset.

Definition at line 60 of file ClockBiasCheck.hpp.

#### 7.2.5.9 INTEGRITY_CLOCK_BIAS_DIAG_DRIFT_RATE_BOUND

```
const std::string pnt_integrity::INTEGRITY_CLOCK_BIAS_DIAG_DRIFT_RATE_BOUND
```

**Initial value:**

```
=
  "INTEGRITY_CLOCK_BIAS_DIAG_DRIFT_RATE_BOUND"
```

String ID for the clock-bias check drift rate bound.

Definition at line 66 of file ClockBiasCheck.hpp.

#### 7.2.5.10 INTEGRITY_CLOCK_BIAS_DIAG_DRIFT_RATE_VAR_BOUND

```
const std::string pnt_integrity::INTEGRITY_CLOCK_BIAS_DIAG_DRIFT_RATE_VAR_BOUND
```

**Initial value:**

```
=
  "INTEGRITY_CLOCK_BIAS_DIAG_DRIFT_RATE_VAR_BOUND"
```

String ID for the clock-bias check drift rate var bound.

Definition at line 69 of file ClockBiasCheck.hpp.

#### 7.2.5.11 INTEGRITY_CLOCK_BIAS_DIAG_EXP_DRIFT

```
const std::string pnt_integrity::INTEGRITY_CLOCK_BIAS_DIAG_EXP_DRIFT
```

**Initial value:**

```
=
  "INTEGRITY_CLOCK_BIAS_DIAG_EXP_DRIFT"
```

String ID for the clock-bias check expected drift.

Definition at line 51 of file ClockBiasCheck.hpp.

#### 7.2.5.12 INTEGRITY_CLOCK_BIAS_DIAG_EXP_DRIFT_VAR

`const std::string pnt_integrity::INTEGRITY_CLOCK_BIAS_DIAG_EXP_DRIFT_VAR`

**Initial value:**

```
=
  "INTEGRITY_CLOCK_BIAS_DIAG_EXP_DRIFT_VAR"
```

String ID for the clock-bias check drift variance.

Definition at line 54 of file ClockBiasCheck.hpp.

#### 7.2.5.13 INTEGRITY_CLOCK_BIAS_DIAG_OFFSET_ERROR

`const std::string pnt_integrity::INTEGRITY_CLOCK_BIAS_DIAG_OFFSET_ERROR`

**Initial value:**

```
=
  "INTEGRITY_CLOCK_BIAS_DIAG_OFFSET_ERROR"
```

String ID for the clock-bias check offset error.

Definition at line 63 of file ClockBiasCheck.hpp.

#### 7.2.5.14 INTEGRITY_CLOCK_BIAS_DIAG_PROP_OFFSET

`const std::string pnt_integrity::INTEGRITY_CLOCK_BIAS_DIAG_PROP_OFFSET`

**Initial value:**

```
=
  "INTEGRITY_CLOCK_BIAS_DIAG_PROP_OFFSET"
```

String ID for the clock-bias check propagation offset.

Definition at line 57 of file ClockBiasCheck.hpp.

#### 7.2.5.15 INTEGRITY_CLOCK_BIAS_DIAGNOSTICS

`const std::string pnt_integrity::INTEGRITY_CLOCK_BIAS_DIAGNOSTICS`

**Initial value:**

```
=
  "INTEGRITY_CLOCK_BIAS_DIAGNOSTICS"
```

String ID for the clock-bias check diagnostic data.

Definition at line 48 of file ClockBiasCheck.hpp.

#### 7.2.5.16 INTEGRITY_NAV_DATA_DIAGNOSTICS

`const std::string pnt_integrity::INTEGRITY_NAV_DATA_DIAGNOSTICS`

**Initial value:**

```
=
  "INTEGRITY_NAV_DATA_DIAGNOSTICS"
```

String ID for the nav data check diagnostic data.

Definition at line 51 of file NavigationDataCheck.hpp.

#### 7.2.5.17 INTEGRITY_NAV_DATA_TOW_VALID_MSG

`const std::string pnt_integrity::INTEGRITY_NAV_DATA_TOW_VALID_MSG`

**Initial value:**

```
=
  "INTEGRITY_NAV_DATA_TOW_VALID_MSG"
```

String ID for the nav data check tow valid flag msg.

Definition at line 60 of file NavigationDataCheck.hpp.

#### 7.2.5.18 INTEGRITY_NAV_DATA_WN_VALID_MSG

```
const std::string pnt_integrity::INTEGRITY_NAV_DATA_WN_VALID_MSG
```

**Initial value:**

```
=
  "INTEGRITY_NAV_DATA_WN_VALID_MSG"
```

String ID for the nav data check week number valid flag msg.

Definition at line 65 of file NavigationDataCheck.hpp.

#### 7.2.5.19 INTEGRITY_POS_JUMP_DIAG_BOUND

```
const std::string pnt_integrity::INTEGRITY_POS_JUMP_DIAG_BOUND
```

**Initial value:**

```
=
  "INTEGRITY_POS_JUMP_DIAG_BOUND"
```

String ID for the position-jump check bound.

Definition at line 52 of file PositionJumpCheck.hpp.

#### 7.2.5.20 INTEGRITY_POS_JUMP_DIAGNOSTICS

```
const std::string pnt_integrity::INTEGRITY_POS_JUMP_DIAGNOSTICS
```

**Initial value:**

```
=
  "INTEGRITY_POS_JUMP_DIAGNOSTICS"
```

String ID for the position-jump check diagnostic data.

Definition at line 49 of file PositionJumpCheck.hpp.

#### 7.2.5.21 INTEGRITY_PVC_DIAG_ERR_THRESH

`const std::string pnt_integrity::INTEGRITY_PVC_DIAG_ERR_THRESH`

**Initial value:**

```
=
  "INTEGRITY_PVC_DIAG_ERR_THRESH"
```

String ID for the PVC diagnostic key for error thresh values.

Definition at line 61 of file PositionVelocityConsistencyCheck.hpp.

#### 7.2.5.22 INTEGRITY_RNG_POS_DIAG_MAX_CALC

`const std::string pnt_integrity::INTEGRITY_RNG_POS_DIAG_MAX_CALC`

**Initial value:**

```
=
  "INTEGRITY_RNG_POS_DIAG_MAX_CALC"
```

String ID for the range-position check max calculated range.

Definition at line 50 of file RangePositionCheck.hpp.

#### 7.2.5.23 INTEGRITY_RNG_POS_DIAG_MAX_MEAS

`const std::string pnt_integrity::INTEGRITY_RNG_POS_DIAG_MAX_MEAS`

**Initial value:**

```
=
  "INTEGRITY_RNG_POS_DIAG_MAX_MEAS"
```

String ID for the range-position check max measured range.

Definition at line 56 of file RangePositionCheck.hpp.

#### 7.2.5.24 INTEGRITY_RNG_POS_DIAG_MIN_CALC

`const std::string pnt_integrity::INTEGRITY_RNG_POS_DIAG_MIN_CALC`

**Initial value:**

```
=
  "INTEGRITY_RNG_POS_DIAG_MIN_CALC"
```

String ID for the range-position check min calculated range.

Definition at line 53 of file RangePositionCheck.hpp.

#### 7.2.5.25 INTEGRITY_RNG_POS_DIAG_MIN_MEAS

`const std::string pnt_integrity::INTEGRITY_RNG_POS_DIAG_MIN_MEAS`

**Initial value:**

```
=
  "INTEGRITY_RNG_POS_DIAG_MIN_MEAS"
```

String ID for the range-position check min measured range.

Definition at line 59 of file RangePositionCheck.hpp.

#### 7.2.5.26 INTEGRITY_RNG_POS_DIAGNOSTICS

`const std::string pnt_integrity::INTEGRITY_RNG_POS_DIAGNOSTICS`

**Initial value:**

```
=
  "INTEGRITY_RNG_POS_DIAGNOSTICS"
```

String ID for the range-position check diagnostic data.

Definition at line 47 of file RangePositionCheck.hpp.

#### 7.2.5.27 INTEGRITY_STAIC_POS_DIAG_ITHRESH

`const std::string pnt_integrity::INTEGRITY_STAIC_POS_DIAG_ITHRESH`

**Initial value:**

```
=
  "INTEGRITY_STAIC_POS_DIAG_ITHRESH"
```

String ID for the static position check survey inconsistent thresh.

Definition at line 68 of file StaticPositionCheck.hpp.

#### 7.2.5.28 INTEGRITY_STAIC_POS_DIAG_PERCENT_OVER

`const std::string pnt_integrity::INTEGRITY_STAIC_POS_DIAG_PERCENT_OVER`

**Initial value:**

```
=
  "INTEGRITY_STAIC_POS_DIAG_PERCENT_OVER"
```

String ID for the static position check percentage threshold.

Definition at line 65 of file StaticPositionCheck.hpp.

#### 7.2.5.29 INTEGRITY_STAIC_POS_DIAG_POS_ALT

`const std::string pnt_integrity::INTEGRITY_STAIC_POS_DIAG_POS_ALT`

**Initial value:**

```
=
  "INTEGRITY_STAIC_POS_DIAG_POS_ALT"
```

String ID for the static position check survey altitude.

Definition at line 59 of file StaticPositionCheck.hpp.

#### 7.2.5.30 INTEGRITY_STAIC_POS_DIAG_POS_CHNG_THRESH

`const std::string pnt_integrity::INTEGRITY_STAIC_POS_DIAG_POS_CHNG_THRESH`

**Initial value:**

```
=
  "INTEGRITY_STAIC_POS_DIAG_POS_CHNG_THRESH"
```

String ID for the static position check change threshold.

Definition at line 62 of file StaticPositionCheck.hpp.

#### 7.2.5.31 INTEGRITY_STAIC_POS_DIAG_POS_LAT

`const std::string pnt_integrity::INTEGRITY_STAIC_POS_DIAG_POS_LAT`

**Initial value:**

```
=
  "INTEGRITY_STAIC_POS_DIAG_POS_LAT"
```

String ID for the static position check survey latitude.

Definition at line 53 of file StaticPositionCheck.hpp.

#### 7.2.5.32 INTEGRITY_STAIC_POS_DIAG_POS_LON

`const std::string pnt_integrity::INTEGRITY_STAIC_POS_DIAG_POS_LON`

**Initial value:**

```
=
  "INTEGRITY_STAIC_POS_DIAG_POS_LON"
```

String ID for the static position check survey longitdue.

Definition at line 56 of file StaticPositionCheck.hpp.

#### 7.2.5.33 INTEGRITY_STAIC_POS_DIAG_UTHRESH

`const std::string pnt_integrity::INTEGRITY_STAIC_POS_DIAG_UTHRESH`

**Initial value:**

```
=
  "INTEGRITY_STAIC_POS_DIAG_UTHRESH"
```

String ID for the static position check survey unassured thresh.

Definition at line 71 of file StaticPositionCheck.hpp.

#### 7.2.5.34 INTEGRITY_STATIC_POS_DIAGNOSTICS

`const std::string pnt_integrity::INTEGRITY_STATIC_POS_DIAGNOSTICS`

**Initial value:**

```
=
  "INTEGRITY_STATIC_POS_DIAGNOSTICS"
```

String ID for the static position check diagnostic data.

Definition at line 50 of file StaticPositionCheck.hpp.

## 7.3 pnt_integrity::data Namespace Reference

Namespace for all integrity data definitions.

## Classes

- struct AccumulatedDistranceTraveled

  *A structure that represents a distance traveled over a time period.*
- struct AgcValue

  *A structure to represent an AGC measurement.*
- struct AssuranceReport

  *A structure to hold a single assurance report.*
- struct AssuranceReports

  *A structure to hold assurance data for all registered checks.*
- class AssuranceState

  *A structure to hold an AssuranceLevel and value.*
- struct ClockOffset

  *A structure for measuring the offset between two clocks.*
- struct GeodeticPosition3d

  *A structure to represent 3D geodetic position.*
- struct GNSSObservable

  *A structure for GNSS observables (pseudorange, carrier, doppler, etc)*
- struct GNSSObservables

  *The GNSSObservables message.*
- struct GNSSSubframe

  *GNSS Subframe data.*
- struct GNSSTime

  *A GNSS time.*
- struct Header

  *The header used for all associated data types.*
- struct IMU

  *A structure that represents IMU measurement data.*
- struct MeasuredRange

  *A structure that represents a distance measurement to a known point.*
- struct PositionVelocity

  *A structure to represent a Position / Velocity message.*
- struct RfSpectrum

  *A structure that represents an RF spectrum measurement.*
- struct Timestamp

  *A timestamp used in all headers.*

## Typedefs

- using GNSSObservableMap = std::map< uint64_t, GNSSObservable >

  *A map to relate a GNSSObservable to a PRN.*

### Enumerations

- enum TimeSystem { **GLO** = 0, **GPS**, **GAL**, **BDT** }

  *Enumeration for all available satellite-based time system sources.*
- enum SatelliteSystem : uint8_t {
  **GPS** = 0, **Glonass**, **Galileo**, **QZSS**,
  **BeiDou**, **IRNSS**, **SBAS**, **Mixed**,
  **Other** }

  *Enumeration for satellite system identification.*
- enum FrequencyBand : uint8_t {
  **Band1** = 0, **Band2**, **Band5**, **Band6**,
  **Band7**, **Band8**, **Band9**, **Band0**,
  **Band10** }

  *Defines all possible frequency types.*
- enum CodeType : uint8_t {
  **SigP** = 0, **SigC**, **SigD**, **SigY**,
  **SigM**, **SigN**, **SigA**, **SigB**,
  **SigI**, **SigQ**, **SigS**, **SigL**,
  **SigX**, **SigW**, **SigZ**, **SigBLANK** }

  *Defines all possible code types.*
- enum AssuranceLevel : int8_t { **Unavailable** = 0, **Unassured**, **Inconsistent**, **Assured** }

  *Defines all available assurance level values.*

### 7.3.1 Detailed Description

Namespace for all integrity data definitions.

# Chapter 8

# Class Documentation

## 8.1 pnt_integrity::data::AccumulatedDistranceTraveled Struct Reference

A structure that represents a distance traveled over a time period.

`#include <IntegrityData.hpp>`

### Public Attributes

- Header header

  *The message header.*
- double dt

  *Time span of accumulated distance (s)*
- double distance

  *Accumulated distance traveled over time period (m)*
- double variance

  *Accumulated distance traveled variance (m^2)*

### 8.1.1 Detailed Description

A structure that represents a distance traveled over a time period.

Definition at line 759 of file IntegrityData.hpp.

The documentation for this struct was generated from the following file:

- include/pnt_integrity/IntegrityData.hpp

## 8.2 pnt_integrity::AcqCheckDiagnostics Struct Reference

Structure for publishing Acquisition Check diagnostics.

`#include <AcquisitionCheck.hpp>`

### Public Attributes

- double highPowerThresh

  *The threshold to indicate high-power in a prn acquisition.*
- double peakRatioThresh

  *The threshold on peak 1 to peak 2 ratio to determine a suspect prn.*
- double acquisitionThresh

  *The threshold on the acuqisition plane to indicate a good prn.*
- double inconsistentThresh

  *The threshold used for determining an overall inconsistent assurance level.*
- double unassuredThresh

  *The threshold used for determining an overall unassured assurance level.*
- double unassuredCount

  *The number of prns flagged as unassured.*
- double inconsistentCount

  *The number of prns flagged as inconsistent.*
- std::map< int, double > ratioMap

  *A map that pairs PRN id to the peak ratio.*

### 8.2.1 Detailed Description

Structure for publishing Acquisition Check diagnostics.

Definition at line 97 of file AcquisitionCheck.hpp.

The documentation for this struct was generated from the following file:

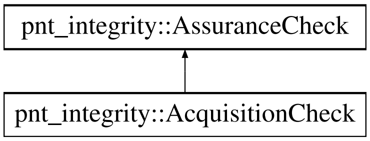
- include/pnt_integrity/AcquisitionCheck.hpp

## 8.3 pnt_integrity::AcquisitionCheck Class Reference

Class implementation for the acquisition check.

`#include <AcquisitionCheck.hpp>`

Inheritance diagram for pnt_integrity::AcquisitionCheck:


pnt_integrity::AssuranceCheck
pnt_integrity::AcquisitionCheck

## Public Member Functions

- AcquisitionCheck (const std::string &name="Acquisition check", const double &highPowerThreshold=2.5e7, const double &peakRatioThreshold=7.0, const double &acqusitionThreshold=3e6, const double &expectedSamplingFreq=5e6, const double &intermediateFreq=0.0, const double &searchBand=10e3, const double &searchStepSize=0.5e3, const double &integrationPeriod=1e-3, const double &codeFrequencyBasis=1.023e6, const int &codeLength=1023, const logutils::LogCallback &log=logutils::printLogToStdOut)

  *Constructor for the check class.*
- bool handleIFSampleData (const double &checkTime, const if_data_utils::IFSampleData< if_data_utils::IFSampleSC8 > &ifData)

  *Handler function for IF sample data (SC8)*
- bool handleIFSampleData (const double &checkTime, const if_data_utils::IFSampleData< if_data_utils::IFSampleSC16 > &ifData)

  *Handler function for IF sample data (SC16)*
- template< typename samp_type >
  bool processIFSampleData (const if_data_utils::IFSampleData< samp_type > &sampleData)

  *Functon to processing incoming samples.*
- void calculateAssuranceLevel (const double &time)

  *Function to explicitly set the assurance level of the check.*
- void setPublishAquisition (std::function< void(const CorrelationResultsMap &)> handler)

  *Connects the internal publishing function to external interface.*
- void setPublishPeakData (std::function< void(const double &, const PeakResultsMap &)> handler)

  *Connects the internal publishing function to external interface.*
- void setPublishDiagnostics (std::function< void(const double &timestamp, const AcqCheckDiagnostics &checkData)> handler)

  *Connects the internal publishing function to external interface.*

## Additional Inherited Members

### 8.3.1 Detailed Description

Class implementation for the acquisition check.

Class implementation of the acquisition check. The class is a child class of AssuranceCheck

Definition at line 120 of file AcquisitionCheck.hpp.

### 8.3.2 Constructor & Destructor Documentation

#### 8.3.2.1 AcquisitionCheck()

```
pnt_integrity::AcquisitionCheck::AcquisitionCheck (
             const std::string & name = "Acquisition check",
             const double & highPowerThreshold = 2.5e7,
             const double & peakRatioThreshold = 7.0,
             const double & acqusitionThreshold = 3e6,
             const double & expectedSamplingFreq = 5e6,
             const double & intermediateFreq = 0.0,
             const double & searchBand = 10e3,
             const double & searchStepSize = 0.5e3,
             const double & integrationPeriod = 1e-3,
             const double & codeFrequencyBasis = 1.023e6,
             const int & codeLength = 1023,
             const logutils::LogCallback & log = logutils::printLogToStdOut )  [inline]
```

Constructor for the check class.

Constructor for the acuqisiton check, the default constructor configures the check for L1-CA at 5 MSps. Acquisition parameters will be recalculated if a different sampling frequency is detected

**Parameters**

| | |
|---|---|
| *name* | A string name for the check instance |
| *highPowerThreshold* | A threshold that indicates abnormally high power levels |
| *peakRatioThreshold* | A threshold for the ratio of the first and second peaks in the acquisition plane, indicating a possible unauthentic signal |
| *acqusitionThreshold* | A threshold for classifiing a signal as acquired or unacquired |
| *expectedSamplingFreq* | The expected sampling frequency |
| *intermediateFreq* | The intermediate frequency of incoming sample stream |
| *searchBand* | The acquisition search band |
| *searchStepSize* | The acquisition search step size (defines bins) |
| *integrationPeriod* | Integration period to use for acquisition |
| *codeFrequencyBasis* | Freqeuncy basis for the code of interest |
| *codeLength* | Length of the code (in chips) |
| *log* | The provided log handler function |

Definition at line 146 of file AcquisitionCheck.hpp.

### 8.3.3 Member Function Documentation

#### 8.3.3.1 calculateAssuranceLevel()

```
void pnt_integrity::AcquisitionCheck::calculateAssuranceLevel (
             const double & time )  [virtual]
```

Function to explicitly set the assurance level of the check.

For this check, this function cycles through all of the individual PRN assurance values, analyzes them, and then sets the master assurance level associted with the check.

Implements pnt_integrity::AssuranceCheck.

#### 8.3.3.2 handleIFSampleData() [1/2]

```
bool pnt_integrity::AcquisitionCheck::handleIFSampleData (
             const double & checkTime,
             const if_data_utils::IFSampleData< if_data_utils::IFSampleSC8 > & ifData )  [inline],
[virtual]
```

Handler function for IF sample data (SC8)

Function to handle provided IF data. (Overriding inherited function from parent class). Calls the common templated function processIfSampleData for convenience

**Parameters**

| | |
|---|---|
| *checkTime* | The timestamp associated with the data |
| *ifData* | The provided IF data sample set |

**Returns**

True if successful

Reimplemented from pnt_integrity::AssuranceCheck.

Definition at line 200 of file AcquisitionCheck.hpp.

#### 8.3.3.3 handleIFSampleData() [2/2]

```
bool pnt_integrity::AcquisitionCheck::handleIFSampleData (
             const double & checkTime,
             const if_data_utils::IFSampleData< if_data_utils::IFSampleSC16 > & ifData )  [inline],
[virtual]
```

Handler function for IF sample data (SC16)

Function to handle provided IF data (Overriding inherited function from parent class). Calls the common templated function processIfSampleData for convenience

**Parameters**

| | |
|---|---|
| *checkTime* | The timestamp associated with the data |
| *ifData* | The provided IF data sample set |

**Returns**

True if successful

Reimplemented from pnt_integrity::AssuranceCheck.

Definition at line 227 of file AcquisitionCheck.hpp.

#### 8.3.3.4 processIFSampleData()

```
template<typename samp_type >
bool pnt_integrity::AcquisitionCheck::processIFSampleData (
             const if_data_utils::IFSampleData< samp_type > & sampleData )
```

Functon to processing incoming samples.

Template function for processing incoming samples

**Parameters**

| | |
|---|---|
| *sampleData* | Incoming sample data |

Definition at line 412 of file AcquisitionCheck.hpp.

#### 8.3.3.5 setPublishAquisition()

```
void pnt_integrity::AcquisitionCheck::setPublishAquisition (
             std::function< void(const CorrelationResultsMap &)> handler )  [inline]
```

Connects the internal publishing function to external interface.

This function connects the internal "publishAcquisitionData" function to an external, custom function of choice

**Parameters**

| | |
|---|---|
| *handler* | Provided handler function |

Definition at line 267 of file AcquisitionCheck.hpp.

#### 8.3.3.6 setPublishDiagnostics()

```
void pnt_integrity::AcquisitionCheck::setPublishDiagnostics (
            std::function< void(const double &timestamp, const AcqCheckDiagnostics &checkData)>
handler )  [inline]
```

Connects the internal publishing function to external interface.

This function connects the internal "publishDiagnostics" function to an external, custom function of choice

**Parameters**

| | |
|---|---|
| *handler* | Provided handler function |

Definition at line 293 of file AcquisitionCheck.hpp.

#### 8.3.3.7 setPublishPeakData()

```
void pnt_integrity::AcquisitionCheck::setPublishPeakData (
            std::function< void(const double &, const PeakResultsMap &)> handler )  [inline]
```

Connects the internal publishing function to external interface.

This function connects the internal "publishPeakData" function to an external, custom function of choice

**Parameters**

| | |
|---|---|
| *handler* | Provided handler function |

Definition at line 280 of file AcquisitionCheck.hpp.

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

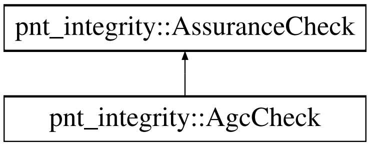
- include/pnt_integrity/AcquisitionCheck.hpp

## 8.4 pnt_integrity::AgcCheck Class Reference

Class implementation for the AGC check.

`#include <AgcCheck.hpp>`

Inheritance diagram for pnt_integrity::AgcCheck:



## Public Member Functions

- AgcCheck (const std::string &name="agc_check", const double &minValue=0.0, const double &maxValue=10000, const logutils::LogCallback &log=logutils::printLogToStdOut)

  *Constructor for the AgcCheck class.*
- bool handleAGC (const data::AgcValue &agcValue)

  *Handler function for AGC value.*
- void calculateAssuranceLevel (const double &)

  *Function to explicitly set the assurance level of the check.*
- void setPublishDiagnostics (std::function< void(const double &timestamp, const AgcCheckDiagnostics &checkData)> handler)

  *Connects the internal publishing function to external interface.*

## Additional Inherited Members

### 8.4.1 Detailed Description

Class implementation for the AGC check.

Definition at line 61 of file AgcCheck.hpp.

### 8.4.2 Constructor & Destructor Documentation

#### 8.4.2.1 AgcCheck()

```
pnt_integrity::AgcCheck::AgcCheck (
             const std::string & name = "agc_check",
             const double & minValue = 0.0,
             const double & maxValue = 10000,
             const logutils::LogCallback & log = logutils::printLogToStdOut )  [inline]
```

Constructor for the AgcCheck class.

**Parameters**

| | |
|---|---|
| *name* | The name of the check |
| *minValue* | The minimum possible reported value for the AGC |
| *maxValue* | The maximum possible reported value for the AGC |
| *log* | Log handler function |

Definition at line 70 of file AgcCheck.hpp.

## 8.4.3 Member Function Documentation

### 8.4.3.1 handleAGC()

```
bool pnt_integrity::AgcCheck::handleAGC (
             const data::AgcValue & agcValue )  [inline], [virtual]
```

Handler function for AGC value.

Function to handle provided AGC values (virtual)

**Parameters**

| | |
|---|---|
| *agcValue* | The provided AGC message / structure |

**Returns**

True if successful

Reimplemented from pnt_integrity::AssuranceCheck.

Definition at line 95 of file AgcCheck.hpp.

### 8.4.3.2 setPublishDiagnostics()

```
void pnt_integrity::AgcCheck::setPublishDiagnostics (
             std::function< void(const double &timestamp, const AgcCheckDiagnostics &checkData)>
handler )  [inline]
```

Connects the internal publishing function to external interface.

This function connects the internal "publishDiagnostics" function to an external, custom function of choice

**Parameters**

| | |
|---|---|
| *handler* | Provided handler function |

Definition at line 111 of file AgcCheck.hpp.

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

- include/pnt_integrity/AgcCheck.hpp

## 8.5 pnt_integrity::AgcCheckDiagnostics Struct Reference

Diagnostic data for AGC check.

`#include <AgcCheck.hpp>`

### Public Attributes

- data::AgcValue values

  *The AGC values.*
- double inconsistentThresh

  *The inconsistent threshold.*

### 8.5.1 Detailed Description

Diagnostic data for AGC check.

Definition at line 52 of file AgcCheck.hpp.

The documentation for this struct was generated from the following file:

- include/pnt_integrity/AgcCheck.hpp

## 8.6 pnt_integrity::data::AgcValue Struct Reference

A structure to represent an AGC measurement.

`#include <IntegrityData.hpp>`

### Public Attributes

- Header header

  *The message header.*
- std::map$<$ FrequencyBand, double $>$ agcValues

  *A vector for AGC values (multiple bands possible)*

### 8.6.1 Detailed Description

A structure to represent an AGC measurement.

Definition at line 837 of file IntegrityData.hpp.

The documentation for this struct was generated from the following file:

- include/pnt_integrity/IntegrityData.hpp

## 8.7 pnt_integrity::AlmanacParameters Struct Reference

### Public Attributes

- uint16_t **prn**
- double **tow**
- SVAlmHealth **svHealth**
- double **eccentricity**
- double **toa**
- double **deltal**
- double **omegaDot**
- double **sqrtA**
- double **omega0**
- double **omega**
- double **m0**
- double **af0**
- double **af1**
- uint16_t **referenceWeek**

### 8.7.1 Detailed Description

Definition at line 74 of file GPSAlmanac.hpp.

The documentation for this struct was generated from the following file:

- include/pnt_integrity/GPSAlmanac.hpp

## 8.8 pnt_integrity::AlmanacSubframeFaults Union Reference

### Classes

- struct FaultType

### Public Attributes

- FaultType **faultType**
- uint16_t **bitfield**

### 8.8.1 Detailed Description

Definition at line 94 of file GPSAlmanac.hpp.

The documentation for this union was generated from the following file:

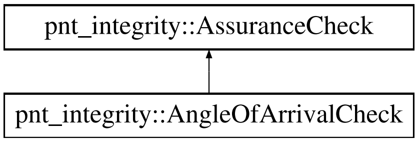
- include/pnt_integrity/GPSAlmanac.hpp

## 8.9 pnt_integrity::AngleOfArrivalCheck Class Reference

Class implementation for the angle of arrival check.

`#include <AngleOfArrivalCheck.hpp>`

Inheritance diagram for pnt_integrity::AngleOfArrivalCheck:


pnt_integrity::AssuranceCheck
pnt_integrity::AngleOfArrivalCheck

## Public Member Functions

- AngleOfArrivalCheck (const std::string &name="AOA check", const AoaCheckData &aoaCheckData=AoaCheckData::UsePseudorange, const double &singleDiffCompareThresh=5.0, const int &prnCountThresh=5, const double &rangeThreshold=5.0, const logutils::LogCallback &log=logutils::printLogToStdOut)

  *Constructor.*
- bool handleGnssObservables (const data::GNSSObservables &gnssObs, const double &time=0)

  *Handler function for GNSS Observables.*
- bool runCheck ()

  *Triggers a manual check calculation.*
- void calculateAssuranceLevel (const double &time)

  *Function to explicitly set the assurance level of the check.*
- void setDifferenceComparisonThreshold (const double &thresh)

  *Sets the difference comparison threshold.*
- void setDifferenceComparisonFailureLimit (const double &thresh)

  *Sets the Percent Failure Limit for the SingleDiffDiff Check.*
- void setPrnCountThreshold (const int &thresh)

  *Sets the prn count threshold.*
- void setRangeThreshold (const double &thresh)

  *Sets the range threshold.*
- void setPublishDiffData (std::function< void(const double &time, const std::string &remoteNodeId, const SingleDiffMap &)> handler)

  *Connects the internal publishing function to external interface.*
- void setPublishDiagnostics (std::function< void(const double &timestamp, const AoaCheckDiagnostics &checkData)> handler)

  *Connects the internal publishing function to external interface.*

## Additional Inherited Members

### 8.9.1 Detailed Description

Class implementation for the angle of arrival check.

Class implementation of the angle of arrival check. The class is a child class of AssuranceCheck

Definition at line 112 of file AngleOfArrivalCheck.hpp.

### 8.9.2 Constructor & Destructor Documentation

#### 8.9.2.1 AngleOfArrivalCheck()

```
pnt_integrity::AngleOfArrivalCheck::AngleOfArrivalCheck (
             const std::string & name = "AOA check",
             const AoaCheckData & aoaCheckData = AoaCheckData::UsePseudorange,
             const double & singleDiffCompareThresh = 5.0,
             const int & prnCountThresh = 5,
             const double & rangeThreshold = 5.0,
             const logutils::LogCallback & log = logutils::printLogToStdOut )  [inline]
```

Constructor.

Constructor for the angle of arrival check class. The constructor defaults the multi-prn support to true for this check, so enableMultiPrnSupport need not be called.

**Parameters**

| | |
|---|---|
| *name* | The name associated with the check |
| *aoaCheckData* | Sets which type of data will be used |
| *singleDiffCompareThresh* | Sets the threshold for comparing single difference values |
| *prnCountThresh* | A threshold used in the AOA check to determine if a PRN has a common AOA with other PRNs. |
| *rangeThreshold* | A distance threshold that is used when to determine if a remote node is to close to the local node to perform the AOA check |
| *log* | A provided log callback function to use |

Definition at line 131 of file AngleOfArrivalCheck.hpp.

### 8.9.3 Member Function Documentation

#### 8.9.3.1 calculateAssuranceLevel()

```
void pnt_integrity::AngleOfArrivalCheck::calculateAssuranceLevel (
             const double & time )  [virtual]
```

Function to explicitly set the assurance level of the check.

For this check, this function cycles through all of the individual PRN assurance values, analyzes them, and then sets the master assurance level associted with the check.

Implements pnt_integrity::AssuranceCheck.

#### 8.9.3.2 handleGnssObservables()

```
bool pnt_integrity::AngleOfArrivalCheck::handleGnssObservables (
             const data::GNSSObservables & gnssObs,
             const double & time = 0 )  [virtual]
```

Handler function for GNSS Observables.

Function to handle provided GNSS Observables. This function simply calls runCheck(), as the provided data has already been added to the repository

**Parameters**

| | |
|---|---|
| *gnssObs* | The provided GNSS observable data |

**Returns**

True if successful

Reimplemented from pnt_integrity::AssuranceCheck.

#### 8.9.3.3 runCheck()

```
bool pnt_integrity::AngleOfArrivalCheck::runCheck ( )  [virtual]
```

Triggers a manual check calculation.

Use this function to run a manual check calculation that is not triggered off the receipt of a message (pure virtual)

**Returns**

True if successful

Implements pnt_integrity::AssuranceCheck.

#### 8.9.3.4 setDifferenceComparisonFailureLimit()

```
void pnt_integrity::AngleOfArrivalCheck::setDifferenceComparisonFailureLimit (
             const double & thresh )  [inline]
```

Sets the Percent Failure Limit for the SingleDiffDiff Check.

Percentage Value expressed as [0 - 1]

**Parameters**

| | |
|---|---|
| *thresh* | The threshold value to use |

Definition at line 204 of file AngleOfArrivalCheck.hpp.

#### 8.9.3.5 setDifferenceComparisonThreshold()

```
void pnt_integrity::AngleOfArrivalCheck::setDifferenceComparisonThreshold (
            const double & thresh )  [inline]
```

Sets the difference comparison threshold.

This threshold is used to determine when 2 separate single differences (between a local and remote node) should be flagged has having a common angle of arrival. The units on this threshold depend on the type of data that is being used for the check (AoaCheckData). For example if, AoaCheckData::UsePseudorange is being used, then the units are in meters.

**Parameters**

| | |
|---|---|
| *thresh* | The threshold value to use |

Definition at line 193 of file AngleOfArrivalCheck.hpp.

#### 8.9.3.6 setPrnCountThreshold()

```
void pnt_integrity::AngleOfArrivalCheck::setPrnCountThreshold (
            const int & thresh )  [inline]
```

Sets the prn count threshold.

This threshold is used to determine when to raise the assurance level of a particular prn. If a PRN is found to have a common AOA with at least [threshold] other PRNS, then the AssuranceLevel is raised

**Parameters**

| | |
|---|---|
| *thresh* | The threshold value to use |

Definition at line 216 of file AngleOfArrivalCheck.hpp.

#### 8.9.3.7 setPublishDiagnostics()

```
void pnt_integrity::AngleOfArrivalCheck::setPublishDiagnostics (
              std::function< void(const double &timestamp, const AoaCheckDiagnostics &checkData)>
handler )  [inline]
```

Connects the internal publishing function to external interface.

This function connects the internal "publishDiagnostics" function to an external, custom function of choice

**Parameters**

| | |
|---|---|
| *handler* | Provided handler function |

Definition at line 256 of file AngleOfArrivalCheck.hpp.

#### 8.9.3.8 setPublishDiffData()

```
void pnt_integrity::AngleOfArrivalCheck::setPublishDiffData (
              std::function< void(const double &time, const std::string &remoteNodeId, const Single↩
DiffMap &)> handler )  [inline]
```

Connects the internal publishing function to external interface.

This function connects the internal "publishSingleDiffData" function to an external, custom function of choice

**Parameters**

| | |
|---|---|
| *handler* | Provided handler function |

Definition at line 242 of file AngleOfArrivalCheck.hpp.

#### 8.9.3.9 setRangeThreshold()

```
void pnt_integrity::AngleOfArrivalCheck::setRangeThreshold (
              const double & thresh )  [inline]
```

Sets the range threshold.

When calculating differences between local and remote observables, if a measured range is available between the two, it is compared to this threshold. If the measured range is less than the threshold, then the difference is not calculated

**Parameters**

| | |
|---|---|
| *thresh* | The threshold to use |

Definition at line 230 of file AngleOfArrivalCheck.hpp.

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

- include/pnt_integrity/AngleOfArrivalCheck.hpp

## 8.10 pnt_integrity::AoaCheckDiagnostics Struct Reference

Structure used to publish diagnostic data.

`#include <AngleOfArrivalCheck.hpp>`

### Public Attributes

- double singleDiffThresh

  *The threshold that is used when comparing single differences.*
- double unavailablePrnPercent

  *The number of PRNS that are unavailable (UNAVAILABLE)*
- double suspectPrnPercent

  *The number of PRNS that appear suspect (UNASSURED or INCONSISTENT)*
- double assuredPrnPercent

  *The number of PRNS that are assured (ASSURED)*
- double inconsistentThresh

  *The threshold used to check against the number of suspect PRNS.*
- double unassuredThresh

  *The threshold used to check against the number of suspect PRNS.*
- double assuredThresh

  *The threshold used to check against the number of assured PRNS.*

### 8.10.1 Detailed Description

Structure used to publish diagnostic data.

Definition at line 76 of file AngleOfArrivalCheck.hpp.

The documentation for this struct was generated from the following file:

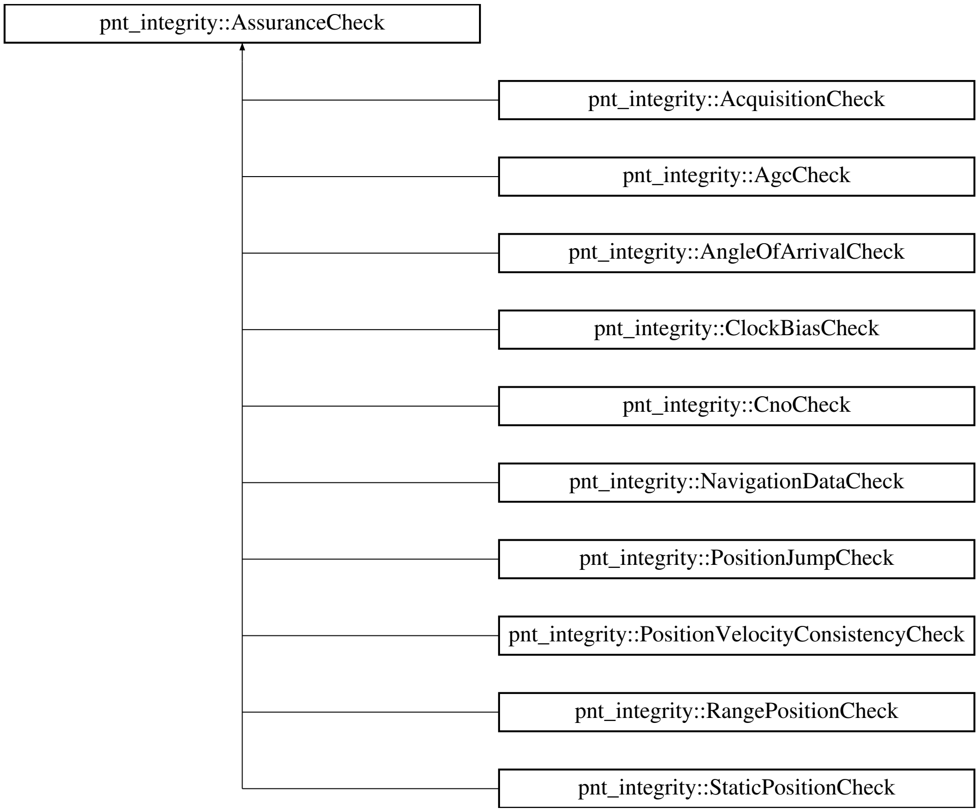
- include/pnt_integrity/AngleOfArrivalCheck.hpp

## 8.11 pnt_integrity::AssuranceCheck Class Reference

Parent class for all integrity checks.

`#include <AssuranceCheck.hpp>`

Inheritance diagram for pnt_integrity::AssuranceCheck:


pnt_integrity::AssuranceCheck
pnt_integrity::AcquisitionCheck
pnt_integrity::AgcCheck
pnt_integrity::AngleOfArrivalCheck
pnt_integrity::ClockBiasCheck
pnt_integrity::CnoCheck
pnt_integrity::NavigationDataCheck
pnt_integrity::PositionJumpCheck
pnt_integrity::PositionVelocityConsistencyCheck
pnt_integrity::RangePositionCheck
pnt_integrity::StaticPositionCheck

### Public Member Functions

- AssuranceCheck (const bool &multiPrnSupport=false, const std::string &checkName="AssuranceCheck", const logutils::LogCallback &log=logutils::printLogToStdOut)

  *Constructor.*
- virtual bool handleGnssObservables (const data::GNSSObservables &, const double &)

  *Handler function for GNSS Observables.*
- virtual bool handleGnssSubframe (const data::GNSSSubframe &)

  *Handler function for GNSS Subframes.*
- MultiPrnAssuranceMap getMultiPrnAssuranceData ()

  *Return function for the multi-prn assurance data.*
- virtual bool handlePositionVelocity (const data::PositionVelocity &, const bool &)

  *Handler function for Position / Velocity message.*
- virtual bool handleEstimatedPositionVelocity (const data::PositionVelocity &)

  *Handler function for an estimated Position / Velocity message.*

- virtual bool handleDistanceTraveled (const data::AccumulatedDistranceTraveled &)

  *Handler function for AccumulatedDistranceTraveled messages.*
- virtual bool handleMeasuredRange (const data::MeasuredRange &)

  *Handler function for measured range.*
- virtual bool handleIFSampleData (const double &, const if_data_utils::IFSampleData< if_data_utils::IFSample←SC8 > &)

  *Handler function for IF sample data (SC8)*
- virtual bool handleIFSampleData (const double &, const if_data_utils::IFSampleData< if_data_utils::IFSample←SC16 > &)

  *Handler function for IF sample data (SC16)*
- virtual bool handleClockOffset (const data::ClockOffset &)

  *Handler function for Clock Offset sample data.*
- virtual bool handleRfSpectrum (const data::RfSpectrum &)

  *Handler function for RF Spectrum value.*
- virtual bool handleAGC (const data::AgcValue &)

  *Handler function for AGC value.*
- data::AssuranceLevel getAssuranceLevel ()

  *Returns the AssuranceLevel enumeration value associated with the check's AssuranceState.*
- double getAssuranceValue ()

  *Returns the interger value associated with the check's AssuranceState.*
- data::AssuranceState getAssuranceState ()

  *Returns the AssuranceState of the check.*
- virtual void calculateAssuranceLevel (const double &time)=0

  *Function to calculate the assurance level of the check.*
- void setAssuranceThresholds (const double &inconsistentThresh, const double &unassuredThresh, const double &assuredThresh=std::numeric_limits< double >::quiet_NaN())

  *Sets the assurance level transition thresholds.*
- virtual bool runCheck ()=0

  *Triggers a manual check calculation.*
- void setLogMessageHandler (const logutils::LogCallback &logMsgHandler)

  *Sets the log message handler to provided callback.*
- void enableMultiPrnSupport ()

  *Enables support for multiple prn checks.*
- bool hasMultiPrnSupport ()

  *Returns value of multiPrnSupport_.*
- std::string getName ()

  *Returns the name of the check.*
- void changeAssuranceLevel (const double &updateTime, const data::AssuranceLevel &newLevel)

  *Changes the check's assurance level to the provided value.*
- void setAssuranceLevelPeriod (const double &levelPeriod)

  *Sets the assurance level period.*
- virtual void setLastGoodPosition (const double &updateTime, const data::GeodeticPosition3d &position)

  *Sets the last known good position.*
- virtual void clearLastGoodPosition ()

  *Clears the last known good position.*
- virtual void setPositionAssurance (const double &, const data::GeodeticPosition3d &, const data::AssuranceLevel &)

*Provides the check with an updated position and assuarance level.*
- void setWeight (const double &weightVal)

  *Sets the weight of the check.*
- double getWeight ()

  *Returns the weight for the check.*
- void setAllowPositiveWeighting (const bool &allowVal)

  *Sets the positive check weighting allowed boolean.*
- bool isCheckUsed ()

  *Returns whether or not the check's level should be weighted.*
- void reset ()

  *Reset the check state.*

## Static Protected Member Functions

- static double calculateDistance (const data::GeodeticPosition3d &pos1, const data::GeodeticPosition3d &pos2)

  *Computes the distance between two geodetic coordinates.*
- static bool checkDistance (const data::GeodeticPosition3d &pos1, const data::GeodeticPosition3d &pos2, const double &distanceThresh, double &distance)

  *Checks if the distance between two points is greater than the provided threshold.*
- static bool checkDistance (const double &dist, const double &thresh)

  *Compares the provided distance value with the threshold.*

## Protected Attributes

- std::recursive_mutex **assuranceCheckMutex_**
- logutils::LogCallback **logMsg_**
- MultiPrnAssuranceMap prnAssuranceLevels_
- double assuranceInconsistentThresh_
- double assuranceUnassuredThresh_
- double assuranceAssuredThresh_
- std::string checkName_

  *The name of the check.*
- double assuranceLevelPeriod_

  *The hold time for an elevated assurance level.*
- double lastAssuranceUpdate_

  *The last time the assurance level was updated.*
- data::GeodeticPosition3d lastKnownGoodPosition_

  *The last known good position set by external application.*
- double lastKnownGoodPositionTime_

  *The time associated with the last known good position.*
- double lastKnownGoodSet_

  *Flag to indicate that last known good has been set.*
- bool allowPositiveWeighting_

### 8.11.1 Detailed Description

Parent class for all integrity checks.

Pure virtual parent class that holds common functionality accross all assurance checks. Any child class that inherits from this must lock assuranceCheckMutex_ before access any protected data in this class. Any child class that inherits from this class can also use assuranceCheckMutex_ to protect private data in the child class.

Definition at line 59 of file AssuranceCheck.hpp.

### 8.11.2 Constructor & Destructor Documentation

#### 8.11.2.1 AssuranceCheck()

```
pnt_integrity::AssuranceCheck::AssuranceCheck (
             const bool & multiPrnSupport = false,
             const std::string & checkName = "AssuranceCheck",
             const logutils::LogCallback & log = logutils::printLogToStdOut )  [inline]
```

Constructor.

Constructor for the parent class. Multiple PRN support is disabled by default

**Parameters**

| | |
|---|---|
| *multiPrnSupport* | Constructor argument to enable / disable multiple PRN support |
| *checkName* | A string name identifier for the check |
| *log* | The provided log callback function |

Definition at line 71 of file AssuranceCheck.hpp.

### 8.11.3 Member Function Documentation

#### 8.11.3.1 calculateAssuranceLevel()

```
virtual void pnt_integrity::AssuranceCheck::calculateAssuranceLevel (
             const double & time )  [pure virtual]
```

Function to calculate the assurance level of the check.

Child classes should define this function to calculate the assurance level of the check by using whatever data / calculation necessary

Implemented in pnt_integrity::AcquisitionCheck, pnt_integrity::PositionJumpCheck, pnt_integrity::AngleOfArrivalCheck, pnt_integrity::ClockBiasCheck, pnt_integrity::StaticPositionCheck, pnt_integrity::NavigationDataCheck, pnt_integrity::RangePositionCheck, pnt_integrity::PositionVelocityConsistencyCheck, pnt_integrity::CnoCheck, and pnt_integrity::AgcCheck.

#### 8.11.3.2 calculateDistance()

```
static double pnt_integrity::AssuranceCheck::calculateDistance (
             const data::GeodeticPosition3d & pos1,
             const data::GeodeticPosition3d & pos2 )  [static], [protected]
```

Computes the distance between two geodetic coordinates.

**Parameters**

| | |
|---|---|
| *pos1* | The first position |
| *pos2* | The second position |

**Returns**

The calculated distance

#### 8.11.3.3 changeAssuranceLevel()

```
void pnt_integrity::AssuranceCheck::changeAssuranceLevel (
             const double & updateTime,
             const data::AssuranceLevel & newLevel )
```

Changes the check's assurance level to the provided value.

This function will change the assurance level of the check. Usually called by internal functions after a calculation based on provided assurance data. The function will raise the assurance level immediately if the provided level is higher than the current level. If the level is lower, the function will not change the level unless a certain period of time has passed since the last assurnace level upgrade (assuranceLevelPeriod_)

**Parameters**

| | |
|---|---|
| *updateTime* | The timestampe associated with the requested level change |
| *newLevel* | The newly provided / requested assurance level |

### 8.11.3.4 checkDistance() [1/2]

```
static bool pnt_integrity::AssuranceCheck::checkDistance (
            const data::GeodeticPosition3d & pos1,
            const data::GeodeticPosition3d & pos2,
            const double & distanceThresh,
            double & distance )  [inline], [static], [protected]
```

Checks if the distance between two points is greater than the provided threshold.

**Parameters**

| | |
|---|---|
| *pos1* | The first position |
| *pos2* | The second position |
| *distanceThresh* | The provided threshold do compare against |
| *distance* | The calculated distance |

**Returns**

True if distance is greater than provided threshold

Definition at line 522 of file AssuranceCheck.hpp.

### 8.11.3.5 checkDistance() [2/2]

```
static bool pnt_integrity::AssuranceCheck::checkDistance (
            const double & dist,
            const double & thresh )  [inline], [static], [protected]
```

Compares the provided distance value with the threshold.

**Parameters**

| | |
|---|---|
| *dist* | The provided distance |
| *thresh* | The threshold to compare against |

**Returns**

True if distance is greater than provided threshold

Definition at line 535 of file AssuranceCheck.hpp.

#### 8.11.3.6 getMultiPrnAssuranceData()

```
MultiPrnAssuranceMap pnt_integrity::AssuranceCheck::getMultiPrnAssuranceData ( )  [inline]
```

Return function for the multi-prn assurance data.

Returns the multiple-prn assurance levels for the check. An assertion is implemented to gaurantee that the function only returns the map when multi-prn support is enabled for the check.

**Returns**

The prn-to-assurance level map

Definition at line 120 of file AssuranceCheck.hpp.

#### 8.11.3.7 getWeight()

```
double pnt_integrity::AssuranceCheck::getWeight ( )  [inline]
```

Returns the weight for the check.

**Returns**

The weight for the check

Definition at line 404 of file AssuranceCheck.hpp.

#### 8.11.3.8 handleAGC()

```
virtual bool pnt_integrity::AssuranceCheck::handleAGC (
             const data::AgcValue &  )  [inline], [virtual]
```

Handler function for AGC value.

Function to handle provided AGC values (virtual)

**Returns**

True if successful

Reimplemented in pnt_integrity::AgcCheck.

Definition at line 215 of file AssuranceCheck.hpp.

#### 8.11.3.9 handleClockOffset()

```
virtual bool pnt_integrity::AssuranceCheck::handleClockOffset (
             const data::ClockOffset &  )  [inline], [virtual]
```

Handler function for Clock Offset sample data.

Function to handle provided Clock Offset data (virtual)

**Returns**

True if successful

Reimplemented in pnt_integrity::ClockBiasCheck.

Definition at line 195 of file AssuranceCheck.hpp.

#### 8.11.3.10 handleDistanceTraveled()

```
virtual bool pnt_integrity::AssuranceCheck::handleDistanceTraveled (
             const data::AccumulatedDistranceTraveled &  )  [inline], [virtual]
```

Handler function for AccumulatedDistranceTraveled messages.

**Returns**

True if successful

Reimplemented in pnt_integrity::PositionJumpCheck.

Definition at line 150 of file AssuranceCheck.hpp.

#### 8.11.3.11 handleEstimatedPositionVelocity()

```
virtual bool pnt_integrity::AssuranceCheck::handleEstimatedPositionVelocity (
             const data::PositionVelocity &  )  [inline], [virtual]
```

Handler function for an estimated Position / Velocity message.

Function to handle provided posivion / velocity messages (virtual)

**Returns**

True if successful

Reimplemented in pnt_integrity::PositionJumpCheck.

Definition at line 142 of file AssuranceCheck.hpp.

#### 8.11.3.12 handleGnssObservables()

```
virtual bool pnt_integrity::AssuranceCheck::handleGnssObservables (
             const data::GNSSObservables & ,
             const double &  )  [inline], [virtual]
```

Handler function for GNSS Observables.

Function to handle provided GNSS Observables. (virtual)

**Returns**

True if successful

Reimplemented in pnt_integrity::AngleOfArrivalCheck, and pnt_integrity::CnoCheck.

Definition at line 97 of file AssuranceCheck.hpp.

#### 8.11.3.13 handleGnssSubframe()

```
virtual bool pnt_integrity::AssuranceCheck::handleGnssSubframe (
             const data::GNSSSubframe &  )  [inline], [virtual]
```

Handler function for GNSS Subframes.

Function to handle provided GNSS Broadcast Nav. Data. (virtual)

**Returns**

True if successful

Reimplemented in pnt_integrity::NavigationDataCheck.

Definition at line 108 of file AssuranceCheck.hpp.

#### 8.11.3.14 handleIFSampleData() [1/2]

```
virtual bool pnt_integrity::AssuranceCheck::handleIFSampleData (
             const double & ,
             const if_data_utils::IFSampleData< if_data_utils::IFSampleSC8 > &  )  [inline],
[virtual]
```

Handler function for IF sample data (SC8)

Function to handle provided IF data (virtual)

**Returns**

True if successful

Reimplemented in pnt_integrity::AcquisitionCheck.

Definition at line 171 of file AssuranceCheck.hpp.

#### 8.11.3.15 handleIFSampleData() [2/2]

```
virtual bool pnt_integrity::AssuranceCheck::handleIFSampleData (
             const double & ,
             const if_data_utils::IFSampleData< if_data_utils::IFSampleSC16 > & ) [inline],
[virtual]
```

Handler function for IF sample data (SC16)

Function to handle provided IF data (virtual)

**Returns**

True if successful

Reimplemented in pnt_integrity::AcquisitionCheck.

Definition at line 183 of file AssuranceCheck.hpp.

#### 8.11.3.16 handleMeasuredRange()

```
virtual bool pnt_integrity::AssuranceCheck::handleMeasuredRange (
             const data::MeasuredRange & ) [inline], [virtual]
```

Handler function for measured range.

Function to handle provided range measurements (virtual)

**Returns**

True if successful

Reimplemented in pnt_integrity::RangePositionCheck.

Definition at line 161 of file AssuranceCheck.hpp.

#### 8.11.3.17 handlePositionVelocity()

```
virtual bool pnt_integrity::AssuranceCheck::handlePositionVelocity (
             const data::PositionVelocity & ,
             const bool & ) [inline], [virtual]
```

Handler function for Position / Velocity message.

Function to handle provided posivion / velocity messages (virtual)

**Returns**

True if successful

Reimplemented in pnt_integrity::StaticPositionCheck, pnt_integrity::PositionJumpCheck, pnt_integrity::RangePositionCheck, and pnt_integrity::PositionVelocityConsistencyCheck.

Definition at line 131 of file AssuranceCheck.hpp.

#### 8.11.3.18 handleRfSpectrum()

```
virtual bool pnt_integrity::AssuranceCheck::handleRfSpectrum (
             const data::RfSpectrum &  )  [inline], [virtual]
```

Handler function for RF Spectrum value.

Function to handle provided RF Spectrum values (virtual)

**Returns**

True if successful

Definition at line 205 of file AssuranceCheck.hpp.

#### 8.11.3.19 isCheckUsed()

```
bool pnt_integrity::AssuranceCheck::isCheckUsed ( )  [inline]
```

Returns whether or not the check's level should be weighted.

If the assurance level is Assured and postive weighting is not allowed, then this function will return false. It will also return false if the level is Unavailable

**Returns**

The flag to indicate if weighting should be used

Definition at line 428 of file AssuranceCheck.hpp.

#### 8.11.3.20 runCheck()

```
virtual bool pnt_integrity::AssuranceCheck::runCheck ( )  [pure virtual]
```

Triggers a manual check calculation.

Use this function to run a manual check calculation that is not triggered off the receipt of a message (pure virtual)

**Returns**

True if successful

Implemented in pnt_integrity::PositionJumpCheck, pnt_integrity::ClockBiasCheck, pnt_integrity::AngleOfArrivalCheck, pnt_integrity::StaticPositionCheck, pnt_integrity::NavigationDataCheck, pnt_integrity::PositionVelocityConsistencyCheck, pnt_integrity::CnoCheck, and pnt_integrity::RangePositionCheck.

#### 8.11.3.21 setAllowPositiveWeighting()

```
void pnt_integrity::AssuranceCheck::setAllowPositiveWeighting (
             const bool & allowVal )  [inline]
```

Sets the positive check weighting allowed boolean.

Positive weighting allows a check to increase the overall assurance level

**Parameters**

| | |
|---|---|
| *allowVal* | The flag indicating whether to allow positive weights |

Definition at line 415 of file AssuranceCheck.hpp.

### 8.11.3.22 setAssuranceLevelPeriod()

```
void pnt_integrity::AssuranceCheck::setAssuranceLevelPeriod (
             const double & levelPeriod )  [inline]
```

Sets the assurance level period.

The assurance level period is the amount of time required to hold an elevated assurance level.

**Parameters**

| | |
|---|---|
| *levelPeriod* | The period (in seconds) the the check is required to hold an elevated assurance level before lowering |

Definition at line 337 of file AssuranceCheck.hpp.

### 8.11.3.23 setAssuranceThresholds()

```
void pnt_integrity::AssuranceCheck::setAssuranceThresholds (
             const double & inconsistentThresh,
             const double & unassuredThresh,
             const double & assuredThresh = std::numeric_limits<double>::quiet_NaN() )  [inline]
```

Sets the assurance level transition thresholds.

Sets arbitrary thresholds that can be used in child classes to indicate or trigger a transition into different assurance levels associated with the check

**Parameters**

| | |
|---|---|
| *inconsistentThresh* | Use this threshold to trigger or indicate a transition into AssuranceLevel::Inconsistent |
| *unassuredThresh* | Use this value to trigger or indicate a transition into AssuranceLevel::Unassured |

Definition at line 258 of file AssuranceCheck.hpp.

#### 8.11.3.24 setLastGoodPosition()

```
virtual void pnt_integrity::AssuranceCheck::setLastGoodPosition (
             const double & updateTime,
             const data::GeodeticPosition3d & position )  [inline], [virtual]
```

Sets the last known good position.

Provides if the assurance check with knowledge of a last known good position for use in calculations (if needed by the specific implementation)

**Parameters**

| | |
|---|---|
| *updateTime* | The timestamp associated with the provided position |
| *position* | The last known good position |

Reimplemented in pnt_integrity::PositionJumpCheck.

Definition at line 356 of file AssuranceCheck.hpp.

#### 8.11.3.25 setLogMessageHandler()

```
void pnt_integrity::AssuranceCheck::setLogMessageHandler (
             const logutils::LogCallback & logMsgHandler )  [inline]
```

Sets the log message handler to provided callback.

**Parameters**

| | |
|---|---|
| *logMsgHandler* | The provided call back function |

Definition at line 292 of file AssuranceCheck.hpp.

#### 8.11.3.26 setPositionAssurance()

```
virtual void pnt_integrity::AssuranceCheck::setPositionAssurance (
             const double & ,
             const data::GeodeticPosition3d & ,
             const data::AssuranceLevel &  )  [inline], [virtual]
```

Provides the check with an updated position and assuarance level.

This method provides the check function with an updted position and associated assurance level for use in the check's calculation. The default behavior is null, but can be overridden in child classes.

Definition at line 384 of file AssuranceCheck.hpp.

#### 8.11.3.27 setWeight()

```
void pnt_integrity::AssuranceCheck::setWeight (
             const double & weightVal )  [inline]
```

Sets the weight of the check.

The weight of the check is used when combining the assurance level of this check with other checks for a cumulative assurance level

**Parameters**

| | |
|---|---|
| *weightVal* | The weight for this check |

Definition at line 395 of file AssuranceCheck.hpp.

### 8.11.4 Member Data Documentation

#### 8.11.4.1 allowPositiveWeighting_

```
bool pnt_integrity::AssuranceCheck::allowPositiveWeighting_  [protected]
```

flag to indicate if the check can be used in a positive weighting (i.e. do you weight the check when its level is assured)

Definition at line 505 of file AssuranceCheck.hpp.

#### 8.11.4.2 assuranceAssuredThresh_

```
double pnt_integrity::AssuranceCheck::assuranceAssuredThresh_  [protected]
```

The arbitrary threshold for elevating the check's overall assurance level to AssuranceLevel::Assured. It is up to the AssuranceCheck implementation on how this threshold is used internally.

Definition at line 483 of file AssuranceCheck.hpp.

#### 8.11.4.3 assuranceInconsistentThresh_

`double pnt_integrity::AssuranceCheck::assuranceInconsistentThresh_ [protected]`

The arbitrary threshold for elevating the check's overall assurance level to AssuranceLevel::Inconsistent. It is up to the AssuranceCheck implementation on how this threshold is used internally.

Definition at line 473 of file AssuranceCheck.hpp.

#### 8.11.4.4 assuranceUnassuredThresh_

`double pnt_integrity::AssuranceCheck::assuranceUnassuredThresh_ [protected]`

The arbitrary threshold for elevating the check's overall assurance level to AssuranceLevel::Unassured. It is up to the AssuranceCheck implementation on how this threshold is used internally.

Definition at line 478 of file AssuranceCheck.hpp.

#### 8.11.4.5 prnAssuranceLevels_

`MultiPrnAssuranceMap pnt_integrity::AssuranceCheck::prnAssuranceLevels_ [protected]`

The assurance level for each PRN (if applicable to the defined check). Should only be populated if enableMultiPrnSupport() has been called

Definition at line 468 of file AssuranceCheck.hpp.

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

- include/pnt_integrity/AssuranceCheck.hpp

## 8.12 pnt_integrity::data::AssuranceReport Struct Reference

A structure to hold a single assurance report.

`#include <IntegrityData.hpp>`

### Public Attributes

- Header header

  *The header for the structure message.*
- AssuranceState state

  *The assurance state.*

### 8.12.1 Detailed Description

A structure to hold a single assurance report.

Definition at line 381 of file IntegrityData.hpp.

The documentation for this struct was generated from the following file:

- include/pnt_integrity/IntegrityData.hpp

## 8.13 pnt_integrity::data::AssuranceReports Struct Reference

A structure to hold assurance data for all registered checks.

`#include <IntegrityData.hpp>`

### Public Member Functions

- AssuranceReports ()

  *Default constructor for the struct. Initializes numStates to 0.*
- void addReport (const AssuranceState &state)

  *Adds a reported state to the vector, increments count.*

### Public Attributes

- Header header

  *The header for the structure message.*
- long numStates

  *Number of assurance states reported.*
- std::vector< AssuranceState > states

  *A vector of AssuranceState, length numStates.*

### 8.13.1 Detailed Description

A structure to hold assurance data for all registered checks.

Definition at line 391 of file IntegrityData.hpp.

The documentation for this struct was generated from the following file:

- include/pnt_integrity/IntegrityData.hpp

## 8.14 pnt_integrity::data::AssuranceState Class Reference

A structure to hold an AssuranceLevel and value.

`#include <IntegrityData.hpp>`

### Public Member Functions

- bool setWithValue (const double &valueIn)

    *Sets the state with a provided value.*
- bool setWithLevel (const AssuranceLevel &levelIn)

    *Sets the state with a provided enumeration.*
- double getAssuranceValue ()

    *Retrieves the internal assurance value.*
- data::AssuranceLevel getAssuranceLevel ()

    *Retrieves the internal assurance level.*
- int getIntegerAssuranceValue () const

    *Retrieves the internal assurance value as an integer.*
- void setWeight (const double &weight)

    *Sets the weight associated with the state.*
- double getWeight ()

    *Retrieves the weight associated with the state.*
- void setName (const std::string &name)

    *Sets the string name of the state.*
- std::string getName ()

    *Retrieves the name of the check.*

### 8.14.1 Detailed Description

A structure to hold an AssuranceLevel and value.

A structure for holding the idea of assurance both as an enumeration and separate numeric value.

Definition at line 266 of file IntegrityData.hpp.

### 8.14.2 Member Function Documentation

#### 8.14.2.1 getAssuranceLevel()

`data::AssuranceLevel pnt_integrity::data::AssuranceState::getAssuranceLevel ( ) [inline]`

Retrieves the internal assurance level.

**Returns**

The assurance level

Definition at line 344 of file IntegrityData.hpp.

#### 8.14.2.2 getAssuranceValue()

`double pnt_integrity::data::AssuranceState::getAssuranceValue ( ) [inline]`

Retrieves the internal assurance value.

**Returns**

The assurance value

Definition at line 340 of file IntegrityData.hpp.

#### 8.14.2.3 getIntegerAssuranceValue()

`int pnt_integrity::data::AssuranceState::getIntegerAssuranceValue ( ) const [inline]`

Retrieves the internal assurance value as an integer.

**Returns**

An integer representation of the assurance value

Definition at line 348 of file IntegrityData.hpp.

#### 8.14.2.4 getName()

```
std::string pnt_integrity::data::AssuranceState::getName ( ) [inline]
```

Retrieves the name of the check.

**Returns**

The string name of the check

Definition at line 362 of file IntegrityData.hpp.

#### 8.14.2.5 getWeight()

```
double pnt_integrity::data::AssuranceState::getWeight ( ) [inline]
```

Retrieves the weight associated with the state.

**Returns**

The weight value

Definition at line 355 of file IntegrityData.hpp.

#### 8.14.2.6 setWithLevel()

```
bool pnt_integrity::data::AssuranceState::setWithLevel (
            const AssuranceLevel & levelIn ) [inline]
```

Sets the state with a provided enumeration.

This function will set the AssuranceState's level enumeration to the provided level and the value is set accordingly. The function returns false if the level is "Unavailable" to indicate that it should not be used in any cumulative calculations.

levelIn The provided level enumeration

Definition at line 318 of file IntegrityData.hpp.

#### 8.14.2.7 setWithValue()

```
bool pnt_integrity::data::AssuranceState::setWithValue (
            const double & valueIn ) [inline]
```

Sets the state with a provided value.

This method allows the state to be set by an arbritrary value which is usually produced by a weighting function. The provided assurance value will be rounded to an integer and then thresholded to the appropriate value and the level enumeration is set appropriately.

**Parameters**

| | |
|---|---|
| *value↩ In* | The provided value. |

Definition at line 283 of file IntegrityData.hpp.

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

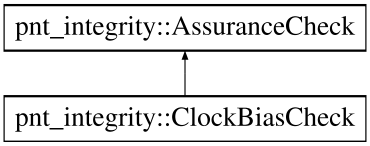
- include/pnt_integrity/IntegrityData.hpp

## 8.15 pnt_integrity::ClockBiasCheck Class Reference

Class implementation for the position velocity check.

`#include <ClockBiasCheck.hpp>`

Inheritance diagram for pnt_integrity::ClockBiasCheck:



### Public Member Functions

- ClockBiasCheck (const std::string &name="Clock Bias Check", const unsigned int &minNumSamples=10, const unsigned int &maxNumSamples=30, const double &minSampleTimeSec=10.0, const double &driftRate↩Bound=5e-7, const double &driftRateVarBound=1e-6, const logutils::LogCallback &log=logutils::printLogToStd↩Out)

  *Default constructor for the check class.*
- bool handleClockOffset (const data::ClockOffset &clockOffset)

  *Handler function for clock offset (bias and drift)*
- void calculateAssuranceLevel (const double &)

  *Function to explicitly set the assurance level of the check.*
- bool runCheck ()

  *Triggers a manual check calculation.*
- void setPublishDiagnostics (std::function< void(const double &timestamp, const ClockBiasCheckDiagnostics &checkData)> handler)

  *Connects the internal publishing function to external interface.*

## Additional Inherited Members

## 8.15.1 Detailed Description

Class implementation for the position velocity check.

The Clock Bias Check calculates the expectation and variance of the clock drift for the most recent set of clock samples, minus the most recent sample. The expectation is used to propagate the clock forward to the most recent single sample's arrival time and check if it is within reasonable bounds. The variance is used to check for zero-bias disruption. The expectation and variance are calculated like normal, using the drift value in each sample. The propagated sample's clock offset is calculated by multiplying the drift expectation by the sample time difference between the second to most recent sample and the most recent sample (called dt), and then adding the second to most recent sample's clock offset. (i.e. velocity∗dt + last position). The propagated sample's clock offset is subtracted from the most recent sample's clock offset (and then run through the absolute value function) to obtain the offset error. If the offset error is greater than the drift rate bound multiplied by dt, then the clock bias check returns Unassured. Else if the clock drift variance is greater than the predetermined drift variance bound, then it returns Inconsistent.

Definition at line 117 of file ClockBiasCheck.hpp.

## 8.15.2 Constructor & Destructor Documentation

### 8.15.2.1 ClockBiasCheck()

```
pnt_integrity::ClockBiasCheck::ClockBiasCheck (
             const std::string & name = "Clock Bias Check",
             const unsigned int & minNumSamples = 10,
             const unsigned int & maxNumSamples = 30,
             const double & minSampleTimeSec = 10.0,
             const double & driftRateBound = 5e-7,
             const double & driftRateVarBound = 1e-6,
             const logutils::LogCallback & log = logutils::printLogToStdOut )  [inline]
```

Default constructor for the check class.

Constructor explicitly disables multi-prn support.

**Parameters**

| | |
|---|---|
| *name* | The name of the check object |
| *minNumSamples* | The minimum number of samples required for the check |
| *maxNumSamples* | The maximum number of samples required for the check |
| *minSampleTimeSec* | The duration of time (in seconds) over which to |
| *driftRateBound* | Maximum allowable drift rate |
| *driftRateVarBound* | Maximum allowable drift rate variance get clock data for checking integrity |
| *log* | A provided log callback function to use |

Definition at line 132 of file ClockBiasCheck.hpp.

## 8.15.3 Member Function Documentation

### 8.15.3.1 calculateAssuranceLevel()

```
void pnt_integrity::ClockBiasCheck::calculateAssuranceLevel (
            const double & ) [inline], [virtual]
```

Function to explicitly set the assurance level of the check.

Uses whatever data is available to calculate the assurance level

Implements pnt_integrity::AssuranceCheck.

Definition at line 170 of file ClockBiasCheck.hpp.

### 8.15.3.2 handleClockOffset()

```
bool pnt_integrity::ClockBiasCheck::handleClockOffset (
            const data::ClockOffset & clockOffset ) [virtual]
```

Handler function for clock offset (bias and drift)

Function to handle provided clock offset.

**Parameters**

| | |
|---|---|
| *clockOffset* | The provided clock offset message |

**Returns**

True if successful

Reimplemented from pnt_integrity::AssuranceCheck.

### 8.15.3.3 runCheck()

```
bool pnt_integrity::ClockBiasCheck::runCheck ( ) [virtual]
```

Triggers a manual check calculation.

Use this function to run a manual check calculation that is not triggered off the receipt of a message (pure virtual)

**Returns**

True if successful

Implements pnt_integrity::AssuranceCheck.

#### 8.15.3.4 setPublishDiagnostics()

```
void pnt_integrity::ClockBiasCheck::setPublishDiagnostics (
            std::function< void(const double &timestamp, const ClockBiasCheckDiagnostics &check↩
Data)> handler )  [inline]
```

Connects the internal publishing function to external interface.

This function connects the internal "publishDiagnostics" function to an external, custom function of choice

**Parameters**

| | |
|---|---|
| *handler* | Provided handler function |

Definition at line 186 of file ClockBiasCheck.hpp.

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

- include/pnt_integrity/ClockBiasCheck.hpp

## 8.16 pnt_integrity::ClockBiasCheckDiagnostics Struct Reference

Structure used to publish diagnostic data.

`#include <ClockBiasCheck.hpp>`

### Public Attributes

- double expectedDrift

  *The excpected drift based on the recent history.*
- double expectedDriftVar

  *The expected variance of the drift based on recent history.*
- double propagatedOffset

  *The clock bias propagated to the current time step.*
- double actualOffset

  *The actual clock bias at the current time stamp.*
- double offsetError

  *The error threshold for comparing the actual and propagated.*
- double driftRateBound

  *The error bound on the drift rate.*
- double driftRateVarBound

  *The error baound on the drift rate variance.*

### 8.16.1 Detailed Description

Structure used to publish diagnostic data.

Definition at line 73 of file ClockBiasCheck.hpp.

The documentation for this struct was generated from the following file:

- include/pnt_integrity/ClockBiasCheck.hpp

## 8.17 pnt_integrity::data::ClockOffset Struct Reference

A structure for measuring the offset between two clocks.

`#include <IntegrityData.hpp>`

### Public Member Functions

- ClockOffset (const Header &headerIn=Header())

  *Constructor for the ClockOffset structure.*

### Public Attributes

- Header header

  *A header for the message.*
- int8_t timecode1
- int8_t timecode2
- double offset

  *Time offset between the two clocks (sec)*
- double drift

  *The drift between the two clocks (sec / sec)*
- double covariance [2][2]

  *The measurement covariance of the offset parameters (2x2 matrix)*

### 8.17.1 Detailed Description

A structure for measuring the offset between two clocks.

Definition at line 145 of file IntegrityData.hpp.

### 8.17.2 Constructor & Destructor Documentation

#### 8.17.2.1 ClockOffset()

```
pnt_integrity::data::ClockOffset::ClockOffset (
             const Header & headerIn = Header() )  [inline]
```

Constructor for the ClockOffset structure.

The default constructor for the clock offset message can be optionally provided with a pre-built header. The offset, drift, and time error covariance is initialized to NaN and must be set to desired values after object construction. Timecodes are initialized to -1 and must also be set after construction to desired values

**Parameters**

| | |
|---|---|
| *header←*<br>*In* | A provided header object |

Definition at line 176 of file IntegrityData.hpp.

## 8.17.3 Member Data Documentation

### 8.17.3.1 timecode1

`int8_t pnt_integrity::data::ClockOffset::timecode1`

Indicator for clock 1 timebase, 0 if synced to TAI, non-zero if device using a specific timebase

Definition at line 152 of file IntegrityData.hpp.

### 8.17.3.2 timecode2

`int8_t pnt_integrity::data::ClockOffset::timecode2`

Indicator for clock 2 timebase, 0 if synced to TAI, non-zero if device using a specific timebase

Definition at line 156 of file IntegrityData.hpp.

The documentation for this struct was generated from the following file:

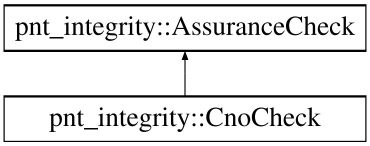
- include/pnt_integrity/IntegrityData.hpp

## 8.18 pnt_integrity::CnoCheck Class Reference

Class implementation of the carrier-to-noise (CnO) assurance check. The check analyzes the CnO values for abnormalities.

`#include <CnoCheck.hpp>`

Inheritance diagram for pnt_integrity::CnoCheck:



## Public Member Functions

- CnoCheck (const std::string &name="Cno Check", const size_t &cnoFilterWindow=10, const logutils::LogCallback &log=logutils::printLogToStdOut)

  *Constructor for the CnoCheck object.*
- bool handleGnssObservables (const data::GNSSObservables &gnssObs, const double &time=0)

  *Handler function for GNSS Observables.*
- void calculateAssuranceLevel (const double &)

  *Function to explicitly set the assurance level of the check.*
- virtual bool runCheck ()

  *Triggers a manual check calculation.*
- void setFilterWindow (const size_t &windowSize)

  *Sets the time filter window for CnO analysis.*
- void setPublishDiagnostics (std::function< void(const double &timestamp, const CnoCheckDiagnostics &checkData)> handler)

  *Connects the internal publishing function to external interface.*

## Additional Inherited Members

### 8.18.1 Detailed Description

Class implementation of the carrier-to-noise (CnO) assurance check. The check analyzes the CnO values for abnormalities.

Definition at line 68 of file CnoCheck.hpp.

### 8.18.2 Constructor & Destructor Documentation

#### 8.18.2.1 CnoCheck()

```
pnt_integrity::CnoCheck::CnoCheck (
            const std::string & name = "Cno Check",
            const size_t & cnoFilterWindow = 10,
            const logutils::LogCallback & log = logutils::printLogToStdOut )  [inline]
```

Constructor for the CnoCheck object.

**Parameters**

| | |
|---|---|
| *name* | The name identifier of the check |
| *cnoFilterWindow* | The time window across which CnO values are analyzed |
| *log* | A provided log callback function to use |

Definition at line 77 of file CnoCheck.hpp.

## 8.18.3 Member Function Documentation

### 8.18.3.1 calculateAssuranceLevel()

```
void pnt_integrity::CnoCheck::calculateAssuranceLevel (
             const double &  )  [inline], [virtual]
```

Function to explicitly set the assurance level of the check.

For this check, this function cycles through all of the individual PRN assurance values, analyzes them, and then sets the master assurance level associted with the check.

Implements pnt_integrity::AssuranceCheck.

Definition at line 108 of file CnoCheck.hpp.

### 8.18.3.2 handleGnssObservables()

```
bool pnt_integrity::CnoCheck::handleGnssObservables (
             const data::GNSSObservables & gnssObs,
             const double & time = 0 )  [virtual]
```

Handler function for GNSS Observables.

Function to handle provided GNSS Observables. This function simply calls runCheck(), as the provided data has already been added to the repository

**Parameters**

| | |
|---|---|
| *gnssObs* | The provided GNSS observable data |

**Returns**

True if successful

Reimplemented from pnt_integrity::AssuranceCheck.

#### 8.18.3.3 runCheck()

```
virtual bool pnt_integrity::CnoCheck::runCheck ( )  [virtual]
```

Triggers a manual check calculation.

Use this function to run a manual check calculation that is not triggered off the receipt of a message (pure virtual)

**Returns**

True if successful

Implements pnt_integrity::AssuranceCheck.

#### 8.18.3.4 setFilterWindow()

```
void pnt_integrity::CnoCheck::setFilterWindow (
             const size_t & windowSize )  [inline]
```

Sets the time filter window for CnO analysis.

**Parameters**

| | |
|---|---|
| *windowSize* | The time of the analysis window to use |

Definition at line 121 of file CnoCheck.hpp.

#### 8.18.3.5 setPublishDiagnostics()

```
void pnt_integrity::CnoCheck::setPublishDiagnostics (
             std::function< void(const double &timestamp, const CnoCheckDiagnostics &checkData)>
handler )  [inline]
```

Connects the internal publishing function to external interface.

This function connects the internal "publishDiagnostics" function to an external, custom function of choice

**Parameters**

| | |
|---|---|
| *handler* | Provided handler function |

Definition at line 133 of file CnoCheck.hpp.

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

- include/pnt_integrity/CnoCheck.hpp

## 8.19 pnt_integrity::CnoCheckDiagnostics Struct Reference

Diagnostic data for the check.

`#include <CnoCheck.hpp>`

### Public Attributes

- int averageCount

  *the number of PRNs within 1 unit of the mode*
- double inconsistentThresh

  *The threshold for the inconsistent assurance level.*
- double unassuredThresh

  *The threshold for the unassured assurance level.*

### 8.19.1 Detailed Description

Diagnostic data for the check.

Definition at line 56 of file CnoCheck.hpp.

The documentation for this struct was generated from the following file:

- include/pnt_integrity/CnoCheck.hpp

## 8.20 pnt_integrity::EphemerisParameters Struct Reference

`#include <GPSEphemeris.hpp>`

## Public Attributes

- uint16_t **prn**
- AlertFlag **alertFlag**
- AntiSpoofFlag **asFlag**
- uint32_t **towSf1**
- uint32_t **towM1**
- uint16_t **weekNumber**
- L2CodeType **codeOnL2**
- uint16_t **uraIndex**
- SVHealth **svHealth**
- uint16_t **iodc**
- L2NavDataFlag **l2PDataFlag**
- double **groupDelay**
- double **clockCorrectionTime**
- double **clockAging3**
- double **clockAging2**
- double **clockAging1**
- double **inPhaseInterSignalCorrection**
- double **quadratureInterSignalCorrection**
- uint32_t **towSf2**
- uint32_t **towM2**
- uint16_t **iodeSf2**
- double **sinOrbitRadius**
- double **meanMotionDifference**
- double **meanMotionDifferenceRate**
- double **meanAnomaly**
- double **cosLatitude**
- double **eccentricity**
- double **sinLatitude**
- double **sqrtSemiMajorAxis**
- double **semiMajorAxisDifference**
- double **semiMajorAxisRate**
- double **timeOfEphemeris**
- FitInterval **fitInterval**
- uint16_t **ageOfDataOffset**
- uint32_t **towSf3**
- uint32_t **towM3**
- double **cosInclination**
- double **rightAscension**
- double **ascensionRateDifference**
- double **sinInclination**
- double **inclinationAngle**
- double **cosOrbitRadius**
- double **argumentOfPerigee**
- double **ascensionRate**
- uint16_t **iodeSf3**
- double **inclinationRate**

### 8.20.1 Detailed Description

Structure to hold the ephemeris parameters as provided in subframes 1 - 3 of IS-GPS-200

Definition at line 116 of file GPSEphemeris.hpp.

The documentation for this struct was generated from the following file:

- include/pnt_integrity/GPSEphemeris.hpp

## 8.21 pnt_integrity::AlmanacSubframeFaults::FaultType Struct Reference

### Public Attributes

- uint16_t **prn**: 1
- uint16_t **tow**: 1
- uint16_t **svHealth**: 1
- uint16_t **eccentricity**: 1
- uint16_t **toa**: 1
- uint16_t **deltaI**: 1
- uint16_t **omegaDot**: 1
- uint16_t **sqrtA**: 1
- uint16_t **omega0**: 1
- uint16_t **omega**: 1
- uint16_t **m0**: 1
- uint16_t **af0**: 1
- uint16_t **af1**: 1
- uint16_t **referenceWeek**: 1

### 8.21.1 Detailed Description

Definition at line 96 of file GPSAlmanac.hpp.

The documentation for this struct was generated from the following file:

- include/pnt_integrity/GPSAlmanac.hpp

## 8.22 geodetic_converter::GeodeticConverter Class Reference

Class to implement gedetic conversions for the pnt_integrity library.

`#include <GeodeticConverter.hpp>`

## Public Member Functions

- GeodeticConverter ()

  *Constructor for converter object.*
- ∼GeodeticConverter ()

  *Destructor for the converter object.*
- bool isInitialised ()

  *Returns the reference flag.*
- void getReference (double ∗latitude, double ∗longitude, double ∗altitude)

  *Returns the reference position.*
- void initialiseReference (const double latitude, const double longitude, const double altitude)

  *Sets the reference position.*
- void geodetic2Ecef (const double latitude, const double longitude, const double altitude, double ∗x, double ∗y, double ∗z)

  *Converts the provided LLA to ECEF.*
- void ecef2Geodetic (const double x, const double y, const double z, double ∗latitude, double ∗longitude, double ∗altitude)

  *Converts the provided ECEF to LLA.*
- void ecef2Ned (const double x, const double y, const double z, double ∗north, double ∗east, double ∗down)

  *Converts the provided ECEF to NED.*
- void ned2Ecef (const double north, const double east, const double down, double ∗x, double ∗y, double ∗z)

  *Converts the provided NED to ECEF.*
- void geodetic2Ned (const double latitude, const double longitude, const double altitude, double ∗north, double ∗east, double ∗down)

  *Converts the provided LLA to NED.*
- void ned2Geodetic (const double north, const double east, const double down, double ∗latitude, double ∗longitude, double ∗altitude)

  *Converts the provided NED to LLA.*
- void geodetic2Enu (const double latitude, const double longitude, const double altitude, double ∗east, double ∗north, double ∗up)

  *Converts the provided LLA to ENU.*
- void enu2Geodetic (const double east, const double north, const double up, double ∗latitude, double ∗longitude, double ∗altitude)

  *Converts the provided ENU to LLA.*

### 8.22.1 Detailed Description

Class to implement gedetic conversions for the pnt_integrity library.

Definition at line 61 of file GeodeticConverter.hpp.

### 8.22.2 Constructor & Destructor Documentation

#### 8.22.2.1 GeodeticConverter()

```
geodetic_converter::GeodeticConverter::GeodeticConverter ( )  [inline]
```

Constructor for converter object.

Constructor initializes the reference flag to false.

Definition at line 67 of file GeodeticConverter.hpp.

### 8.22.3 Member Function Documentation

#### 8.22.3.1 ecef2Geodetic()

```
void geodetic_converter::GeodeticConverter::ecef2Geodetic (
             const double x,
             const double y,
             const double z,
             double * latitude,
             double * longitude,
             double * altitude )  [inline]
```

Converts the provided ECEF to LLA.

**Parameters**

| | |
|---|---|
| *latitude* | Latitude in radians |
| *longitude* | Longitude in radians |
| *altitude* | Altitude in meters |
| *x* | The ECEF X psoition in meters |
| *y* | The ECEF Y position in meters |
| *z* | The ECEF Z position in meters |

Definition at line 165 of file GeodeticConverter.hpp.

#### 8.22.3.2 ecef2Ned()

```
void geodetic_converter::GeodeticConverter::ecef2Ned (
             const double x,
             const double y,
             const double z,
             double * north,
             double * east,
             double * down )  [inline]
```

Converts the provided ECEF to NED.

**Parameters**

| | |
|---|---|
| *east* | NED east in meters |
| *north* | NED north in meters |
| *down* | NED down in meters |
| *x* | The ECEF X psoition in meters |
| *y* | The ECEF Y position in meters |
| *z* | The ECEF Z position in meters |

Definition at line 213 of file GeodeticConverter.hpp.

#### 8.22.3.3 enu2Geodetic()

```
void geodetic_converter::GeodeticConverter::enu2Geodetic (
            const double east,
            const double north,
            const double up,
            double * latitude,
            double * longitude,
            double * altitude )  [inline]
```

Converts the provided ENU to LLA.

**Parameters**

| | |
|---|---|
| *latitude* | Latitude in radians |
| *longitude* | Longitude in radians |
| *altitude* | Altitude in meters |
| *east* | ENU east in meters |
| *north* | ENU north in meters |
| *up* | ENU up in meters |

Definition at line 336 of file GeodeticConverter.hpp.

#### 8.22.3.4 geodetic2Ecef()

```
void geodetic_converter::GeodeticConverter::geodetic2Ecef (
            const double latitude,
            const double longitude,
            const double altitude,
            double * x,
            double * y,
            double * z )  [inline]
```

Converts the provided LLA to ECEF.

**Parameters**

| | |
|---|---|
| *latitude* | Latitude in radians |
| *longitude* | Longitude in radians |
| *altitude* | Altitude in meters |
| *x* | The ECEF X psoition in meters |
| *y* | The ECEF Y position in meters |
| *z* | The ECEF Z position in meters |

Definition at line 138 of file GeodeticConverter.hpp.

#### 8.22.3.5 geodetic2Enu()

```
void geodetic_converter::GeodeticConverter::geodetic2Enu (
             const double latitude,
             const double longitude,
             const double altitude,
             double * east,
             double * north,
             double * up )  [inline]
```

Converts the provided LLA to ENU.

**Parameters**

| | |
|---|---|
| *latitude* | Latitude in radians |
| *longitude* | Longitude in radians |
| *altitude* | Altitude in meters |
| *east* | ENU east in meters |
| *north* | ENU north in meters |
| *up* | ENU up in meters |

Definition at line 309 of file GeodeticConverter.hpp.

#### 8.22.3.6 geodetic2Ned()

```
void geodetic_converter::GeodeticConverter::geodetic2Ned (
             const double latitude,
             const double longitude,
             const double altitude,
             double * north,
             double * east,
             double * down )  [inline]
```

Converts the provided LLA to NED.

**Parameters**

| | |
|---|---|
| *latitude* | Latitude in radians |
| *longitude* | Longitude in radians |
| *altitude* | Altitude in meters |
| *east* | NED east in meters |
| *north* | NED north in meters |
| *down* | NED down in meters |

Definition at line 267 of file GeodeticConverter.hpp.

#### 8.22.3.7 getReference()

```
void geodetic_converter::GeodeticConverter::getReference (
             double * latitude,
             double * longitude,
             double * altitude )  [inline]
```

Returns the reference position.

Returns the reference position with the latitude / longitude in radians and altitude in meters

**Parameters**

| | |
|---|---|
| *latitude* | Latitude in radians |
| *longitude* | Longitude in radians |
| *altitude* | Altitude in meters |

Definition at line 88 of file GeodeticConverter.hpp.

#### 8.22.3.8 initialiseReference()

```
void geodetic_converter::GeodeticConverter::initialiseReference (
            const double latitude,
            const double longitude,
            const double altitude ) [inline]
```

Sets the reference position.

Sets the reference to the provided position (LLA)

**Parameters**

| | |
|---|---|
| *latitude* | Latitude in radians |
| *longitude* | Longitude in radians |
| *altitude* | Altitude in meters |

Definition at line 102 of file GeodeticConverter.hpp.

#### 8.22.3.9 isInitialised()

```
bool geodetic_converter::GeodeticConverter::isInitialised ( ) [inline]
```

Returns the reference flag.

Returns a flag to indicate if the converter's reference position has been set.

Definition at line 78 of file GeodeticConverter.hpp.

#### 8.22.3.10 ned2Ecef()

```
void geodetic_converter::GeodeticConverter::ned2Ecef (
            const double north,
            const double east,
            const double down,
            double * x,
            double * y,
            double * z ) [inline]
```

Converts the provided NED to ECEF.

**Parameters**

| | |
|---|---|
| *east* | NED east in meters |
| *north* | NED north in meters |
| *down* | NED down in meters |
| *x* | The ECEF X psoition in meters |
| *y* | The ECEF Y position in meters |
| *z* | The ECEF Z position in meters |

Definition at line 241 of file GeodeticConverter.hpp.

#### 8.22.3.11 ned2Geodetic()

```
void geodetic_converter::GeodeticConverter::ned2Geodetic (
             const double north,
             const double east,
             const double down,
             double * latitude,
             double * longitude,
             double * altitude )  [inline]
```

Converts the provided NED to LLA.

**Parameters**

| | |
|---|---|
| *latitude* | Latitude in radians |
| *longitude* | Longitude in radians |
| *altitude* | Altitude in meters |
| *east* | NED east in meters |
| *north* | NED north in meters |
| *down* | NED down in meters |

Definition at line 288 of file GeodeticConverter.hpp.

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

- include/pnt_integrity/GeodeticConverter.hpp

## 8.23 pnt_integrity::data::GeodeticPosition3d Struct Reference

A structure to represent 3D geodetic position.

`#include <IntegrityData.hpp>`

### Public Member Functions

- GeodeticPosition3d (const double &latIn=std::numeric_limits< double >::quiet_NaN(), const double &lonIn=std::numeric_limits< double >::quiet_NaN(), const double &altIn=std::numeric_limits< double >::quiet_NaN())

  *Constructor for the 3D geodetic position.*
- bool getECEF (double *ecef) const

  *Returns the coordinates from lla to ecef using WGS84.*

## Public Attributes

- double latitude

  *The latitude in radians.*
- double longitude

  *The longitude in radians.*
- double altitude

  *The altitude in meters above the WGS-84 ellipsoid.*

### 8.23.1 Detailed Description

A structure to represent 3D geodetic position.

This structure represents that latitude, longitude, and altitude of a geodetic position

Definition at line 597 of file IntegrityData.hpp.

### 8.23.2 Constructor & Destructor Documentation

#### 8.23.2.1 GeodeticPosition3d()

```
pnt_integrity::data::GeodeticPosition3d::GeodeticPosition3d (
            const double & latIn = std::numeric_limits<double>::quiet_NaN(),
            const double & lonIn = std::numeric_limits<double>::quiet_NaN(),
            const double & altIn = std::numeric_limits<double>::quiet_NaN() )  [inline]
```

Constructor for the 3D geodetic position.

**Parameters**

| | |
|---|---|
| *latIn* | The latitude of the 3d position (radians) |
| *lonIn* | The longitude of the 3d position (radians) |
| *altIn* | The altitude of the 3d position (meters above WGS-84) |

Definition at line 611 of file IntegrityData.hpp.

### 8.23.3 Member Function Documentation

#### 8.23.3.1 getECEF()

```
bool pnt_integrity::data::GeodeticPosition3d::getECEF (
            double * ecef ) const  [inline]
```

Returns the coordinates from lla to ecef using WGS84.

**Parameters**

| | |
|---|---|
| *ecef* | The output location for the generated coordinates, must be at least 3∗sizeof(double) large |

**Returns**

false if any values are NaN, true otherwise

Definition at line 623 of file IntegrityData.hpp.

The documentation for this struct was generated from the following file:

- include/pnt_integrity/IntegrityData.hpp

## 8.24 pnt_integrity::data::GNSSObservable Struct Reference

A structure for GNSS observables (pseudorange, carrier, doppler, etc)

`#include <IntegrityData.hpp>`

### Public Member Functions

- GNSSObservable (const uint16_t &prnIn=0, const SatelliteSystem &satTypeIn=SatelliteSystem::Other, const CodeType &codeTypeIn=CodeType::SigBLANK, const FrequencyBand &freqTypeIn=FrequencyBand::Band0, const AssuranceLevel &assuranceLevelIn=AssuranceLevel::Unavailable, const double &cnoIn=std::numeric_limits< double >::quiet_NaN(), const bool &psrValIn=false, const double &psrIn=std::numeric_limits< double >::quiet_NaN(), const double &psrVarIn=std::numeric_limits< double >::quiet_NaN(), const bool &doppValIn=false, const double &doppIn=std::numeric_limits< double >::quiet_NaN(), const double &doppVarIn=std::numeric_limits< double >::quiet_NaN(), const bool &cpValIn=false, const double &cpIn=std::numeric_limits< double >::quiet_NaN(), const double &cpVarIn=std::numeric_limits< double >::quiet_NaN(), const bool &lossOfLockIn=false)

  *Default constructor to initialize values.*
- uint64_t getUniqueID ()

  *Returns a unique identifier for the observable.*

## Public Attributes

- uint16_t prn

  *Satellite ID or PRN.*
- SatelliteSystem satelliteType

  *The satellite system that the observable originates.*
- CodeType codeType

  *The code type of the received signal.*
- FrequencyBand frequencyType

  *The frequency carrier of the received signal.*
- AssuranceLevel assurance

  *Assurance level for this observable.*
- double carrierToNoise

  *The carrier to noise ratio (C_no) of the received signal.*
- bool pseudorangeValid

  *Flag to indiate the validity of the observable pseudorange.*
- double pseudorange

  *The pseudorange measurement.*
- double pseudorangeVariance

  *The pseudorange measurement's variance.*
- bool dopplerValid

  *Flag to indicate the validity of the observable doppler.*
- double doppler

  *The doppler measurement.*
- double dopplerVariance

  *The variance of the doppler measurement.*
- bool carrierPhaseValid

  *Flag to indicate the validity of the observable carrier phase.*
- double carrierPhase

  *The carrier-phase measurement.*
- double carrierPhaseVariance

  *The variance of the carrier-phase measurement.*
- bool lossOfLock

  *Flag to indicate loss of carrier lock.*

### 8.24.1 Detailed Description

A structure for GNSS observables (pseudorange, carrier, doppler, etc)

Definition at line 415 of file IntegrityData.hpp.

### 8.24.2 Constructor & Destructor Documentation

#### 8.24.2.1 GNSSObservable()

```
pnt_integrity::data::GNSSObservable::GNSSObservable (
             const uint16_t & prnIn = 0,
             const SatelliteSystem & satTypeIn = SatelliteSystem::Other,
             const CodeType & codeTypeIn = CodeType::SigBLANK,
             const FrequencyBand & freqTypeIn = FrequencyBand::Band0,
             const AssuranceLevel & assuranceLevelIn = AssuranceLevel::Unavailable,
             const double & cnoIn = std::numeric_limits<double>::quiet_NaN(),
             const bool & psrValIn = false,
             const double & psrIn = std::numeric_limits<double>::quiet_NaN(),
             const double & psrVarIn = std::numeric_limits<double>::quiet_NaN(),
             const bool & doppValIn = false,
             const double & doppIn = std::numeric_limits<double>::quiet_NaN(),
             const double & doppVarIn = std::numeric_limits<double>::quiet_NaN(),
             const bool & cpValIn = false,
             const double & cpIn = std::numeric_limits<double>::quiet_NaN(),
             const double & cpVarIn = std::numeric_limits<double>::quiet_NaN(),
             const bool & lossOfLockIn = false )  [inline]
```

Default constructor to initialize values.

The constructor initializes all member variables to null states (i.e. NAN for double values, false for booleans, and unknown types for signal parameters

**Parameters**

| | |
|---|---|
| *prnIn* | Satellite ID or PRN |
| *satTypeIn* | The satellite system that the observable originates |
| *codeTypeIn* | The code type of the observable |
| *freqTypeIn* | The frequency carrier of the received signal |
| *assurance↵ LevelIn* | The assurance level for this observable |
| *cnoIn* | The carrier to noise ratio (C_no) of the received signal |
| *psrValIn* | Flag to indiate the validity of the pseudorange |
| *psrIn* | The pseudorange measurement |
| *psrVarIn* | The pseudorange measurement's variance |
| *doppValIn* | Flag to indicate the validity of the doppler |
| *doppIn* | The doppler measurement |
| *doppVarIn* | The variance of the doppler measurement |
| *cpValIn* | Flag to indicate the validity of the carrier phase |
| *cpIn* | The carrier-phase measurement |
| *cpVarIn* | The variance of the carrier-phase measurement |
| *lossOfLockIn* | Flag to indicate loss of carrier lock |

Definition at line 487 of file IntegrityData.hpp.

### 8.24.3 Member Function Documentation

#### 8.24.3.1 getUniqueID()

`uint64_t pnt_integrity::data::GNSSObservable::getUniqueID ( ) [inline]`

Returns a unique identifier for the observable.

Returns a unique identifier by multiplying the prn, satellite type, code type, and frequency type enumeration values (adjusted for zero-based entries). This function assumes that there are no enumeration values that have a value of -1.

**Returns**

A long integer representing the unique identifier

Definition at line 529 of file IntegrityData.hpp.

The documentation for this struct was generated from the following file:

- include/pnt_integrity/IntegrityData.hpp

## 8.25 pnt_integrity::data::GNSSObservables Struct Reference

The GNSSObservables message.

`#include <IntegrityData.hpp>`

### Public Member Functions

- GNSSObservables ()

  *Default constructor for the structure.*
- GNSSObservables (const Header &header, const GNSSTime &gnssTime, const GNSSObservableMap obsMap)

  *Constructor with provided data.*

### Public Attributes

- Header header

  *The message header.*
- GNSSTime gnssTime

  *The GNSSTime associated with the observable data.*
- GNSSObservableMap observables

  *A map of observables, keyed off of satellite id (or prn)*

### 8.25.1 Detailed Description

The GNSSObservables message.

This data structure represents the message format for a GNSS observable

Definition at line 546 of file IntegrityData.hpp.

### 8.25.2 Constructor & Destructor Documentation

#### 8.25.2.1 GNSSObservables()

```
pnt_integrity::data::GNSSObservables::GNSSObservables (
                const Header & header,
                const GNSSTime & gnssTime,
                const GNSSObservableMap obsMap )  [inline]
```

Constructor with provided data.

**Parameters**

| | |
|---|---|
| *header* | The provided header structure |
| *gnssTime* | A provided GNSSTime object |
| *obsMap* | The map of observables, keyed off of prn |

Definition at line 566 of file IntegrityData.hpp.

The documentation for this struct was generated from the following file:

- include/pnt_integrity/IntegrityData.hpp

## 8.26 pnt_integrity::data::GNSSSubframe Struct Reference

GNSS Subframe data.

`#include <IntegrityData.hpp>`

### Public Attributes

- Header header

  *The message header.*
- uint16_t prn

  *Satellite ID or PRN.*
- SatelliteSystem satelliteType

  *The satellite system that the observable originates.*
- std::vector< uint8_t > subframeData

  *Broadcast navigation data subframe bytes.*

### 8.26.1 Detailed Description

GNSS Subframe data.

This data structure represents a complete subframe of broadcast navigation data decoded from a single signal.

Definition at line 577 of file IntegrityData.hpp.

The documentation for this struct was generated from the following file:

- include/pnt_integrity/IntegrityData.hpp

## 8.27 pnt_integrity::data::GNSSTime Struct Reference

A GNSS time.

`#include <IntegrityData.hpp>`

### Public Member Functions

- GNSSTime (const int &week=0, const double &seconds=0.0, const TimeSystem &system=TimeSystem::GPS)

  *Default constructor for GNSSTime.*

### Public Attributes

- int weekNumber

  *The number of elapsed week since a pre-defined epoch (non-rollover)*
- double secondsOfWeek

  *Seconds into the week.*
- TimeSystem timeSystem

  *The reference time system.*

### 8.27.1 Detailed Description

A GNSS time.

Definition at line 93 of file IntegrityData.hpp.

### 8.27.2 Constructor & Destructor Documentation

#### 8.27.2.1 GNSSTime()

```
pnt_integrity::data::GNSSTime::GNSSTime (
             const int & week = 0,
             const double & seconds = 0.0,
             const TimeSystem & system = TimeSystem::GPS )  [inline]
```

Default constructor for GNSSTime.

**Parameters**

| | |
|---|---|
| *week* | The number of elapsed week since a pre-defined epoch (non-rollover) |
| *seconds* | Seconds into the week |
| *system* | The base timesystem used for the GNSS time |

Definition at line 107 of file IntegrityData.hpp.

The documentation for this struct was generated from the following file:

- include/pnt_integrity/IntegrityData.hpp

## 8.28 pnt_integrity::GpsAlmanac Class Reference

Class to parse and store almanac data for a GPS Satellite.

`#include <GPSAlmanac.hpp>`

### Public Member Functions

- GpsAlmanac (unsigned int &prn, double &tow, SVAlmHealth &svHealth, double &eccentricity, double &toa, double &deltaI, double &omegaDot, double &sqrtA, double &omega0, double &omega, double &m0, double &af0, double &af1, uint16_t &wna)

  *Constructs a GpsAlmanac object from parsed values.*

- GpsAlmanac (const unsigned int prn, const uint8_t(&subframe)[30])

  *Constructs a GpsAlmanac object from a compressed subframe Constructs a GpsAlmanac object by parsing the contents of the subframe into its engineering unit values. The subframe contains the 10, 30-bit words of a GPS subframe minus the 6 bits of parity resulting in 10, 24-bit words for a total of 30 bytes.*
- ∼GpsAlmanac ()

  *Destructor for the GpsAlmanac object.*
- void setAlmanac (const unsigned int &prn, const double &tow, const SVAlmHealth &svHealth, const double &eccentricity, const double &toa, const double &deltaI, const double &omegaDot, const double &sqrtA, const double &omega0, const double &omega, const double &m0, const double &af0, const double &af1, bool checkForValidity=true)

  *Set the Almanac fields using engineering parameter values.*
- void setAlmanac (const AlmanacParameters &param, bool checkForValidity=true)

  *Set the parsed almanac values for the almanac object.*
- bool getAlmanac (unsigned int &prn, double &tow, SVAlmHealth &svHealth, double &eccentricity, double &toa, double &deltaI, double &omegaDot, double &sqrtA, double &omega0, double &omega, double &m0, double &af0, double &af1) const

  *Get the Almanac fields in engineering units.*
- AlmanacParameters getAlmanac () const

  *Get almanac parameters.*
- void **getSvState** (const double &receiveTime, double &positionEcefX, double &positionEcefY, double &positionEcefZ, double &velocityEcefX, double &velocityEcefY, double &velocityEcefZ, double &svClockCorrection, const double &pseudorange=0.0) const
- unsigned int getPrn () const

  *Get the satellite PRN ID.*
- void setReferenceWeek (const uint16_t week, bool checkForValidity=true)

  *Set the full reference GPS week.*
- uint16_t getReferenceWeek () const

  *Get the full GPS reference week.*
- AlmanacSubframeFaults **getSubframeFaults** () const
- bool setSubframe (const unsigned int prn, const uint8_t(&subframe)[30], bool checkForValidity=true)

  *Parse the given subframe into the Almanac object.*
- const uint8_t $*$ getSubframe () const

  *Get a pointer to the compressed subframe data.*
- void getSubframe (uint8_t(&subframe)[30])

  *Get a copy of the subframe data subframe array to copy subframe data into.*
- double **getTow** ()
- std::string subframeToString () const

  *Returns a hex string containing the raw subframe value.*
- bool isSubframeValid () const

  *Indicates if the Alamanac subframe is valid.*
- bool isSvHealthy () const

  *Checks to see if almanac data for this SV is listed as healthy.*
- NavDataTimeOfArrival checkSubframeTOA (const uint8_t(&subframe)[30])

  *Checks to see if the supplied subframe is newer than the existing subframe by comparing TOA (Almanac Reference Time)*

## Static Public Member Functions

- static void setThresholds (const std::pair$<$ AlmanacParameters, AlmanacParameters $>$ &thresholds)

  *Set thresholds used to define validity of parsed subframe data.*
- static std::pair$<$ AlmanacParameters, AlmanacParameters $>$ getThresholds ()

  *Return stored threshold values.*

### 8.28.1 Detailed Description

Class to parse and store almanac data for a GPS Satellite.

The GpsAlmanac class stores almanac data from a GPS navigation data subframe in both parsed and compressed forms. The class is capable of both parsing compressed almanac subframes into its engineering unit components and generating compressed subframes from the individual almanac values.

Definition at line 112 of file GPSAlmanac.hpp.

### 8.28.2 Constructor & Destructor Documentation

#### 8.28.2.1 GpsAlmanac() [1/2]

```
pnt_integrity::GpsAlmanac::GpsAlmanac (
             unsigned int & prn,
             double & tow,
             SVAlmHealth & svHealth,
             double & eccentricity,
             double & toa,
             double & deltaI,
             double & omegaDot,
             double & sqrtA,
             double & omega0,
             double & omega,
             double & m0,
             double & af0,
             double & af1,
             uint16_t & wna )
```

Constructs a GpsAlmanac object from parsed values.

**Parameters**

| | |
|---|---|
| *prn* | Satellite PRN ID |
| *tow* | Time of week from subframe HOW |
| *eccentricity* | Eccentricity [dimensionless] |
| *toa* | Time of almanac [seconds] |
| *deltaI* | delta inclination angle [rad] |
| *omegaDot* | Rate of right ascension [rad/sec] |
| *sqrtA* | Square root of the semi-major axis [sqrt(meters)] |
| *omega0* | Right ascension angle [rad] |
| *omega* | Argument of perigee [rad] |
| *m0* | Mean anomaly at reference tim [rad] |
| *af0* | Clock aging term 1 [seconds] |
| *af1* | Clock aging term 2 [seconds/second] |
| *wna* | Almanac reference week (Full week number) |

#### 8.28.2.2 GpsAlmanac() [2/2]

```
pnt_integrity::GpsAlmanac::GpsAlmanac (
             const unsigned int prn,
             const uint8_t(&) subframe[30] )
```

Constructs a GpsAlmanac object from a compressed subframe Constructs a GpsAlmanac object by parsing the contents of the subframe into its engineering unit values. The subframe contains the 10, 30-bit words of a GPS subframe minus the 6 bits of parity resulting in 10, 24-bit words for a total of 30 bytes.

**Parameters**

| | |
|---|---|
| *prn* | PRN number of the Almanac subframe |
| *subframe* | Byte array containing the 240 bit subframe with no parity |

### 8.28.3 Member Function Documentation

#### 8.28.3.1 getAlmanac() [1/2]

```
bool pnt_integrity::GpsAlmanac::getAlmanac (
             unsigned int & prn,
             double & tow,
             SVAlmHealth & svHealth,
             double & eccentricity,
             double & toa,
             double & deltaI,
             double & omegaDot,
             double & sqrtA,
             double & omega0,
             double & omega,
             double & m0,
             double & af0,
             double & af1 ) const
```

Get the Almanac fields in engineering units.

**Parameters**

| | |
|---|---|
| *prn,Satellite* | PRN ID |
| *tow,Time* | of week from subframe HOW [sec] |
| *eccentricity,Eccentricity* | [dimensionless] |
| *toa,Time* | of almanac [seconds] |

**Parameters**

| | |
|---|---|
| *deltaI,delta* | inclination angle [rad] |
| *omegaDot,Rate* | of right ascension [rad/sec] |
| *sqrtA,Square* | root of the semi-major axis [sqrt(meters)] |
| *omega0,Right* | ascension angle [rad] |
| *omega,Argument* | of perigee [rad] |
| *m0,Mean* | anomaly at reference tim [rad] |
| *af0,Clock* | aging term 1 [seconds] |
| *af1,Clock* | aging term 2 [seconds/second] |

#### 8.28.3.2 getAlmanac() [2/2]

```
AlmanacParameters pnt_integrity::GpsAlmanac::getAlmanac ( ) const
```

Get almanac parameters.

**Returns**

Structure of stored almanac parameters

#### 8.28.3.3 getSubframe()

```
const uint8_t* pnt_integrity::GpsAlmanac::getSubframe ( ) const  [inline]
```

Get a pointer to the compressed subframe data.

**Returns**

a constant pointer to a 30 byte array containing the almanac data

Definition at line 255 of file GPSAlmanac.hpp.

#### 8.28.3.4 getThresholds()

```
static std::pair<AlmanacParameters, AlmanacParameters> pnt_integrity::GpsAlmanac::getThresholds (
)  [static]
```

Return stored threshold values.

Note static member function cannot be const

#### 8.28.3.5 isSubframeValid()

```
bool pnt_integrity::GpsAlmanac::isSubframeValid ( ) const  [inline]
```

Indicates if the Alamanac subframe is valid.

**Returns**

true if the almanac data has ben set and is valid

Definition at line 268 of file GPSAlmanac.hpp.

#### 8.28.3.6 setAlmanac() [1/2]

```
void pnt_integrity::GpsAlmanac::setAlmanac (
             const unsigned int & prn,
             const double & tow,
             const SVAlmHealth & svHealth,
             const double & eccentricity,
             const double & toa,
             const double & deltaI,
             const double & omegaDot,
             const double & sqrtA,
             const double & omega0,
             const double & omega,
             const double & m0,
             const double & af0,
             const double & af1,
             bool checkForValidity = true )
```

Set the Almanac fields using engineering parameter values.

**Parameters**

| | |
|---|---|
| *prn* | Satellite PRN ID |
| *tow* | Time of week from subframe HOW |
| *eccentricity* | Eccentricity [dimensionless] |
| *toa* | Time of almanac [seconds] |
| *deltaI* | Rate of inclination angle??? [semi-circles] |
| *omegaDot* | Rate of right ascension [semi-circles/sec] |
| *sqrtA* | Square root of the semi-major axis [sqrt(meters)] |
| *omega0* | Right ascension angle [semi-circles] |
| *omega* | Argument of perigee [semi-circles] |
| *m0* | Mean anomaly at reference tim [semi-circles] |
| *af0* | Clock aging term 1 [seconds] |
| *af1* | Clock aging term 2 [seconds/second] |

#### 8.28.3.7 setAlmanac() [2/2]

```
void pnt_integrity::GpsAlmanac::setAlmanac (
             const AlmanacParameters & param,
             bool checkForValidity = true )
```

Set the parsed almanac values for the almanac object.

Internally this method calls void setAlmanac(...).

**Parameters**

| | |
|---|---|
| *EphemerisParameters* | structure containing parsed data |

#### 8.28.3.8 setSubframe()

```
bool pnt_integrity::GpsAlmanac::setSubframe (
             const unsigned int prn,
             const uint8_t(&) subframe[30],
             bool checkForValidity = true )
```

Parse the given subframe into the Almanac object.

**Parameters**

| | |
|---|---|
| *subframe* | Byte array containing the 240 bit subframe with no parity |

#### 8.28.3.9 setThresholds()

```
static void pnt_integrity::GpsAlmanac::setThresholds (
             const std::pair< AlmanacParameters, AlmanacParameters > & thresholds )  [static]
```

Set thresholds used to define validity of parsed subframe data.

**Parameters**

| | |
|---|---|
| *thresholds* | Pair of AlmanacParameters structs defining minimum (first) and maximum (second) thresholds for each field. |

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

- include/pnt_integrity/GPSAlmanac.hpp

## 8.29 pnt_integrity::GpsEphemeris Class Reference

### Public Member Functions

- **GpsEphemeris** (const uint16_t &prn, const AlertFlag &alertFlag, const AntiSpoofFlag &asFlag, const uint32_t &towSf1, const uint16_t &weekNumber, const L2CodeType &codeOnL2, const uint16_t &uraIndex, const SVHealth &svHealth, const uint16_t &iodc, const L2NavDataFlag &l2PDataFlag, const double &groupDelay, const double &clockCorrectionTime, const double &clockAging3, const double &clockAging2, const double &clockAging1, const uint32_t &towSf2, const uint16_t &iodeSf2, const double &sinOrbitRadius, const double &meanMotionDifference, const double &meanAnomaly, const double &cosLatitude, const double &eccentricity, const double &sinLatitude, const double &sqrtSemiMajorAxis, const double &timeOfEphemeris, const FitInterval &fitInterval, const uint16_t &ageOfDataOffset, const uint32_t &towSf3, const double &cosInclination, const double &rightAscension, const double &sinInclination, const double &inclinationAngle, const double &cosOrbitRadius, const double &argumentOfPerigee, const double &ascensionRate, const uint16_t &iodeSf3, const double &inclinationRate, bool checkForValidity=true)
- **GpsEphemeris** (uint16_t prn, const uint32_t(&subframe1)[10], const uint32_t(&subframe2)[10], const uint32_t(&subframe3)[10], bool checkForValidity=true)
- **GpsEphemeris** (uint16_t prn, const uint8_t(&subframe1)[30], const uint8_t(&subframe2)[30], const uint8_t(&subframe3)[30], bool checkForValidity=true)
- bool getSvState (const double &receiveTime, double &positionEcefX, double &positionEcefY, double &positionEcefZ, double &velocityEcefX, double &velocityEcefY, double &velocityEcefZ, double &svClockCorrection, const double &pseudorange=0.0) const

  *Compute satellite ECEF position and velocity.*
- bool setEphemeris (const uint16_t &prn, const AlertFlag &alertFlag, const AntiSpoofFlag &asFlag, const uint32_t &towSf1, const uint16_t &weekNumber, const L2CodeType &codeOnL2, const uint16_t &uraIndex, const SVHealth &svHealth, const uint16_t &iodc, const L2NavDataFlag &l2PDataFlag, const double &groupDelay, const double &clockCorrectionTime, const double &clockAging3, const double &clockAging2, const double &clockAging1, const uint32_t &towSf2, const uint16_t &iodeSf2, const double &sinOrbitRadius, const double &meanMotionDifference, const double &meanAnomaly, const double &cosLatitude, const double &eccentricity, const double &sinLatitude, const double &sqrtSemiMajorAxis, const double &timeOfEphemeris, const FitInterval &fitInterval, const uint16_t &ageOfDataOffset, const uint32_t &towSf3, const double &cosInclination, const double &rightAscension, const double &sinInclination, const double &inclinationAngle, const double &cosOrbitRadius, const double &argumentOfPerigee, const double &ascensionRate, const uint16_t &iodeSf3, const double &inclinationRate, bool checkForValidity=true)

  *Set the parsed ephemeris values for the ephemeris object.*
- bool setEphemeris (const EphemerisParameters &params, bool checkForValidity=true)

  *Set the parsed ephemeris values for the ephemeris object.*
- bool getEphemeris (uint16_t &prn, AlertFlag &alertFlag, AntiSpoofFlag &asFlag, uint32_t &towSf1, uint16_t &weekNumber, L2CodeType &codeOnL2, uint16_t &uraIndex, SVHealth &svHealth, uint16_t &iodc, L2NavDataFlag &l2PDataFlag, double &groupDelay, double &clockCorrectionTime, double &clockAging3, double &clockAging2, double &clockAging1, uint32_t &towSf2, uint16_t &iodeSf2, double &sinOrbitRadius, double &meanMotionDifference, double &meanAnomaly, double &cosLatitude, double &eccentricity, double &sinLatitude, double &sqrtSemiMajorAxis, double &timeOfEphemeris, FitInterval &fitInterval, uint16_t &ageOfDataOffset, uint32_t &towSf3, double &cosInclination, double &rightAscension, double &sinInclination, double &inclinationAngle, double &cosOrbitRadius, double &argumentOfPerigee, double &ascensionRate, uint16_t &iodeSf3, double &inclinationRate) const

*Get stored ephemeris parameters.*
- EphemerisParameters getEphemeris () const

  *Get strored ephemeris parameters.*
- uint16_t getPrn () const

  *Get the satellite PRN ID.*
- double getTimeOfEphemeris () const

  *Get the time of ephemeris.*
- uint32_t **getTowSf1** () const
- uint32_t **getTowSf2** () const
- uint32_t **getTowSf3** () const
- uint16_t getWeekNumber () const

  *Get the transmission time week number.*
- bool isSvHealthy () const

  *Indicates if the satellite signal and navigation data are healthy.*
- Subframe1Fault getSubframe1Faults () const

  *Get fault status of subframe data.*
- Subframe2Fault **getSubframe2Faults** () const
- Subframe3Fault **getSubframe3Faults** () const
- bool setSubframe (const uint16_t &prn, const uint8_t(&subframe)[30], bool checkForValidity=true)

  *Adds an individual subframe to the ephemeris object.*
- bool setSubframes (const uint16_t prn, const uint8_t(&subframe1)[30], const uint8_t(&subframe2)[30], const uint8_t(&subframe3)[30], bool checkForValidity=true)

  *Adds all subframes to the ephemeris object.*
- int checkSubframeIssueDate (const uint8_t(&subframe)[30])
- void getSubframe1 (uint8_t(&subframe)[30])

  *Get a copy of subframe 1 data subframe array to copy subframe 1 data into.*
- void getSubframe2 (uint8_t(&subframe)[30])

  *Get a copy of subframe 2 data subframe array to copy subframe 2 data into.*
- void getSubframe3 (uint8_t(&subframe)[30])

  *Get a copy of subframe 3 data subframe array to copy subframe 3 data into.*
- const uint8_t $*$ getSubframe1 () const

  *Get a pointer to the compressed subframe 1 data.*
- const uint8_t $*$ getSubframe2 () const

  *Get a pointer to the compressed subframe 2 data.*
- const uint8_t $*$ getSubframe3 () const

  *Get a pointer to the compressed subframe 3 data.*
- bool isSubframe1Valid ()

  *Indicates if the subframe 1 data is valid.*
- bool isSubframe2Valid ()

  *Indicates if the subframe 2 data is valid.*
- bool isSubframe3Valid ()

  *Indicates if the subframe 3 data is valid.*
- bool **areAllSubramesValid** ()
- bool isEphemerisValid ()

  *Indicates if the ephemeris data is valid This checks that all subframes have been set and pass validity checks and that the issue date matches on each subframe.*
- void **clear** ()
- bool checkSubframesForFaults ()

*Check that ephemeris parameters are witihin defined thresholds.*

- std::string sf1FaultsToString () const

  *Converts the Ephemeris fault data into a readable string.*
- std::string **sf2FaultsToString** () const
- std::string **sf3FaultsToString** () const
- std::string toString () const

  *Converts the Ephemeris data into a readable string.*
- std::string toHexString () const

  *Converts the raw ephemeris subframes into a readable hex string.*
- std::string **sf1ToHexString** () const
- std::string **sf2ToHexString** () const
- std::string **sf3ToHexString** () const

## Static Public Member Functions

- static void setBounds (const std::pair< EphemerisParameters, EphemerisParameters > &bounds)

  *Set bounds used to define validity of parsed subframe data.*
- static std::pair< EphemerisParameters, EphemerisParameters > getBounds ()

  *Return stored threshold values.*

### 8.29.1 Detailed Description

Definition at line 212 of file GPSEphemeris.hpp.

### 8.29.2 Member Function Documentation

#### 8.29.2.1 checkSubframeIssueDate()

```
int pnt_integrity::GpsEphemeris::checkSubframeIssueDate (
             const uint8_t(&) subframe[30] )
```

Check if the issue date on the given subframes matches the issue date on any other subframes that have been already set. 0 - subframe matches issue date of current ephemeris subframes 1 - subframe is older than current ephemeris subframes 2 - subframe is newer than current ephemeris subframes

#### 8.29.2.2 checkSubframesForFaults()

```
bool pnt_integrity::GpsEphemeris::checkSubframesForFaults ( )
```

Check that ephemeris parameters are witihin defined thresholds.

Checks subframe parameters to determine if their values exceed minimum and maximum thresholds defined in the following member variable: std::pair<EphemerisParameters, EphemerisParameters> thresholds_; Note fields alertFlag, asFlag, codeOnL2, and fitInterval are not checked.

**Returns**

True if data is valid, false if one or more parameters exceed threshold

#### 8.29.2.3 getBounds()

```
static std::pair<EphemerisParameters, EphemerisParameters> pnt_integrity::GpsEphemeris::getBounds
( )  [static]
```

Return stored threshold values.

Note static member function cannot be const

#### 8.29.2.4 getEphemeris() [1/2]

```
bool pnt_integrity::GpsEphemeris::getEphemeris (
             uint16_t & prn,
             AlertFlag & alertFlag,
             AntiSpoofFlag & asFlag,
             uint32_t & towSf1,
             uint16_t & weekNumber,
             L2CodeType & codeOnL2,
             uint16_t & uraIndex,
             SVHealth & svHealth,
             uint16_t & iodc,
             L2NavDataFlag & l2PDataFlag,
             double & groupDelay,
             double & clockCorrectionTime,
             double & clockAging3,
             double & clockAging2,
             double & clockAging1,
             uint32_t & towSf2,
             uint16_t & iodeSf2,
             double & sinOrbitRadius,
             double & meanMotionDifference,
             double & meanAnomaly,
             double & cosLatitude,
             double & eccentricity,
             double & sinLatitude,
             double & sqrtSemiMajorAxis,
             double & timeOfEphemeris,
             FitInterval & fitInterval,
             uint16_t & ageOfDataOffset,
             uint32_t & towSf3,
             double & cosInclination,
             double & rightAscension,
             double & sinInclination,
             double & inclinationAngle,
             double & cosOrbitRadius,
             double & argumentOfPerigee,
             double & ascensionRate,
             uint16_t & iodeSf3,
             double & inclinationRate ) const
```

Get stored ephemeris parameters.

Input arguments are populated with the appropriate parameters.

**Returns**

True if all subframes are valid, false if one or more are not

#### 8.29.2.5 getEphemeris() [2/2]

```
EphemerisParameters pnt_integrity::GpsEphemeris::getEphemeris ( ) const
```

Get strored ephemeris parameters.

**Returns**

Structure of stored ephemeris parameters

#### 8.29.2.6 getSubframe1()

```
const uint8_t* pnt_integrity::GpsEphemeris::getSubframe1 ( ) const [inline]
```

Get a pointer to the compressed subframe 1 data.

**Returns**

a constant pointer to a 30 byte array containing the subframe data

Definition at line 553 of file GPSEphemeris.hpp.

#### 8.29.2.7 getSubframe1Faults()

`Subframe1Fault pnt_integrity::GpsEphemeris::getSubframe1Faults ( ) const [inline]`

Get fault status of subframe data.

Faults are detected by thresholding parsed subframe data. If data exceeds a minimum or maximum threshold, a fault is indicated.

**Returns**

Fault status structure, where 1 indicates a detected fault

Definition at line 487 of file GPSEphemeris.hpp.

#### 8.29.2.8 getSubframe2()

`const uint8_t* pnt_integrity::GpsEphemeris::getSubframe2 ( ) const [inline]`

Get a pointer to the compressed subframe 2 data.

**Returns**

a constant pointer to a 30 byte array containing the subframe data

Definition at line 558 of file GPSEphemeris.hpp.

#### 8.29.2.9 getSubframe3()

`const uint8_t* pnt_integrity::GpsEphemeris::getSubframe3 ( ) const [inline]`

Get a pointer to the compressed subframe 3 data.

**Returns**

a constant pointer to a 30 byte array containing the subframe data

Definition at line 563 of file GPSEphemeris.hpp.

#### 8.29.2.10 getSvState()

```
bool pnt_integrity::GpsEphemeris::getSvState (
             const double & receiveTime,
             double & positionEcefX,
             double & positionEcefY,
             double & positionEcefZ,
             double & velocityEcefX,
             double & velocityEcefY,
             double & velocityEcefZ,
             double & svClockCorrection,
             const double & pseudorange = 0.0 ) const
```

Compute satellite ECEF position and velocity.

**Parameters**

| | |
|---|---|
| *receiveTime* | measurement time associated with the pseudorange measurement (s into GPS week). |
| *positionEcefX* | ECEF X position (m) |
| *positionEcefY* | ECEF Y position (m) |
| *positionEcefY* | ECEF Z position (m) |
| *velocityEcefX* | ECEF X speed (m/s) |
| *velocityEcefY* | ECEF Y speed (m/s) |
| *velocityEcefZ* | ECEF Z speed (m/s) |
| *svClockCorrection* | satellite clock correction including polynomial fit, group delay, and relativistic effect (s) |
| *pseudorange* | (optional) user to satellite range (m). Inputting a pseudorange improves SV position and velocity accuracy. |

#### 8.29.2.11 getTimeOfEphemeris()

`double pnt_integrity::GpsEphemeris::getTimeOfEphemeris ( ) const [inline]`

Get the time of ephemeris.

**Returns**

the time of ephemeris in seconds into the week

Definition at line 464 of file GPSEphemeris.hpp.

#### 8.29.2.12 getWeekNumber()

`uint16_t pnt_integrity::GpsEphemeris::getWeekNumber ( ) const [inline]`

Get the transmission time week number.

**Returns**

the GPS week number modulo 1024

Definition at line 472 of file GPSEphemeris.hpp.

#### 8.29.2.13 isEphemerisValid()

`bool pnt_integrity::GpsEphemeris::isEphemerisValid ( )  [inline]`

Indicates if the ephemeris data is valid This checks that all subframes have been set and pass validity checks and that the issue date matches on each subframe.

**Returns**

True if

Definition at line 586 of file GPSEphemeris.hpp.

#### 8.29.2.14 isSubframe1Valid()

`bool pnt_integrity::GpsEphemeris::isSubframe1Valid ( )  [inline]`

Indicates if the subframe 1 data is valid.

**Returns**

True if subframe 1 has been set and passes validity checks

Definition at line 567 of file GPSEphemeris.hpp.

#### 8.29.2.15 isSubframe2Valid()

`bool pnt_integrity::GpsEphemeris::isSubframe2Valid ( )  [inline]`

Indicates if the subframe 2 data is valid.

**Returns**

True if subframe 2 has been set and passes validity checks

Definition at line 571 of file GPSEphemeris.hpp.

#### 8.29.2.16 isSubframe3Valid()

```
bool pnt_integrity::GpsEphemeris::isSubframe3Valid ( )  [inline]
```

Indicates if the subframe 3 data is valid.

**Returns**

True if subframe 3 has been set and passes validity checks

Definition at line 575 of file GPSEphemeris.hpp.

#### 8.29.2.17 setBounds()

```
static void pnt_integrity::GpsEphemeris::setBounds (
            const std::pair< EphemerisParameters, EphemerisParameters > & bounds )  [static]
```

Set bounds used to define validity of parsed subframe data.

**Parameters**

| | |
|---|---|
| *bounds* | Pair of EphemerisParameters structs defining minimum (first) and maximum (second) thresholds for each field. |

### 8.29.2.18 setEphemeris() [1/2]

```
bool pnt_integrity::GpsEphemeris::setEphemeris (
            const uint16_t & prn,
            const AlertFlag & alertFlag,
            const AntiSpoofFlag & asFlag,
            const uint32_t & towSf1,
            const uint16_t & weekNumber,
            const L2CodeType & codeOnL2,
            const uint16_t & uraIndex,
            const SVHealth & svHealth,
            const uint16_t & iodc,
            const L2NavDataFlag & l2PDataFlag,
            const double & groupDelay,
            const double & clockCorrectionTime,
            const double & clockAging3,
            const double & clockAging2,
            const double & clockAging1,
            const uint32_t & towSf2,
            const uint16_t & iodeSf2,
            const double & sinOrbitRadius,
            const double & meanMotionDifference,
            const double & meanAnomaly,
            const double & cosLatitude,
            const double & eccentricity,
            const double & sinLatitude,
            const double & sqrtSemiMajorAxis,
            const double & timeOfEphemeris,
            const FitInterval & fitInterval,
            const uint16_t & ageOfDataOffset,
            const uint32_t & towSf3,
            const double & cosInclination,
            const double & rightAscension,
            const double & sinInclination,
            const double & inclinationAngle,
            const double & cosOrbitRadius,
            const double & argumentOfPerigee,
            const double & ascensionRate,
            const uint16_t & iodeSf3,
            const double & inclinationRate,
            bool checkForValidity = true )
```

Set the parsed ephemeris values for the ephemeris object.

**Parameters**

| | |
|---|---|
| *prn* | PRN number [1-32] |
| *groupDelay* | T_GD - Estimated group delay differential [sec] |
| *clockCorrectionTime* | t_0c - |
| *clockAging3* | a_f2 - Sv clock drift rate [sec/sec^2] |
| *clockAging2* | a_f1 - Sv clock drift [sec/sec] |
| *clockAging1* | a_f0 - Sv clock bias [sec] |
| *cosOrbitRadius* | C_rc - Amplitude of the cosine harmonic correction term to the orbit radius [m] |
| *sinOrbitRadius* | C_rs - Amplitude of the sine harmonic correction term to the orbit radius [m] |
| *cosLatitude* | C_uc - Amplitude of the cosine harmonic correction term to the argument of latitude [rad] |
| *sinLatitude* | C_us - Amplitude of the sin harmonic correction term to the argument of latitude [rad] |
| *cosInclination* | C_ic - Amplitude of the cosine harmonic correction term to the angle of inclination [rad] |
| *sinInclination* | C_is - Amplitude of the sine harmonic correction term to the angle of inclination [rad] |
| *meanMotionDifference* | deltaN - Mean motion difference from the computed value [rad/sec] |
| *meanAnomaly* | M_0 - Mean anomaly at reference time [rad] |
| *eccentricity* | e - Eccentricity [dimensionless] |
| *sqrtSemiMajorAxis* | sqrtA - Square root of the semi-major axis [m^1/2] |
| *rightAscension* | Omega_0 - Longitude of ascending node of orbit plane at weekly epoch [rad] |
| *inclinationAngle* | i_0 - Inclination angle at reference time [semicircle] |
| *argumentOfPerigee* | omega - Argument of perigee [rad] |
| *ascensionRate* | Omega_dot - Rate of right ascension [rad/s] |
| *inclinationRate* | I_dot - Rate of inclination angle [rad/s] |
| *timeOfEphemeris* | t_0e - Ephmeris reference time [sec] |

**Returns**

true if the ephemeris data contains no faults or faults were not checked

### 8.29.2.19 setEphemeris() [2/2]

```
bool pnt_integrity::GpsEphemeris::setEphemeris (
             const EphemerisParameters & params,
             bool checkForValidity = true )
```

Set the parsed ephemeris values for the ephemeris object.

Internally this method calls void setEphemeris(...).

**Parameters**

| | |
|---|---|
| *EphemerisParameters* | structure containing parsed data |

#### 8.29.2.20 setSubframe()

```
bool pnt_integrity::GpsEphemeris::setSubframe (
             const uint16_t & prn,
             const uint8_t(&) subframe[30],
             bool checkForValidity = true )
```

Adds an individual subframe to the ephemeris object.

Add and parse a single subframe. The subframe ID is extracted and the appropriate member variables are set.

**Parameters**

| | |
|---|---|
| *subframe* | Subframe data formated into 30 8 bit words with no parity |

**Returns**

True if the subframe is 1,2, or 3 and successfully parsed.

#### 8.29.2.21 setSubframes()

```
bool pnt_integrity::GpsEphemeris::setSubframes (
             const uint16_t prn,
             const uint8_t(&) subframe1[30],
             const uint8_t(&) subframe2[30],
             const uint8_t(&) subframe3[30],
             bool checkForValidity = true )
```

Adds all subframes to the ephemeris object.

Add and parse subframes. The subframe ID is extracted and the appropriate member variables are set.

**Parameters**

| | |
|---|---|
| *subframe1* | Subframe 1 formated into 30 8 bit words with no parity |
| *subframe2* | Subframe 2 formated into 30 8 bit words with no parity |
| *subframe3* | Subframe 3 formated into 30 8 bit words with no parity |

**Returns**

True if all subframes are successfully parsed and valid.

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

- include/pnt_integrity/GPSEphemeris.hpp

## 8.30 pnt_integrity::data::Header Struct Reference

The header used for all associated data types.

`#include <IntegrityData.hpp>`

### Public Member Functions

- Header (const long &seq=0, const Timestamp &ts_arrival=Timestamp(), const Timestamp &ts_valid=Timestamp(), const std::string &dev_id="")
  *Default constructor for a header.*

### Public Attributes

- long seq_num
  *The sequence number of the header.*
- Timestamp timestampArrival
  *The arrival time of the header at the data transport layer.*
- Timestamp timestampValid
  *The valid time of the header / measurement data.*
- std::string deviceId
  *Unique identifier for the measurement system / sensor / source.*

### 8.30.1 Detailed Description

The header used for all associated data types.

Definition at line 115 of file IntegrityData.hpp.

### 8.30.2 Constructor & Destructor Documentation

#### 8.30.2.1 Header()

```
pnt_integrity::data::Header::Header (
            const long & seq = 0,
            const Timestamp & ts_arrival = Timestamp(),
            const Timestamp & ts_valid = Timestamp(),
            const std::string & dev_id = "" )  [inline]
```

Default constructor for a header.

**Parameters**

| | |
|---|---|
| *seq* | The sequence number of the header |
| *ts_arrival* | The arrival time of the header at the data transport layer |
| *ts_valid* | The valid time of the header / data |
| *dev_id* | Unique identifier for the measurement source |

Definition at line 133 of file IntegrityData.hpp.

The documentation for this struct was generated from the following file:

- include/pnt_integrity/IntegrityData.hpp

## 8.31 pnt_integrity::data::IMU Struct Reference

A structure that represents IMU measurement data.

`#include <IntegrityData.hpp>`

### Public Attributes

- Header header

    *The message header.*
- double delta_v [3]
- double delta_theta [3]

### 8.31.1 Detailed Description

A structure that represents IMU measurement data.

Definition at line 776 of file IntegrityData.hpp.

### 8.31.2 Member Data Documentation

#### 8.31.2.1 delta_theta

`double pnt_integrity::data::IMU::delta_theta[3]`

Angular rate integrated over period delta_t, providing an "average change in angle" measurement. units: rad

Definition at line 787 of file IntegrityData.hpp.

#### 8.31.2.2 delta_v

`double pnt_integrity::data::IMU::delta_v[3]`

Acceleration integrated over period delta_t, providing an "average change in velocity" measurement. units: m/s

Definition at line 783 of file IntegrityData.hpp.

The documentation for this struct was generated from the following file:

- include/pnt_integrity/IntegrityData.hpp

## 8.32 pnt_integrity::IntegrityDataRepository Class Reference

Class definition for the history of data at a single PNT node.

`#include <IntegrityDataRepository.hpp>`

### Public Member Functions

- IntegrityDataRepository (IntegrityDataRepository const &)=delete

  *Delete the copy constructor.*
- void operator= (IntegrityDataRepository constIntegrityDataRepository)=delete

  *Delete the assignment operator.*
- template<class T >
  void addEntry (const double &timeOfWeek, const T &data)

  *Adds a local data entry to the repo.*
- template<class T >
  void addEntry (const double &timeOfWeek, const std::string &nodeId, const T &data)

  *Adds a remote data entry to the repo.*
- template<class T >
  bool getData (const double &timeOfWeek, T &data)

  *Returns the local data entry at the specified time.*
- template<class T >
  bool getNewestData (T &data, double &time)

  *Returns the newest available local data entry of type T.*
- template<class T >
  bool getData (const double &timeOfWeek, const std::string &nodeId, T &data)

  *Returns the remote data entry at the specified time.*
- template<class T >
  bool getNewestData (const std::string &nodeId, T &data, double &time)

  *Returns the newest available remote data entry of type T.*
- void addEntry (const double &timeOfWeek, const uint32_t &satelliteID, const data::GNSSObservable &gnssObs)

  *Adds a local GNSSObservable value entry.*
- bool getData (const double &timeOfWeek, const uint32_t &satelliteID, data::GNSSObservable &gnssObs)

  *Retrieves a local GNSS observable from the time history.*

- void addEntry (const double &timeOfWeek, const std::string &nodeID, const uint32_t &satelliteID, const data::GNSSObservable &gnssObs)

  *Adds a remote GNSSObservable entry.*
- bool getData (const double &timeOfWeek, const std::string &nodeID, const uint32_t &satelliteID, data::GNSSObservable &gnssObs)

  *Retrieves a local GNSS observable from the time history.*
- void setHistoryPeriod (const double &period)

  *Sets the history period.*
- void setLogMessageHandler (const logutils::LogCallback &logMsgHandler)

  *Sets the log message handler to provided callback.*
- void manageHistory ()

  *Trims the stored data in the repository.*
- size_t getRepoSize ()

  *Returns the size of the repo (number of time entries)*
- bool getNewestEntry (TimeEntry &timeEntry)

  *Returns the newest time entry.*
- bool getEntry (const double &timeOfWeek, TimeEntry &timeEntry)

  *Returns the time entry for the specified time.*
- bool getNewestEntries (std::vector$<$ TimeEntry $>$ &timeEntryVec, double startTime)

  *Returns the newest time entries that start appear after a given time.*
- void clearEntries ()

  *Clear the repository contents.*

### Static Public Member Functions

- static IntegrityDataRepository & getInstance ()

  *Function to gain a singlton instance of the history.*
- static bool sortTimeEntry (TimeEntry &t0, TimeEntry &t1)

  *Comparator function for sorting TimeEntry objects by their time of week.*

## 8.32.1 Detailed Description

Class definition for the history of data at a single PNT node.

The IntegrityDataRepository object is a singleton, so therefore only 1 observable history lives in the application

Definition at line 94 of file IntegrityDataRepository.hpp.

## 8.32.2 Constructor & Destructor Documentation

#### 8.32.2.1 IntegrityDataRepository()

```
pnt_integrity::IntegrityDataRepository::IntegrityDataRepository (
             IntegrityDataRepository const &  )  [delete]
```

Delete the copy constructor.

Deleting the copy constructor to help ensure singleton

### 8.32.3 Member Function Documentation

#### 8.32.3.1 addEntry() [1/4]

```
template<class T >
void pnt_integrity::IntegrityDataRepository::addEntry (
             const double & timeOfWeek,
             const T & data )
```

Adds a local data entry to the repo.

Adds a local measurement to the entry

**Parameters**

| | |
|---|---|
| *timeOfWeek* | The time associated with the observable |
| *data* | The local data structure |

Definition at line 353 of file IntegrityDataRepository.hpp.

#### 8.32.3.2 addEntry() [2/4]

```
template<class T >
void pnt_integrity::IntegrityDataRepository::addEntry (
             const double & timeOfWeek,
             const std::string & nodeId,
             const T & data )
```

Adds a remote data entry to the repo.

Adds a local remote to the entry

**Parameters**

| | |
|---|---|
| *timeOfWeek* | The time associated with the observable |
| *nodeId* | The name or node ID of the remote |
| *data* | The remote data structure |

Definition at line 369 of file IntegrityDataRepository.hpp.

### 8.32.3.3 addEntry() [3/4]

```
void pnt_integrity::IntegrityDataRepository::addEntry (
             const double & timeOfWeek,
             const uint32_t & satelliteID,
             const data::GNSSObservable & gnssObs )
```

Adds a local GNSSObservable value entry.

**Parameters**

| | |
|---|---|
| *timeOfWeek* | The time associated with the observable |
| *satelliteID* | The ID number for the GNSS observable's origin |
| *gnssObs* | The GNSS observable to add to the repository |

### 8.32.3.4 addEntry() [4/4]

```
void pnt_integrity::IntegrityDataRepository::addEntry (
             const double & timeOfWeek,
             const std::string & nodeID,
             const uint32_t & satelliteID,
             const data::GNSSObservable & gnssObs )
```

Adds a remote GNSSObservable entry.

**Parameters**

| | |
|---|---|
| *timeOfWeek* | The time associated with the observable |
| *nodeID* | The identifier of the remote node |
| *satelliteID* | The ID number for the GNSS observable's origin |
| *gnssObs* | The GNSS observable. |

#### 8.32.3.5 getData() [1/4]

```
template<class T >
bool pnt_integrity::IntegrityDataRepository::getData (
            const double & timeOfWeek,
            T & data )
```

Returns the local data entry at the specified time.

**Parameters**

| | |
|---|---|
| *timeOfWeek* | The time of the desired data |
| *data* | The requested local data entry |

Definition at line 405 of file IntegrityDataRepository.hpp.

#### 8.32.3.6 getData() [2/4]

```
template<class T >
bool pnt_integrity::IntegrityDataRepository::getData (
            const double & timeOfWeek,
            const std::string & nodeId,
            T & data )
```

Returns the remote data entry at the specified time.

**Parameters**

| | |
|---|---|
| *timeOfWeek* | The time of the desired data |
| *nodeId* | The identifier string for the desired node |
| *data* | The requested remote data entry |

Definition at line 462 of file IntegrityDataRepository.hpp.

#### 8.32.3.7 getData() [3/4]

```
bool pnt_integrity::IntegrityDataRepository::getData (
            const double & timeOfWeek,
            const uint32_t & satelliteID,
            data::GNSSObservable & gnssObs )
```

Retrieves a local GNSS observable from the time history.

**Parameters**

| | |
|---|---|
| *timeOfWeek* | The time associated with the observable |
| *satelliteID* | The ID number for the GNSS observable's origin |
| *gnssObs* | The GNSS observable. |

**Returns**

True if the observable exists

#### 8.32.3.8 getData() [4/4]

```
bool pnt_integrity::IntegrityDataRepository::getData (
             const double & timeOfWeek,
             const std::string & nodeID,
             const uint32_t & satelliteID,
             data::GNSSObservable & gnssObs )
```

Retrieves a local GNSS observable from the time history.

**Parameters**

| | |
|---|---|
| *timeOfWeek* | The time associated with the observable |
| *nodeID* | The identifier of the remote node |
| *satelliteID* | The ID number for the GNSS observable's origin |
| *gnssObs* | The GNSS observable. |

**Returns**

True if the remote node and observable exist

#### 8.32.3.9 getEntry()

```
bool pnt_integrity::IntegrityDataRepository::getEntry (
             const double & timeOfWeek,
             TimeEntry & timeEntry )
```

Returns the time entry for the specified time.

Just a public wrapper for findEntry

**Parameters**

| | |
|---|---|
| *timeOfWeek* | The time to get the entry for |
| *timeEntry* | The time entry returned by reference |

**Returns**

True if the repository is not empty

### 8.32.3.10 getInstance()

```
static IntegrityDataRepository& pnt_integrity::IntegrityDataRepository::getInstance ( )  [inline],
[static]
```

Function to gain a singlton instance of the history.

**Returns**

The unique instance of the history object

Definition at line 104 of file IntegrityDataRepository.hpp.

### 8.32.3.11 getNewestData() [1/2]

```
template<class T >
bool pnt_integrity::IntegrityDataRepository::getNewestData (
             T & data,
             double & time )
```

Returns the newest available local data entry of type T.

**Parameters**

| | |
|---|---|
| *data* | The requested local data entry |
| *time* | The time of the found data |

Definition at line 430 of file IntegrityDataRepository.hpp.

#### 8.32.3.12 getNewestData() `[2/2]`

```
template<class T >
bool pnt_integrity::IntegrityDataRepository::getNewestData (
            const std::string & nodeId,
            T & data,
            double & time )
```

Returns the newest available remote data entry of type T.

**Parameters**

| | |
|---|---|
| *node↩ Id* | The identifier string for the desired node |
| *data* | The requested remote data entry |
| *time* | The time of the found data |

Definition at line 504 of file IntegrityDataRepository.hpp.

#### 8.32.3.13 getNewestEntries()

```
bool pnt_integrity::IntegrityDataRepository::getNewestEntries (
            std::vector< TimeEntry > & timeEntryVec,
            double startTime )
```

Returns the newest time entries that start appear after a given time.

This function will return the time entries that are within the time range of now and the given start time. It will return as many as it can before running out of entries.

**Parameters**

| | |
|---|---|
| *timeEntryVec* | The vector of the newest time entries |
| *startTime* | The earliest time entry to return |

**Returns**

True if the repository is not empty

#### 8.32.3.14 getNewestEntry()

```
bool pnt_integrity::IntegrityDataRepository::getNewestEntry (
            TimeEntry & timeEntry )
```

Returns the newest time entry.

This function will return the newest time entry in the repo

**Parameters**

| | |
|---|---|
| *timeEntry* | The newest time entry returned by reference |

**Returns**

True if the repository is not empty

#### 8.32.3.15 getRepoSize()

```
size_t pnt_integrity::IntegrityDataRepository::getRepoSize ( )  [inline]
```

Returns the size of the repo (number of time entries)

Returns the number of time entries into the repsoitory

**Returns**

The number of time entries

Definition at line 263 of file IntegrityDataRepository.hpp.

#### 8.32.3.16 manageHistory()

```
void pnt_integrity::IntegrityDataRepository::manageHistory ( )
```

Trims the stored data in the repository.

This function will trim the repository to the length determined by setHistoryPeriod. The default history is 10 if not set

#### 8.32.3.17 operator=()

```
void pnt_integrity::IntegrityDataRepository::operator= (
             IntegrityDataRepository constIntegrityDataRepository )  [delete]
```

Delete the assignment operator.

Deleting the assignment operator to help insure singleton

#### 8.32.3.18 setHistoryPeriod()

```
void pnt_integrity::IntegrityDataRepository::setHistoryPeriod (
             const double & period )  [inline]
```

Sets the history period.

Defines the time history length that resides in the data history. Defaults to 10 if this function is not called

**Parameters**

| | |
|---|---|
| *period* | The time (in seconds) that will be kept in the history |

Definition at line 234 of file IntegrityDataRepository.hpp.

#### 8.32.3.19 setLogMessageHandler()

```
void pnt_integrity::IntegrityDataRepository::setLogMessageHandler (
             const logutils::LogCallback & logMsgHandler )  [inline]
```

Sets the log message handler to provided callback.

**Parameters**

| | |
|---|---|
| *logMsgHandler* | The provided call back function |

Definition at line 243 of file IntegrityDataRepository.hpp.

#### 8.32.3.20 sortTimeEntry()

```
static bool pnt_integrity::IntegrityDataRepository::sortTimeEntry (
             TimeEntry & t0,
             TimeEntry & t1 )  [inline], [static]
```

Comparator function for sorting TimeEntry objects by their time of week.

Can be used with std::sort on vectors of TimeEntry objects.

**Parameters**

| | |
|---|---|
| *t0* | The first TimeEntry to compare |
| *t1* | The second TimeEntry to compare |

**Returns**

true if t0.timeOfWeek_ $<$ t1.timeOfWeek_, false otherwise

Definition at line 306 of file IntegrityDataRepository.hpp.

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

- include/pnt_integrity/IntegrityDataRepository.hpp

## 8.33 pnt_integrity::IntegrityMonitor Class Reference

Class implementation of integrity monitoring using AssuranceChecks and IntegrityData.

`#include <IntegrityMonitor.hpp>`

### Public Member Functions

- IntegrityMonitor (const logutils::LogCallback &log=logutils::printLogToStdOut)

  *Default constructor.*
- IntegrityDataRepository & getRepo ()

  *Returns an instance to the repository.*
- bool registerCheck (const std::string &checkName, AssuranceCheck ∗checkPtr)

  *Function to register user-defined check.*
- void setMultiPrnAssuranceData (MultiPrnAssuranceMap al)

  *Return function for the multi-prn assurance data.*
- MultiPrnAssuranceMap getMultiPrnAssuranceData ()

  *Return function for the multi-prn assurance data.*
- data::AssuranceLevel getAssuranceLevel ()

  *Returns overall assurance level.*
- double getAssuranceValue ()

  *Returns overall assurance value.*
- data::AssuranceReports getAssuranceReports ()

  *Returns assurance reports from all registered checks.*
- void determineAssuranceLevels ()

  *Calculates overall assurance levels accross all registered checks.*
- void handleGnssObservables (const data::GNSSObservables &gnssObs, const bool &localFlag=true)

  *Handler function for GNSSObservables.*
- void handleGnssSubframe (const data::GNSSSubframe &gnssObs, const bool &localFlag=true)

  *Handler function for GNSSSubframe.*
- void handlePositionVelocity (const data::PositionVelocity &posVel, const bool &localFlag=true)

  *Handler function for PositionVelocity messages.*
- void handleEstimatedPositionVelocity (const data::PositionVelocity &posVel, const bool &localFlag=true)

  *Handler function for Estimated PositionVelocity messages.*
- void handleDistanceTraveled (const data::AccumulatedDistranceTraveled &dist)

  *Handler function for AccumulatedDistranceTraveled messages.*
- void handleMeasuredRange (const data::MeasuredRange &range, const bool &localFlag=true)

  *Handler function for MeasuredRange messages.*
- template< typename samp_type >
  void handleIfSampleData (const double &time, const if_data_utils::IFSampleData< samp_type > &ifData)

  *Handler function for IFSampleData messages.*
- void handleClockOffset (const data::ClockOffset &clockOffset, const bool &localFlag)

  *Handler function for ClockOffset messages.*
- void handleAGC (const data::AgcValue &agcValue)

  *Handler function AGC setting.*

- template<class T >
  double getCorrectedEntryTime (const double &time, const T &data, const bool &local=true, const std::string &deviceId=std::string())

  *Template function that determines the correct timestamp.*
- template<class T >
  void addDataToRepo (const double &time, const T &data, const bool &local=true, const std::string &deviceId=std::string())

  *Template function that adds received data to the repository.*
- void setLogMessageHandler (const logutils::LogCallback &logMsgHandler)

  *Sets the log message handler to provided callback.*
- size_t getNumUsedChecks ()

  *Returns the number of assurance checks currently used in the monitor.*
- bool isCheckUsed (const std::string &checkName)

  *Returns a flag to indicate if check was used in current level calculation.*
- void reset ()

  *Reset the integrity monitor.*
- void **setLastKnownGoodPosition** (const data::PositionVelocity &posVel)
- void **clearLastKnownGoodPosition** ()

## 8.33.1 Detailed Description

Class implementation of integrity monitoring using AssuranceChecks and IntegrityData.

Definition at line 60 of file IntegrityMonitor.hpp.

## 8.33.2 Constructor & Destructor Documentation

### 8.33.2.1 IntegrityMonitor()

```
pnt_integrity::IntegrityMonitor::IntegrityMonitor (
            const logutils::LogCallback & log = logutils::printLogToStdOut )
```

Default constructor.

The constructor set's the repository's log handling to use the logging function provided by the integrity monitor

**Parameters**

| | |
|---|---|
| *log* | A log callback function for log messages |

### 8.33.3 Member Function Documentation

#### 8.33.3.1 addDataToRepo()

```
template<class T >
void pnt_integrity::IntegrityMonitor::addDataToRepo (
             const double & time,
             const T & data,
             const bool & local = true,
             const std::string & deviceId = std::string() )
```

Template function that adds received data to the repository.

**Parameters**

| | |
|---|---|
| *time* | The timestamp used for time entries into the repo |
| *data* | The received data message /structure to be entered |
| *local* | A flag to indicate if the data is local or remote data |
| *device↩ Id* | A string to identify remote data entries |

Definition at line 418 of file IntegrityMonitor.hpp.

#### 8.33.3.2 getCorrectedEntryTime()

```
template<class T >
double pnt_integrity::IntegrityMonitor::getCorrectedEntryTime (
             const double & time,
             const T & data,
             const bool & local = true,
             const std::string & deviceId = std::string() )
```

Template function that determines the correct timestamp.

**Parameters**

| | |
|---|---|
| *time* | The timestamp used for time entries into the repo |
| *data* | The received data message /structure to be entered |
| *local* | A flag to indicate if the data is local or remote data |
| *device↩ Id* | A string to identify remote data entries |

Definition at line 325 of file IntegrityMonitor.hpp.

#### 8.33.3.3 getNumUsedChecks()

`size_t pnt_integrity::IntegrityMonitor::getNumUsedChecks ( ) [inline]`

Returns the number of assurance checks currently used in the monitor.

**Returns**

The number of assurance checks

Definition at line 250 of file IntegrityMonitor.hpp.

#### 8.33.3.4 getRepo()

`IntegrityDataRepository& pnt_integrity::IntegrityMonitor::getRepo ( ) [inline]`

Returns an instance to the repository.

**Returns**

A singleton instance of the repository

Definition at line 75 of file IntegrityMonitor.hpp.

#### 8.33.3.5 handleAGC()

```
void pnt_integrity::IntegrityMonitor::handleAGC (
             const data::AgcValue & agcValue )
```

Handler function AGC setting.

Call this function on receipt of an AGC setting

**Parameters**

| | |
|---|---|
| *agcValue* | The current AGC setting from a a receiver |

### 8.33.3.6 handleClockOffset()

```
void pnt_integrity::IntegrityMonitor::handleClockOffset (
             const data::ClockOffset & clockOffset,
             const bool & localFlag )
```

Handler function for ClockOffset messages.

Call this function on receipt of a ClockOffset message. The function will call the handleClockOffset on all registered checks

**Parameters**

| | |
|---|---|
| *clockOffset* | The clock bias and drift with Header for timestamp |
| *localFlag* | Flag to indicate local or remote data |

### 8.33.3.7 handleDistanceTraveled()

```
void pnt_integrity::IntegrityMonitor::handleDistanceTraveled (
             const data::AccumulatedDistranceTraveled & dist )
```

Handler function for AccumulatedDistranceTraveled messages.

**Parameters**

| | |
|---|---|
| *dist* | The provided distance traveled message |

### 8.33.3.8 handleEstimatedPositionVelocity()

```
void pnt_integrity::IntegrityMonitor::handleEstimatedPositionVelocity (
             const data::PositionVelocity & posVel,
             const bool & localFlag = true )
```

Handler function for Estimated PositionVelocity messages.

Call this function on receipt of a PositionVelocity message that contains an external estimate of the current position and velocity. The function will call the handleEstimatedPositionVelocity in all registered checks

**Parameters**

| | |
|---|---|
| *localFlag* | A flag to indicate if the source of the data is from a local or remote source (defaults to "True") |
| *posVel* | The provided data message |

#### 8.33.3.9 handleGnssObservables()

```
void pnt_integrity::IntegrityMonitor::handleGnssObservables (
             const data::GNSSObservables & gnssObs,
             const bool & localFlag = true )
```

Handler function for GNSSObservables.

Call this function on receipt of a GNSSObservables message. The function will call the handleGnssObservables in all registered checks

**Parameters**

| | |
|---|---|
| *localFlag* | A flag to indicate if the source of the observable data is from a local or remote source (defaults to "True") |
| *gnssObs* | The provided data message |

#### 8.33.3.10 handleGnssSubframe()

```
void pnt_integrity::IntegrityMonitor::handleGnssSubframe (
             const data::GNSSSubframe & gnssObs,
             const bool & localFlag = true )
```

Handler function for GNSSSubframe.

Call this function on receipt of a GNSSSubframe message. The function will call the handleGnssSubframe in all registered checks

**Parameters**

| | |
|---|---|
| *localFlag* | A flag to indicate if the source of the observable data is from a local or remote source (defaults to "True") |
| *gnssObs* | The provided data message |

#### 8.33.3.11 handleIfSampleData()

```
template<typename samp_type >
void pnt_integrity::IntegrityMonitor::handleIfSampleData (
            const double & time,
            const if_data_utils::IFSampleData< samp_type > & ifData )
```

Handler function for IFSampleData messages.

Call this function on receipt of an IFSampleData message. The function will call the handleIfSampleData on all registered checks

**Parameters**

| | |
|---|---|
| *time* | The timestamp of the IF Data |
| *ifData* | The incoming IF sample data |

Definition at line 439 of file IntegrityMonitor.hpp.

#### 8.33.3.12 handleMeasuredRange()

```
void pnt_integrity::IntegrityMonitor::handleMeasuredRange (
            const data::MeasuredRange & range,
            const bool & localFlag = true )
```

Handler function for MeasuredRange messages.

Call this function on receipt of a MeasuredRange message. The function will call the handleMeasuredRange in all registered checks

**Parameters**

| | |
|---|---|
| *localFlag* | A flag to indicate if the source of the data is from a local or remote source (defaults to "True") |
| *range* | The provided measured range message |

#### 8.33.3.13 handlePositionVelocity()

```
void pnt_integrity::IntegrityMonitor::handlePositionVelocity (
            const data::PositionVelocity & posVel,
            const bool & localFlag = true )
```

Handler function for PositionVelocity messages.

Call this function on receipt of a PositionVelocity message. The function will call the handlePositionVelocity in all registered checks

**Parameters**

| | |
|---|---|
| *localFlag* | A flag to indicate if the source of the data is from a local or remote source (defaults to "True") |
| *posVel* | The provided data message |

#### 8.33.3.14 registerCheck()

```
bool pnt_integrity::IntegrityMonitor::registerCheck (
            const std::string & checkName,
            AssuranceCheck * checkPtr )
```

Function to register user-defined check.

Register's an assurance check with the monitor. The process simply adds a provided pointer to the check to an internally held vector of check pointers

**Parameters**

| | |
|---|---|
| *checkName* | The name of the check object |
| *checkPtr* | A pointer to an AssuranceCheck |

**Returns**

True if successful

#### 8.33.3.15 setLogMessageHandler()

```
void pnt_integrity::IntegrityMonitor::setLogMessageHandler (
            const logutils::LogCallback & logMsgHandler )  [inline]
```

Sets the log message handler to provided callback.

**Parameters**

| | |
|---|---|
| *logMsgHandler* | The provided call back function |

Definition at line 238 of file IntegrityMonitor.hpp.

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

- include/pnt_integrity/IntegrityMonitor.hpp

## 8.34 pnt_integrity::data::MeasuredRange Struct Reference

A structure that represents a distance measurement to a known point.

`#include <IntegrityData.hpp>`

### Public Member Functions

- MeasuredRange (const bool &valid=false)

  *Default constructor.*

### Public Attributes

- Header header

  *The message header.*
- bool rangeValid

  *Flag to indicate validity of range measurement.*
- double range

  *The range measurement to the feature.*
- double variance

  *The variance associated with the range measurement.*
- GeodeticPosition3d featurePosition

  *The feature location.*
- double feature_position_covariance_ [3][3]

  *The covariance of the geodetic position.*

### 8.34.1 Detailed Description

A structure that represents a distance measurement to a known point.

This structure holds all relative data that represents a measured distance to a feature with a known location

Definition at line 795 of file IntegrityData.hpp.

### 8.34.2 Constructor & Destructor Documentation

#### 8.34.2.1 MeasuredRange()

```
pnt_integrity::data::MeasuredRange::MeasuredRange (
            const bool & valid = false )  [inline]
```

Default constructor.

**Parameters**

| | |
|---|---|
| *valid* | Flag to indicate measurement validity |

Definition at line 814 of file IntegrityData.hpp.

The documentation for this struct was generated from the following file:

- include/pnt_integrity/IntegrityData.hpp

## 8.35 pnt_integrity::NavDataCheckDiagnostics Struct Reference

Structure for check diagnostics.

`#include <NavigationDataCheck.hpp>`

### Public Attributes

- bool **dataValid**
- std::string **dataValidMsg**
- bool **towValid**
- std::string **towValidMsg**
- bool **wnValid**
- std::string **wnValidMsg**

### 8.35.1 Detailed Description

Structure for check diagnostics.

Definition at line 69 of file NavigationDataCheck.hpp.

The documentation for this struct was generated from the following file:

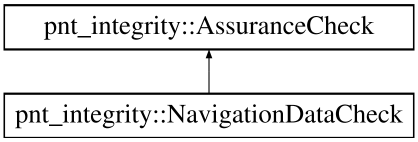
- include/pnt_integrity/NavigationDataCheck.hpp

## 8.36 pnt_integrity::NavigationDataCheck Class Reference

Class implementation for the navigation data check.

`#include <NavigationDataCheck.hpp>`

Inheritance diagram for pnt_integrity::NavigationDataCheck:


pnt_integrity::AssuranceCheck
pnt_integrity::NavigationDataCheck

## Public Member Functions

- NavigationDataCheck (const std::string &name="navigation_data_check", const logutils::LogCallback &log=logutils↩ ::printLogToStdOut)

  *Constructor for the check class.*
- void **stopThreads** ()
- virtual bool handleGnssSubframe (const data::GNSSSubframe &gnssSubframe)

  *Handler function for GNSS Subframes.*
- virtual bool runCheck ()

  *Triggers a manual check calculation.*
- virtual void calculateAssuranceLevel (const double &)

  *Function to explicitly set the assurance level of the check.*
- void setPublishDiagnostics (std::function< void(const double &, const NavDataCheckDiagnostics &)> handler)

  *Connects the internal publishing function to external interface.*

## Additional Inherited Members

### 8.36.1 Detailed Description

Class implementation for the navigation data check.

Definition at line 87 of file NavigationDataCheck.hpp.

### 8.36.2 Constructor & Destructor Documentation

#### 8.36.2.1 NavigationDataCheck()

```
pnt_integrity::NavigationDataCheck::NavigationDataCheck (
             const std::string & name = "navigation_data_check",
             const logutils::LogCallback & log = logutils::printLogToStdOut )  [inline]
```

Constructor for the check class.

**Parameters**

| | |
|---|---|
| *name* | The name of the check |
| *bounds* | The minimum amount of position jump that will trigger the check (meters). |
| *log* | A provided log callback function to use for log mesages |

Definition at line 98 of file NavigationDataCheck.hpp.

## 8.36.3 Member Function Documentation

### 8.36.3.1 calculateAssuranceLevel()

```
virtual void pnt_integrity::NavigationDataCheck::calculateAssuranceLevel (
             const double & ) [inline], [virtual]
```

Function to explicitly set the assurance level of the check.

Uses whatever data is available to calculate the current assurance level

Implements pnt_integrity::AssuranceCheck.

Definition at line 145 of file NavigationDataCheck.hpp.

### 8.36.3.2 handleGnssSubframe()

```
virtual bool pnt_integrity::NavigationDataCheck::handleGnssSubframe (
             const data::GNSSSubframe & gnssSubframe ) [virtual]
```

Handler function for GNSS Subframes.

Function to handle provided GNSS Broadcast Nav. Data.

**Returns**

True if successful

Reimplemented from pnt_integrity::AssuranceCheck.

### 8.36.3.3 runCheck()

```
virtual bool pnt_integrity::NavigationDataCheck::runCheck ( ) [virtual]
```

Triggers a manual check calculation.

Use this function to run a manual check calculation that is not triggered off the receipt of a message (pure virtual)

**Returns**

True if successful

Implements pnt_integrity::AssuranceCheck.

### 8.36.3.4 setPublishDiagnostics()

```
void pnt_integrity::NavigationDataCheck::setPublishDiagnostics (
             std::function< void(const double &, const NavDataCheckDiagnostics &)> handler )
[inline]
```

Connects the internal publishing function to external interface.

This function connects the internal "publishDiagnostics" function to an external, custom function of choice

**Parameters**

| | |
|---|---|
| *handler* | Provided handler function |

Definition at line 153 of file NavigationDataCheck.hpp.

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

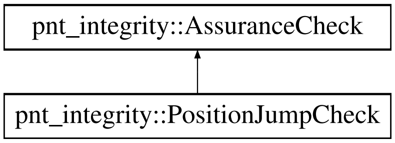
- include/pnt_integrity/NavigationDataCheck.hpp

## 8.37 pnt_integrity::PositionJumpCheck Class Reference

Class implementation for the position-jump check.

`#include <PositionJumpCheck.hpp>`

Inheritance diagram for pnt_integrity::PositionJumpCheck:


pnt_integrity::AssuranceCheck
pnt_integrity::PositionJumpCheck

### Public Member Functions

- PositionJumpCheck (const std::string &name="position_jump_check", const double &minimumBound=15.0, const bool &useEstimatedPv=false, const bool &useDistTraveled=false, const double &maximumVelocity=5.0, const double &posStdDevMultiplier=6.0, const logutils::LogCallback &log=logutils::printLogToStdOut)

  *Constructor for the check class.*
- virtual bool handlePositionVelocity (const data::PositionVelocity &posVel, const bool &localFlag)

  *Handler function for Position / Velocity message.*
- virtual bool handleEstimatedPositionVelocity (const data::PositionVelocity &pv)

  *Handler function for an estimated Position / Velocity message.*
- virtual bool handleDistanceTraveled (const data::AccumulatedDistranceTraveled &dist)

  *Handler function for AccumulatedDistranceTraveled messages.*
- void setLastGoodPosition (const double &updateTime, const data::GeodeticPosition3d &position)

  *Sets the last known good position.*
- virtual bool runCheck ()

  *Triggers a manual check calculation.*
- virtual void calculateAssuranceLevel (const double &)

  *Function to explicitly set the assurance level of the check.*
- double getCalculatedDistance ()

  *Returns the calculated distance between the current position to the last good position.*
- double getDistanceTraveled ()

  *Returns the currently estimated distance traveled since the last known good position.*
- double getBound ()

  *Returns the current bound that is used by the check.*
- void setPublishDiagnostics (std::function< void(const double &, const PosJumpCheckDiagnostics &)> handler)

  *Connects the internal publishing function to external interface.*
- void **clearCurrentEstimatedPosition** ()

## Additional Inherited Members

### 8.37.1 Detailed Description

Class implementation for the position-jump check.

Definition at line 68 of file PositionJumpCheck.hpp.

### 8.37.2 Constructor & Destructor Documentation

#### 8.37.2.1 PositionJumpCheck()

```
pnt_integrity::PositionJumpCheck::PositionJumpCheck (
             const std::string & name = "position_jump_check",
             const double & minimumBound = 15.0,
             const bool & useEstimatedPv = false,
             const bool & useDistTraveled = false,
             const double & maximumVelocity = 5.0,
             const double & posStdDevMultiplier = 6.0,
             const logutils::LogCallback & log = logutils::printLogToStdOut )  [inline]
```

Constructor for the check class.

**Parameters**

| | |
|---|---|
| *name* | The string name of the check |
| *minimumBound* | The minimum amount of position jump that will trigger the check (meters). |
| *useEstimatedPv* | Flag to tell check to use the incoming estimated position rather than distance traveled or max velocity propagation |
| *useDistTraveled* | Flag to indicate whether or not the check should use a provided distance traveled to compute the jump bound |
| *maximumVelocity* | The maximum velocity of the platform that will be used to calculate the bound if a distance traveled is not used (m/s) |
| *posStdDevMultiplier* | Scale factor on input position standard deviation |
| *log* | A provided log callback function to use for log mesages |

Definition at line 91 of file PositionJumpCheck.hpp.

### 8.37.3 Member Function Documentation

#### 8.37.3.1 calculateAssuranceLevel()

```
virtual void pnt_integrity::PositionJumpCheck::calculateAssuranceLevel (
            const double & ) [inline], [virtual]
```

Function to explicitly set the assurance level of the check.

Uses whatever data is available to calculate the current assurance level

Implements pnt_integrity::AssuranceCheck.

Definition at line 187 of file PositionJumpCheck.hpp.

#### 8.37.3.2 getBound()

```
double pnt_integrity::PositionJumpCheck::getBound ( ) [inline]
```

Returns the current bound that is used by the check.

**Returns**

The current jump bound

Definition at line 209 of file PositionJumpCheck.hpp.

#### 8.37.3.3 getCalculatedDistance()

```
double pnt_integrity::PositionJumpCheck::getCalculatedDistance ( ) [inline]
```

Returns the calculated distance between the current position to the last good position.

**Returns**

The calculated distance

Definition at line 192 of file PositionJumpCheck.hpp.

#### 8.37.3.4 getDistanceTraveled()

```
double pnt_integrity::PositionJumpCheck::getDistanceTraveled ( )  [inline]
```

Returns the currently estimated distance traveled since the last known good position.

**Returns**

The estimated distance traveled

Definition at line 201 of file PositionJumpCheck.hpp.

#### 8.37.3.5 handleDistanceTraveled()

```
virtual bool pnt_integrity::PositionJumpCheck::handleDistanceTraveled (
             const data::AccumulatedDistranceTraveled & dist )  [inline], [virtual]
```

Handler function for AccumulatedDistranceTraveled messages.

**Parameters**

| | |
|---|---|
| *dist* | The provided distance traveled message |

**Returns**

True if successful

Reimplemented from pnt_integrity::AssuranceCheck.

Definition at line 153 of file PositionJumpCheck.hpp.

### 8.37.3.6 handleEstimatedPositionVelocity()

```
virtual bool pnt_integrity::PositionJumpCheck::handleEstimatedPositionVelocity (
             const data::PositionVelocity & pv )  [virtual]
```

Handler function for an estimated Position / Velocity message.

Function to handle provided posivion / velocity messages (virtual)

**Parameters**

| | |
|---|---|
| *pv* | The provided estimated position velocity message / structure |

**Returns**

True if successful

Reimplemented from pnt_integrity::AssuranceCheck.

### 8.37.3.7 handlePositionVelocity()

```
virtual bool pnt_integrity::PositionJumpCheck::handlePositionVelocity (
             const data::PositionVelocity & posVel,
             const bool & localFlag )  [virtual]
```

Handler function for Position / Velocity message.

Function to handle provided posivion / velocity messages

**Parameters**

| | |
|---|---|
| *posVel* | The provided position velocity message / structure |
| *localFlag* | Indicates if this is a local or remote message |

**Returns**

True if successful

Reimplemented from pnt_integrity::AssuranceCheck.

### 8.37.3.8 runCheck()

```
virtual bool pnt_integrity::PositionJumpCheck::runCheck ( )  [virtual]
```

Triggers a manual check calculation.

Use this function to run a manual check calculation that is not triggered off the receipt of a message (pure virtual)

**Returns**

True if successful

Implements pnt_integrity::AssuranceCheck.

### 8.37.3.9 setLastGoodPosition()

```
void pnt_integrity::PositionJumpCheck::setLastGoodPosition (
             const double & updateTime,
             const data::GeodeticPosition3d & position )  [virtual]
```

Sets the last known good position.

Provides if the assurance check with knowledge of a last known good position for use in calculations (if needed by the specific implementation)

**Parameters**

| | |
|---|---|
| *updateTime* | The time associated with the last good position |
| *position* | The last known good position |

Reimplemented from pnt_integrity::AssuranceCheck.

#### 8.37.3.10 setPublishDiagnostics()

```
void pnt_integrity::PositionJumpCheck::setPublishDiagnostics (
             std::function< void(const double &, const PosJumpCheckDiagnostics &)> handler )
[inline]
```

Connects the internal publishing function to external interface.

This function connects the internal "publishDiagnostics" function to an external, custom function of choice

**Parameters**

| | |
|---|---|
| *handler* | Provided handler function |

Definition at line 221 of file PositionJumpCheck.hpp.

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

- include/pnt_integrity/PositionJumpCheck.hpp

## 8.38 pnt_integrity::data::PositionVelocity Struct Reference

A structure to represent a Position / Velocity message.

`#include <IntegrityData.hpp>`

### Public Member Functions

- PositionVelocity (const Header &headerIn=Header(), const GeodeticPosition3d &positionIn=GeodeticPosition3d())

  *Constructor for the PositionVelocity structure.*
- bool isPositionValid ()

  *Checks the validity of the position.*
- bool isPositionCovarianceValid ()

  *Checks the validity of the covariance.*
- bool isVelocityValid ()

  *Checks the validity of the veocity.*
- bool isVelocityCovarianceValid ()

  *Checks the validity of the covariance.*
- bool checkValidity ()

  *Checks the structure to make sure all data fields are valid.*

## Public Attributes

- Header header

  *The message header.*
- GeodeticPosition3d position

  *The 3D geodetic position.*
- double velocity [3]

  *The velocity in north-east-down (NED)*
- double covariance [6][6]

  *The cross-covariance for position / velocity in NED (6x6)*

### 8.38.1 Detailed Description

A structure to represent a Position / Velocity message.

This structure represents a data message that contains a geodetic 3d position, an NED velocity, and a 6x6 cross-covariance of position / velocity

Definition at line 655 of file IntegrityData.hpp.

### 8.38.2 Constructor & Destructor Documentation

#### 8.38.2.1 PositionVelocity()

```
pnt_integrity::data::PositionVelocity::PositionVelocity (
             const Header & headerIn = Header(),
             const GeodeticPosition3d & positionIn = GeodeticPosition3d() )  [inline]
```

Constructor for the PositionVelocity structure.

The default constructor for the position / velocity message can be optionally provided with a pre-built header and position structure. The velocity and covariance arrays are initialized to NaN and must be set to desired values after object construction

**Parameters**

| | |
|---|---|
| *headerIn* | A provided header object |
| *positionIn* | A provided 3D position (geodetic) |

Definition at line 675 of file IntegrityData.hpp.

The documentation for this struct was generated from the following file:

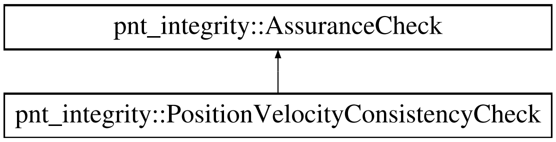
- include/pnt_integrity/IntegrityData.hpp

## 8.39 pnt_integrity::PositionVelocityConsistencyCheck Class Reference

Class implementation for the position velocity check.

`#include <PositionVelocityConsistencyCheck.hpp>`

Inheritance diagram for pnt_integrity::PositionVelocityConsistencyCheck:


pnt_integrity::AssuranceCheck
pnt_integrity::PositionVelocityConsistencyCheck

### Public Member Functions

- PositionVelocityConsistencyCheck (const std::string &name="Position Velocity Check", const double &sampleWindow=5.0, const double &errorThreshSF=2.0, const logutils::LogCallback &log=logutils::printLogToStdOut)

  *Default constructor for the check class.*
- bool handlePositionVelocity (const data::PositionVelocity &posVel, const bool &localFlag)

  *Handler function for Position / Velocity message.*
- void calculateAssuranceLevel (const double &)

  *Function to explicitly set the assurance level of the check.*
- bool runCheck ()

  *Triggers a manual check calculation.*
- void setPublishDiagnostics (std::function< void(const double &timestamp, const PosVelConsCheckDiagnostics &checkData)> handler)

  *Connects the internal publishing function to external interface.*

### Additional Inherited Members

### 8.39.1 Detailed Description

Class implementation for the position velocity check.

Definition at line 80 of file PositionVelocityConsistencyCheck.hpp.

### 8.39.2 Constructor & Destructor Documentation

#### 8.39.2.1 PositionVelocityConsistencyCheck()

```
pnt_integrity::PositionVelocityConsistencyCheck::PositionVelocityConsistencyCheck (
                const std::string & name = "Position Velocity Check",
                const double & sampleWindow = 5.0,
                const double & errorThreshSF = 2.0,
                const logutils::LogCallback & log = logutils::printLogToStdOut )  [inline]
```

Default constructor for the check class.

Constructor explicitly disables multi-prn support.

**Parameters**

| | |
|---|---|
| *name* | The name of the check object |
| *sampleWindow* | The duration of time (in seconds) over which to get position and velocity data for checking integrity |
| *errorThreshSF* | The scale factor to apply to the velocity variance that is used as an error threshold |
| *log* | A provided log callback function to use |

Definition at line 93 of file PositionVelocityConsistencyCheck.hpp.

## 8.39.3 Member Function Documentation

### 8.39.3.1 calculateAssuranceLevel()

```
void pnt_integrity::PositionVelocityConsistencyCheck::calculateAssuranceLevel (
             const double &  )  [inline], [virtual]
```

Function to explicitly set the assurance level of the check.

Uses whatever data is available to calculate the assurance level

Implements pnt_integrity::AssuranceCheck.

Definition at line 123 of file PositionVelocityConsistencyCheck.hpp.

### 8.39.3.2 handlePositionVelocity()

```
bool pnt_integrity::PositionVelocityConsistencyCheck::handlePositionVelocity (
             const data::PositionVelocity & posVel,
             const bool & localFlag )  [virtual]
```

Handler function for Position / Velocity message.

Function to handle provided posivion / velocity messages

**Parameters**

| | |
|---|---|
| *posVel* | The provided position velocity message / structure |
| *localFlag* | Indicates if this is a local or remote message |

**Returns**

True if successful

Reimplemented from pnt_integrity::AssuranceCheck.

#### 8.39.3.3 runCheck()

```
bool pnt_integrity::PositionVelocityConsistencyCheck::runCheck ( )  [virtual]
```

Triggers a manual check calculation.

Use this function to run a manual check calculation that is not triggered off the receipt of a message (pure virtual)

**Returns**

True if successful

Implements pnt_integrity::AssuranceCheck.

#### 8.39.3.4 setPublishDiagnostics()

```
void pnt_integrity::PositionVelocityConsistencyCheck::setPublishDiagnostics (
             std::function< void(const double &timestamp, const PosVelConsCheckDiagnostics &check↩
Data)> handler )  [inline]
```

Connects the internal publishing function to external interface.

This function connects the internal "publishDiagnostics" function to an external, custom function of choice

**Parameters**

| | |
|---|---|
| *handler* | Provided handler function |

Definition at line 139 of file PositionVelocityConsistencyCheck.hpp.

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

- include/pnt_integrity/PositionVelocityConsistencyCheck.hpp

## 8.40 pnt_integrity::PosJumpCheckDiagnostics Struct Reference

Structure for check diagnostics.

`#include <PositionJumpCheck.hpp>`

### Public Attributes

- double distance
- double bound

### 8.40.1 Detailed Description

Structure for check diagnostics.

Definition at line 58 of file PositionJumpCheck.hpp.

### 8.40.2 Member Data Documentation

#### 8.40.2.1 bound

`double pnt_integrity::PosJumpCheckDiagnostics::bound`

The bound on the distance, determined by maximum velocity, distance traveled, or the covariance on the estimated position

Definition at line 65 of file PositionJumpCheck.hpp.

#### 8.40.2.2 distance

`double pnt_integrity::PosJumpCheckDiagnostics::distance`

Depending on the mode, this is either the distance to the last known good position, or the distance to the current estimated position

Definition at line 62 of file PositionJumpCheck.hpp.

The documentation for this struct was generated from the following file:

- include/pnt_integrity/PositionJumpCheck.hpp

## 8.41 pnt_integrity::PosVelConsCheckDiagnostics Struct Reference

Structure used to publish diagnostic data.

`#include <PositionVelocityConsistencyCheck.hpp>`

### Public Attributes

- std::vector< double > errorVals

  *The error values over all examined pairs.*
- std::vector< double > errorThresh

  *The threshold for each error based on velocity variance.*
- double percentBad

  *Percentage of pairs that are above the threshold.*
- double inconsistentThresh

  *The inconsistent threshold used.*
- double unassuredThresh

  *The unassured threshold used.*

### 8.41.1 Detailed Description

Structure used to publish diagnostic data.

Definition at line 65 of file PositionVelocityConsistencyCheck.hpp.

The documentation for this struct was generated from the following file:

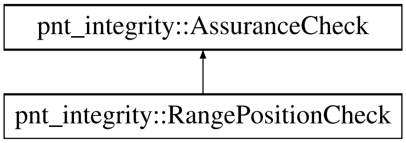
- include/pnt_integrity/PositionVelocityConsistencyCheck.hpp

## 8.42 pnt_integrity::RangePositionCheck Class Reference

Class implementation for the range / position check.

`#include <RangePositionCheck.hpp>`

Inheritance diagram for pnt_integrity::RangePositionCheck:


pnt_integrity::AssuranceCheck
pnt_integrity::RangePositionCheck

## Public Member Functions

- RangePositionCheck (const std::string &name="Range-Position check", const logutils::LogCallback &log=logutils::printLogToStdOut)

  *Default constructor for the check class.*
- bool runCheck ()

  *Triggers a manual check calculation.*
- virtual bool handlePositionVelocity (const data::PositionVelocity &posVel, const bool &local)

  *Handler function for Position / Velocity message.*
- virtual bool handleMeasuredRange (const data::MeasuredRange &range)

  *Handler function for measrured range.*
- void calculateAssuranceLevel (const double &time)

  *Function to explicitly set the assurance level of the check.*
- void setPublishDiagnostics (std::function< void(const double &, const RngPosCheckDiagnostics &)> handler)

  *Connects the internal publishing function to external interface.*

## Additional Inherited Members

### 8.42.1 Detailed Description

Class implementation for the range / position check.

Definition at line 85 of file RangePositionCheck.hpp.

### 8.42.2 Constructor & Destructor Documentation

#### 8.42.2.1 RangePositionCheck()

```
pnt_integrity::RangePositionCheck::RangePositionCheck (
            const std::string & name = "Range-Position check",
            const logutils::LogCallback & log = logutils::printLogToStdOut )  [inline]
```

Default constructor for the check class.

Constructor explicitly disables multi-prn support.

**Parameters**

| | |
|---|---|
| *name* | The name of the check object |
| *log* | A provided log callback function to use |

Definition at line 94 of file RangePositionCheck.hpp.

### 8.42.3 Member Function Documentation

#### 8.42.3.1 calculateAssuranceLevel()

```
void pnt_integrity::RangePositionCheck::calculateAssuranceLevel (
            const double & time )  [virtual]
```

Function to explicitly set the assurance level of the check.

For this check, this function cycles through all of the individual PRN assurance values, analyzes them, and then sets the master assurance level associted with the check.

Implements pnt_integrity::AssuranceCheck.

#### 8.42.3.2 handleMeasuredRange()

```
virtual bool pnt_integrity::RangePositionCheck::handleMeasuredRange (
            const data::MeasuredRange & range )  [virtual]
```

Handler function for measrured range.

Function to handle provided range measurements (pure virtual)

**Parameters**

| | |
|---|---|
| *range* | The provided range measurement message / structure |

**Returns**

True if successful

Reimplemented from pnt_integrity::AssuranceCheck.

#### 8.42.3.3 handlePositionVelocity()

```
virtual bool pnt_integrity::RangePositionCheck::handlePositionVelocity (
            const data::PositionVelocity & posVel,
            const bool & local )  [virtual]
```

Handler function for Position / Velocity message.

Function to handle provided posivion / velocity messages

**Parameters**

| | |
|---|---|
| *posVel* | The provided position velocity message / structure |
| *local* | Indicates if this is a local or remote message |

**Returns**

True if successful

Reimplemented from pnt_integrity::AssuranceCheck.

#### 8.42.3.4 runCheck()

```
bool pnt_integrity::RangePositionCheck::runCheck ( )  [virtual]
```

Triggers a manual check calculation.

Use this function to run a manual check calculation that is not triggered off the receipt of a message (pure virtual)

**Returns**

True if successful

Implements pnt_integrity::AssuranceCheck.

#### 8.42.3.5 setPublishDiagnostics()

```
void pnt_integrity::RangePositionCheck::setPublishDiagnostics (
            std::function< void(const double &, const RngPosCheckDiagnostics &)> handler )  [inline]
```

Connects the internal publishing function to external interface.

This function connects the internal "publishDiagnostics" function to an external, custom function of choice

**Parameters**

| | |
|---|---|
| *handler* | Provided handler function |

Definition at line 146 of file RangePositionCheck.hpp.

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

- include/pnt_integrity/RangePositionCheck.hpp

## 8.43 pnt_integrity::RepositoryEntry Class Reference

Class definition for an entry into the repository.

`#include <RepositoryEntry.hpp>`

### Public Member Functions

- RepositoryEntry (const DataLocaleType &type=DataLocaleType::Local, const std::string &nodeID="local", const logutils::LogCallback &log=logutils::printLogToStdOut)

  *Constructor for an entry into the repository.*
- void addEntry (const uint32_t &satelliteID, const data::GNSSObservable &gnssObs)

  *Adds a provided GNSS observable into the data entry.*
- bool getData (const uint32_t &satelliteID, data::GNSSObservable &gnssObs) const

  *Returns a GNSS Observable.*
- void addEntry (const data::GNSSObservableMap &gnssObsMap)

  *Adds a provided GNSS observable map into the data entry.*
- void addEntry (const data::GNSSObservables &gnssObservables)

  *Adds a provided GNSS observables as the data entry.*
- bool getData (data::GNSSObservables &gnssObservables) const

  *Returns a GNSSObservables.*
- void getData (data::GNSSObservableMap &gnssObsMap) const

  *Returns a GNSS ObservableMap.*
- void addEntry (const data::MeasuredRange &range)

  *Adds an a measured (RF) range to another location or node.*
- bool getData (data::MeasuredRange &range) const

  *Returns the RF range observable.*
- void addEntry (const data::PositionVelocity &posVel)

  *Adds position velocity measurement data to the entry.*
- bool getData (data::PositionVelocity &posVel) const

  *Returns the position velocity data from the repo entry.*
- void addEntry (const data::ClockOffset &clockOffset)

  *Adds clock offset data to the entry.*
- bool getData (data::ClockOffset &clockOffset) const

  *Returns the clock offset data from the repo entry.*
- void setLogMessageHandler (const logutils::LogCallback &logMsgHandler)

  *Sets the log message handler to provided callback.*

### 8.43.1 Detailed Description

Class definition for an entry into the repository.

A RepositoryEntry represents a collection of integrity data measurements from a single node at a unique time. Currently the object containes an RfRange and a GNSSObservableMap. More data structures will be added as new integrity checks are added to the framework.

Definition at line 72 of file RepositoryEntry.hpp.

## 8.43.2 Constructor & Destructor Documentation

### 8.43.2.1 RepositoryEntry()

```
pnt_integrity::RepositoryEntry::RepositoryEntry (
             const DataLocaleType & type = DataLocaleType::Local,
             const std::string & nodeID = "local",
             const logutils::LogCallback & log = logutils::printLogToStdOut )  [inline]
```

Constructor for an entry into the repository.

The constructor takes in a string that indicates what node the data / measurement / observable belongs to. Defaults to "local" to indicate that the observable was taken at this node's location.

Definition at line 80 of file RepositoryEntry.hpp.

## 8.43.3 Member Function Documentation

### 8.43.3.1 addEntry() [1/6]

```
void pnt_integrity::RepositoryEntry::addEntry (
             const uint32_t & satelliteID,
             const data::GNSSObservable & gnssObs )  [inline]
```

Adds a provided GNSS observable into the data entry.

This function will place the provided GNSS observable structure in the map with the corresponding satellite ID key.

**Note**

If data for the provided satelite ID already exists, it will be overwritten.

**Parameters**

| | |
|---|---|
| *satelliteID* | The satellite id number (or PRN) |
| *gnssObs* | The GNSS observable data structure |

Definition at line 99 of file RepositoryEntry.hpp.

#### 8.43.3.2 addEntry() [2/6]

```
void pnt_integrity::RepositoryEntry::addEntry (
            const data::GNSSObservableMap & gnssObsMap )  [inline]
```

Adds a provided GNSS observable map into the data entry.

This function will place the provided GNSS observable map into the entry. It overwrites the existing map entry with the provided one. Use the addEntry function for a single GNSSObservable to add to the existing map

**Parameters**

| | |
|---|---|
| *gnssObsMap* | The provided GNSSObs Map |

Definition at line 124 of file RepositoryEntry.hpp.

#### 8.43.3.3 addEntry() [3/6]

```
void pnt_integrity::RepositoryEntry::addEntry (
            const data::GNSSObservables & gnssObservables )  [inline]
```

Adds a provided GNSS observables as the data entry.

This function will place the provided GNSS observables as the entry. It overwrites the existing entry with the provided one.

**Parameters**

| | |
|---|---|
| *gnssObservables* | The provided GNSSObservables |

Definition at line 135 of file RepositoryEntry.hpp.

#### 8.43.3.4 addEntry() [4/6]

```
void pnt_integrity::RepositoryEntry::addEntry (
            const data::MeasuredRange & range )
```

Adds an a measured (RF) range to another location or node.

**Note**

> Any existing value will be overwritten

**Parameters**

| | |
|---|---|
| *range* | The measured range |

#### 8.43.3.5 addEntry() [5/6]

```
void pnt_integrity::RepositoryEntry::addEntry (
             const data::PositionVelocity & posVel )  [inline]
```

Adds position velocity measurement data to the entry.

**Parameters**

| | |
|---|---|
| *posVel* | The provided position / velocity structure |

Definition at line 191 of file RepositoryEntry.hpp.

#### 8.43.3.6 addEntry() [6/6]

```
void pnt_integrity::RepositoryEntry::addEntry (
             const data::ClockOffset & clockOffset )  [inline]
```

Adds clock offset data to the entry.

**Parameters**

| | |
|---|---|
| *clockOffset* | The provided clock offset structure |

Definition at line 214 of file RepositoryEntry.hpp.

#### 8.43.3.7 getData() [1/4]

```
bool pnt_integrity::RepositoryEntry::getData (
             const uint32_t & satelliteID,
             data::GNSSObservable & gnssObs ) const
```

Returns a GNSS Observable.

**Parameters**

| | |
|---|---|
| *satelliteID* | The satellite id number (or PRN) |
| *gnssObs* | The returned GNSS observable data structure |

**Returns**

True if the observable exists

#### 8.43.3.8 getData() [2/4]

```
bool pnt_integrity::RepositoryEntry::getData (
            data::GNSSObservables & gnssObservables ) const  [inline]
```

Returns a GNSSObservables.

**Parameters**

| | |
|---|---|
| *gnssObservables* | The returned GNSS observables |

**Returns**

True if the observable map exists

Definition at line 144 of file RepositoryEntry.hpp.

#### 8.43.3.9 getData() [3/4]

```
void pnt_integrity::RepositoryEntry::getData (
            data::GNSSObservableMap & gnssObsMap ) const  [inline]
```

Returns a GNSS ObservableMap.

**Parameters**

| | |
|---|---|
| *gnssObsMap* | The returned GNSS observable map |

**Returns**

True if the observable map exists

Definition at line 157 of file RepositoryEntry.hpp.

#### 8.43.3.10 getData() [4/4]

```
bool pnt_integrity::RepositoryEntry::getData (
             data::MeasuredRange & range ) const  [inline]
```

Returns the RF range observable.

**Parameters**

| | |
|---|---|
| *range* | The returned value |

Definition at line 176 of file RepositoryEntry.hpp.

#### 8.43.3.11 setLogMessageHandler()

```
void pnt_integrity::RepositoryEntry::setLogMessageHandler (
             const logutils::LogCallback & logMsgHandler )  [inline]
```

Sets the log message handler to provided callback.

**Parameters**

| | |
|---|---|
| *logMsgHandler* | The provided call back function |

Definition at line 237 of file RepositoryEntry.hpp.

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

- include/pnt_integrity/RepositoryEntry.hpp

## 8.44 pnt_integrity::data::RfSpectrum Struct Reference

A structure that represents an RF spectrum measurement.

`#include <IntegrityData.hpp>`

## Public Attributes

- Header header

  *The message header.*
- int span

  *Spectrum span [Hz].*
- int center_frequency

  *Center of spectrum span [Hz].*
- std::vector< double > spectrum

### 8.44.1 Detailed Description

A structure that represents an RF spectrum measurement.

Definition at line 819 of file IntegrityData.hpp.

### 8.44.2 Member Data Documentation

#### 8.44.2.1 spectrum

`std::vector<double> pnt_integrity::data::RfSpectrum::spectrum`

Vector of spectrum measurments [dB] TODO: add comment that defines frequency for each bin

Definition at line 832 of file IntegrityData.hpp.

The documentation for this struct was generated from the following file:

- include/pnt_integrity/IntegrityData.hpp

## 8.45 pnt_integrity::RngPosCheckNodeDiagnostic Struct Reference

Structure for check diagnostics.

`#include <RangePositionCheck.hpp>`

### Public Attributes

- double minCalculatedRange

    *The minimum calculated distance based on both positions and variances.*
- double maxCalculatedRange

    *The maximum calculated distance based on both positions and variances.*
- double minMeasRange

    *The minimum possible distance based on measured range and variance.*
- double maxMeasRange

    *The maximum possible distance based on measured range and variance.*

### 8.45.1 Detailed Description

Structure for check diagnostics.

Definition at line 70 of file RangePositionCheck.hpp.

The documentation for this struct was generated from the following file:

- include/pnt_integrity/RangePositionCheck.hpp

## 8.46 pnt_integrity::StaticPosCheckDiagnostics Struct Reference

Structure used to publish diagnostic data.

`#include <StaticPositionCheck.hpp>`

### Public Attributes

- data::GeodeticPosition3d staticPosition

    *The static position used in the check (surveyed or provided)*
- double posChangeThresh

    *Threshold to check current position against static position.*
- double percentOverThresh

    *The percent of positions in the window that are over the threshold.*
- double inconsistentThresh

    *The threshold used for the INCONSISTENT assurance level.*
- double unassuredThresh

    *The threshold used for the UNASSURED assurance level.*

### 8.46.1 Detailed Description

Structure used to publish diagnostic data.

Definition at line 75 of file StaticPositionCheck.hpp.

The documentation for this struct was generated from the following file:

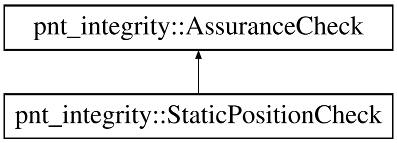
- include/pnt_integrity/StaticPositionCheck.hpp

## 8.47 pnt_integrity::StaticPositionCheck Class Reference

Class implementation for the static-position check.

```
#include <StaticPositionCheck.hpp>
```

Inheritance diagram for pnt_integrity::StaticPositionCheck:


pnt_integrity::AssuranceCheck
pnt_integrity::StaticPositionCheck

### Public Member Functions

- StaticPositionCheck (const std::string &name="static_position_check", const size_t &numPositionsForInit=60, const unsigned int &checkWindowSize=10, const double &posChangeThresh=5.0, const logutils::LogCallback &log=logutils::printLogToStdOut)

  *Default constructor for the check class.*
- bool handlePositionVelocity (const data::PositionVelocity &posVel, const bool &local)

  *Handler function for Position / Velocity message.*
- virtual bool runCheck ()

  *Triggers a manual check calculation.*
- virtual void calculateAssuranceLevel (const double &)

  *Function to explicitly set the assurance level of the check.*
- void setStaticPosition (const data::GeodeticPosition3d &staticPos)

  *Sets the expected static position that will be used for the check.*
- void setPublishDiagnostics (std::function< void(const double &timestamp, const StaticPosCheckDiagnostics &checkData)> handler)

  *Connects the internal publishing function to external interface.*

## Additional Inherited Members

### 8.47.1 Detailed Description

Class implementation for the static-position check.

Definition at line 90 of file StaticPositionCheck.hpp.

### 8.47.2 Constructor & Destructor Documentation

#### 8.47.2.1 StaticPositionCheck()

```
pnt_integrity::StaticPositionCheck::StaticPositionCheck (
             const std::string & name = "static_position_check",
             const size_t & numPositionsForInit = 60,
             const unsigned int & checkWindowSize = 10,
             const double & posChangeThresh = 5.0,
             const logutils::LogCallback & log = logutils::printLogToStdOut )  [inline]
```

Default constructor for the check class.

Constructor explicitly disables multi-prn support.

**Parameters**

| | |
|---|---|
| *name* | The name of the check object |
| *numPositionsForInit* | The number of static positiosn required for the initialization survey |
| *checkWindowSize* | The minimum number of samples (after throwing out invalid positions) necessary to make an informed statement about integrity, includes start positions if in averaging mode |
| *posChangeThresh* | The threshold radius (in meters) for noisy position changes |
| *log* | A provided log callback function to use |

Definition at line 111 of file StaticPositionCheck.hpp.

### 8.47.3 Member Function Documentation

#### 8.47.3.1 calculateAssuranceLevel()

```
virtual void pnt_integrity::StaticPositionCheck::calculateAssuranceLevel (
             const double &  )  [inline], [virtual]
```

Function to explicitly set the assurance level of the check.

Uses whatever data is available to calculate the current assurance level

Implements pnt_integrity::AssuranceCheck.

Definition at line 154 of file StaticPositionCheck.hpp.

### 8.47.3.2 handlePositionVelocity()

```
bool pnt_integrity::StaticPositionCheck::handlePositionVelocity (
             const data::PositionVelocity & posVel,
             const bool & local )  [virtual]
```

Handler function for Position / Velocity message.

Function to handle provided posivion / velocity messages

**Parameters**

| | |
|---|---|
| *posVel* | The provided position velocity message / structure |
| *local* | Indicates if this is a local or remote message |

**Returns**

True if successful

Reimplemented from pnt_integrity::AssuranceCheck.

### 8.47.3.3 runCheck()

```
virtual bool pnt_integrity::StaticPositionCheck::runCheck ( )  [virtual]
```

Triggers a manual check calculation.

Use this function to run a manual check calculation that is not triggered off the receipt of a message (pure virtual)

**Returns**

True if successful

Implements pnt_integrity::AssuranceCheck.

#### 8.47.3.4 setPublishDiagnostics()

```
void pnt_integrity::StaticPositionCheck::setPublishDiagnostics (
             std::function< void(const double &timestamp, const StaticPosCheckDiagnostics &check←
Data)> handler )  [inline]
```

Connects the internal publishing function to external interface.

This function connects the internal "publishDiagnostics" function to an external, custom function of choice

**Parameters**

| | |
|---|---|
| *handler* | Provided handler function |

Definition at line 167 of file StaticPositionCheck.hpp.

#### 8.47.3.5 setStaticPosition()

```
void pnt_integrity::StaticPositionCheck::setStaticPosition (
             const data::GeodeticPosition3d & staticPos )
```

Sets the expected static position that will be used for the check.

**Parameters**

| | |
|---|---|
| *staticPos* | The provided position value |

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

- include/pnt_integrity/StaticPositionCheck.hpp

## 8.48 pnt_integrity::Subframe1Fault Struct Reference

### Public Attributes

- bool **towSf1**
- bool **weekNumber**
- bool **codeOnL2**
- bool **uraIndex**
- bool **svHealth**
- bool **iodc**
- bool **l2PDataFlag**
- bool **groupDelay**
- bool **clockCorrectionTime**
- bool **clockAging3**
- bool **clockAging2**
- bool **clockAging1**

### 8.48.1 Detailed Description

Definition at line 166 of file GPSEphemeris.hpp.

The documentation for this struct was generated from the following file:

- include/pnt_integrity/GPSEphemeris.hpp

## 8.49 pnt_integrity::Subframe2Fault Struct Reference

### Public Attributes

- bool **towSf2**
- bool **iodeSf2**
- bool **sinOrbitRadius**
- bool **meanMotionDifference**
- bool **meanAnomaly**
- bool **cosLatitude**
- bool **eccentricity**
- bool **sinLatitude**
- bool **sqrtSemiMajorAxis**
- bool **timeOfEphemeris**
- bool **fitInterval**
- bool **ageOfDataOffset**

### 8.49.1 Detailed Description

Definition at line 182 of file GPSEphemeris.hpp.

The documentation for this struct was generated from the following file:

- include/pnt_integrity/GPSEphemeris.hpp

## 8.50 pnt_integrity::Subframe3Fault Struct Reference

### Public Attributes

- bool **towSf3**
- bool **cosInclination**
- bool **rightAscension**
- bool **sinInclination**
- bool **inclinationAngle**
- bool **cosOrbitRadius**
- bool **argumentOfPerigee**
- bool **ascensionRate**
- bool **iodeSf3**
- bool **inclinationRate**

### 8.50.1 Detailed Description

Definition at line 198 of file GPSEphemeris.hpp.

The documentation for this struct was generated from the following file:

- include/pnt_integrity/GPSEphemeris.hpp

## 8.51 pnt_integrity::SVAlmHealth Struct Reference

```
#include <GPSAlmanac.hpp>
```

### Public Attributes

- SVNavHealth **navDataHealth**
- SVSignalHealth **signalHealth**

### 8.51.1 Detailed Description

Structure to hold the satellite health field given in bits 18 to 22 of subframe 1 and in the bottom 5 bits of the 8 bit satellite health field in Almanac subframes 4 and 5 Defined in paragraph 20.3.3.5.1.3 of IS-GPS-200

Definition at line 68 of file GPSAlmanac.hpp.

The documentation for this struct was generated from the following file:

- include/pnt_integrity/GPSAlmanac.hpp

## 8.52 pnt_integrity::SVHealth Struct Reference

Structure to hold the SV health status from subframe 1, word 3, bits 17-22.

```
#include <GPSEphemeris.hpp>
```

### Public Attributes

- bool someOrAllNavDataBad

  *Summary of the navigation data health field given in the ephemeris data.*
- SVSignalHealth signalHealth

  *5-bit satellite signal health given in the ephemeris data*

### 8.52.1 Detailed Description

Structure to hold the SV health status from subframe 1, word 3, bits 17-22.

Definition at line 106 of file GPSEphemeris.hpp.

The documentation for this struct was generated from the following file:

- include/pnt_integrity/GPSEphemeris.hpp

## 8.53 pnt_integrity::TimeEntry Struct Reference

Structure for a time entry into the repository.

`#include <IntegrityDataRepository.hpp>`

### Public Member Functions

- TimeEntry ()

  *Default constructor.*
- TimeEntry (const double &timeOfWeek)

  *Constructor for creation of entry with time field already known.*

### Public Attributes

- double timeOfWeek_

  *The time of week the data were measured or correspond to.*
- RepositoryEntry localData_

  *The local observables.*
- RemoteRepoEntries remoteData_

  *A map of remote observables.*

### 8.53.1 Detailed Description

Structure for a time entry into the repository.

The structure contains the time corresponding to the observables, the local observables, and a set of remote observables contained in a map that is keyed off of remote node id

Definition at line 61 of file IntegrityDataRepository.hpp.

### 8.53.2 Constructor & Destructor Documentation

#### 8.53.2.1 TimeEntry() `[1/2]`

```
pnt_integrity::TimeEntry::TimeEntry ( )  [inline]
```

Default constructor.

Declaring the default constructor implicitly allows for copy construction which is used when a new time entry is created

Definition at line 74 of file IntegrityDataRepository.hpp.

#### 8.53.2.2 TimeEntry() `[2/2]`

```
pnt_integrity::TimeEntry::TimeEntry (
              const double & timeOfWeek )  [inline]
```

Constructor for creation of entry with time field already known.

**Parameters**

| | |
|---|---|
| *timeOfWeek* | The time of week that all observables in the entry correspond to |

Definition at line 80 of file IntegrityDataRepository.hpp.

The documentation for this struct was generated from the following file:

- include/pnt_integrity/IntegrityDataRepository.hpp

## 8.54 pnt_integrity::data::Timestamp Struct Reference

A timestamp used in all headers.

`#include <IntegrityData.hpp>`

### Public Member Functions

- Timestamp (const int64_t &secIn=0, const int32_t &nsIn=0, const int8_t &timecodeIn=0)

    *Default constructor for timestamp.*

### Public Attributes

- int64_t sec

    *The whole seconds portion of the timestamp.*
- int32_t nanoseconds
- int8_t timecode

### 8.54.1 Detailed Description

A timestamp used in all headers.

Definition at line 57 of file IntegrityData.hpp.

### 8.54.2 Constructor & Destructor Documentation

#### 8.54.2.1 Timestamp()

```
pnt_integrity::data::Timestamp::Timestamp (
             const int64_t & secIn = 0,
             const int32_t & nsIn = 0,
             const int8_t & timecodeIn = 0 )  [inline]
```

Default constructor for timestamp.

**Parameters**

| | |
|---|---|
| *secIn* | The whole seconds portion of the timestamp |
| *nsIn* | Fractional portion of the timestamp represented in ns |
| *timecode←In* | Indicator for timebase, 0 for TAI, non-zero for other |

Definition at line 75 of file IntegrityData.hpp.

### 8.54.3 Member Data Documentation

#### 8.54.3.1 nanoseconds

```
int32_t pnt_integrity::data::Timestamp::nanoseconds
```

Fractional portion of the timestamp represented in ns, giving the timestamp 1 ns resolution

Definition at line 64 of file IntegrityData.hpp.

#### 8.54.3.2 timecode

`int8_t pnt_integrity::data::Timestamp::timecode`

Indicator for timebase, 0 if synced to TAI, non-zero if device using a specific timebase

Definition at line 68 of file IntegrityData.hpp.

The documentation for this struct was generated from the following file:

- include/pnt_integrity/IntegrityData.hpp

# Chapter 9

# File Documentation

## 9.1 include/pnt_integrity/AcquisitionCheck.hpp File Reference

Class defined for the acquisition level checks.

```
#include "if_data_utils/IFSampleData.hpp"
#include "pnt_integrity/AssuranceCheck.hpp"
#include <Eigen/Dense>
#include <Eigen/StdVector>
#include <list>
#include <map>
#include <unsupported/Eigen/FFT>
#include <vector>
```

### Classes

- struct pnt_integrity::AcqCheckDiagnostics

  *Structure for publishing Acquisition Check diagnostics.*
- class pnt_integrity::AcquisitionCheck

  *Class implementation for the acquisition check.*

### Namespaces

- pnt_integrity

  *Namespace for all pnt_integrity applications.*

## Typedefs

- using pnt_integrity::CodeMap = std::map< int, std::vector< float > >

  *A map for holding PRN codes, indexed on prn.*
- using pnt_integrity::CodeMapEntry = std::pair< int, std::vector< float > >

  *A pair for holding a PRN and it's code.*
- using pnt_integrity::CodeFreqMap = std::map< int, Eigen::ArrayXcf >

  *A map for holding frequency bin values.*
- using pnt_integrity::CodeFreqMapEntry = std::pair< int, Eigen::ArrayXcf >

  *A pair for holding a frequency bin number its values.*
- using pnt_integrity::CorrelationResultsMap = std::map< int, Eigen::ArrayXXf >

  *A map that stores the correlation results for a prn.*
- using pnt_integrity::PeakResultsMap = std::map< int, std::pair< double, double > >
- using pnt_integrity::PrnList = std::vector< int >

  *A vector type for a list of prns.*

## Variables

- const std::string pnt_integrity::INTEGRITY_ACQ_PEAK_VALS = "INTEGRITY_ACQ_PEAK_VALS"

  *String ID for the ACQ check peak vals.*
- const std::string pnt_integrity::INTEGRITY_ACQ_PEAK1_KEY = "INT_ACQ_PEAK1_"

  *String ID for the ACQ check peak 1 key.*
- const std::string pnt_integrity::INTEGRITY_ACQ_PEAK2_KEY = "INT_ACQ_PEAK2_"

  *String ID for the ACQ check peak 2 key.*
- const std::string pnt_integrity::INTEGRITY_ACQ_DIAGNOSTICS = "INTEGRITY_ACQ_DIAGNOSTICS"

  *String ID for the ACQ check diagnostic data.*
- const std::string pnt_integrity::INT_ACQ_DIAG_HI_PWR_THRESH = "INT_ACQ_DIAG_HI_PWR_THRESH"

  *String ID for the ACQ check high power threshold.*
- const std::string pnt_integrity::INT_ACQ_DIAG_PEAK_RATIO_THRESH

  *String ID for the ACQ check peak ratio threshold.*
- const std::string pnt_integrity::INT_ACQ_DIAG_ACQ_THRESH = "INT_ACQ_DIAG_ACQ_THRESH"

  *String ID for the ACQ check acquisition threshold.*
- const std::string pnt_integrity::INT_ACQ_DIAG_ITHRESH = "INT_ACQ_DIAG_ITHRESH"

  *String ID for the ACQ check survey inconsistent thresh.*
- const std::string pnt_integrity::INT_ACQ_DIAG_UTHRESH = "INT_ACQ_DIAG_UTHRESH"

  *String ID for the ACQ check survey unassured thresh.*
- const std::string pnt_integrity::INT_ACQ_DIAG_ICOUNT = "INT_ACQ_DIAG_ICOUNT"

  *String ID for the ACQ check survey inconsistent count.*
- const std::string pnt_integrity::INT_ACQ_DIAG_UCOUNT = "INT_ACQ_DIAG_UCOUNT"

  *String ID for the ACQ check survey unassured count.*
- const std::string pnt_integrity::INT_ACQ_DIAG_PEAK_RATIO_KEY = "INT_ACQ_DIAG_PEAK_RATIO_KEY↩
_"

  *String ID for the ACQ check survey peak ratio key.*

### 9.1.1 Detailed Description

Class defined for the acquisition level checks.

**Author**

Josh Clanton `josh.clanton@is4s.com`

**Date**

September 30, 2019

## 9.2 include/pnt_integrity/AgcCheck.hpp File Reference

Class defined for the AGC check.

`#include "pnt_integrity/AssuranceCheck.hpp"`

### Classes

- struct pnt_integrity::AgcCheckDiagnostics

  *Diagnostic data for AGC check.*
- class pnt_integrity::AgcCheck

  *Class implementation for the AGC check.*

### Namespaces

- pnt_integrity

  *Namespace for all pnt_integrity applications.*

### Variables

- const std::string pnt_integrity::INTEGRITY_AGC_DIAGNOSTICS = "INTEGRITY_AGC_DIAGNOSTICS"

  *String ID for the AGC check diagnostic data.*
- const std::string pnt_integrity::INTEGRITY_AGC_DIAG_ITHRESH = "INTEGRITY_AGC_DIAG_ITHRESH"

  *String ID for the AGC check survey inconsistent thresh.*

### 9.2.1 Detailed Description

Class defined for the AGC check.

**Author**

Josh Clanton `josh.clanton@is4s.com`

**Date**

February 18, 2020

## 9.3 include/pnt_integrity/AngleOfArrivalCheck.hpp File Reference

AssurancCheck class defined for the angle of arrival check.

```
#include <chrono>
#include "pnt_integrity/AssuranceCheck.hpp"
#include "pnt_integrity/IntegrityData.hpp"
```

### Classes

- struct pnt_integrity::AoaCheckDiagnostics

  *Structure used to publish diagnostic data.*
- class pnt_integrity::AngleOfArrivalCheck

  *Class implementation for the angle of arrival check.*

### Namespaces

- pnt_integrity

  *Namespace for all pnt_integrity applications.*

### Typedefs

- using pnt_integrity::SingleDiffMap = std::map< int, double >

  *Defines a type that maps PRN to a calculated difference.*
- using pnt_integrity::PrnAssuranceEachNode = std::map< int, std::vector< data::AssuranceLevel > >

  *Defines a map that holds an assurance level for each prn for each node.*

### Enumerations

- enum pnt_integrity::AoaCheckData { **UsePseudorange** = 0, **UseCarrierPhase**, **UseBoth** }

  *Enumeration to indicate what data field to use for the AOA check.*

### Variables

- const std::string pnt_integrity::INTEGRITY_AOA_DIFF_DIAGNOSTICS

  *String ID for the AOA check difference diagnostic data.*
- const std::string pnt_integrity::INTEGRITY_AOA_DIFF_NODE_ID = "INTEGRITY_AOA_DIFF_NODE_ID"

  *String ID for the AOA check diagnostic node id.*
- const std::string pnt_integrity::INTEGRITY_AOA_DIAGNOSTICS = "INTEGRITY_AOA_DIAGNOSTICS"

  *String ID for the AOA check diagnostic data.*
- const std::string pnt_integrity::INTEGRITY_AOA_DIAG_DIFF_THRESH

  *String ID for the AOA check diagnostic difference threshold.*
- const std::string pnt_integrity::INTEGRITY_AOA_DIAG_SUSPECT_PRN_PERCENT

  *String ID for the AOA check diagnostic suspect prn percent.*
- const std::string pnt_integrity::INTEGRITY_AOA_DIAG_UNAVAILABLE_PRN_PERCENT

  *String ID for the AOA check diagnostic unavailable prn percent.*
- const std::string pnt_integrity::INTEGRITY_AOA_DIAG_ASSURED_PRN_PERCENT

  *String ID for the AOA check diagnostic assured prn percent.*
- const std::string pnt_integrity::INTEGRITY_AOA_DIAG_ITHRESH = "INTEGRITY_AOA_DIAG_ITHRESH"

  *String ID for the AOA check survey inconsistent thresh.*
- const std::string pnt_integrity::INTEGRITY_AOA_DIAG_UTHRESH = "INTEGRITY_AOA_DIAG_UTHRESH"

  *String ID for the AOA check survey unassured thresh.*
- const std::string pnt_integrity::INTEGRITY_AOA_DIAG_ATHRESH = "INTEGRITY_AOA_DIAG_ATHRESH"

  *String ID for the AOA check survey assured thresh.*

### 9.3.1 Detailed Description

AssurancCheck class defined for the angle of arrival check.

**Author**

Josh Clanton `josh.clanton@is4s.com`

**Date**

June 3, 2019

## 9.4 include/pnt_integrity/AssuranceCheck.hpp File Reference

Base / parent class for a PNT assurance check.

```
#include "if_data_utils/IFSampleData.hpp"
#include "logutils/logutils.hpp"
#include "pnt_integrity/IntegrityData.hpp"
#include "pnt_integrity/IntegrityDataRepository.hpp"
```

## Classes

- class pnt_integrity::AssuranceCheck

  *Parent class for all integrity checks.*

## Namespaces

- pnt_integrity

  *Namespace for all pnt_integrity applications.*

## Typedefs

- using pnt_integrity::MultiPrnAssuranceMap = std::map< int, data::AssuranceLevel >

  *A map for pairing an assurance level to each PRN.*

### 9.4.1 Detailed Description

Base / parent class for a PNT assurance check.

**Author**

Josh Clanton `josh.clanton@is4s.com`

**Date**

May 28, 2019

## 9.5 include/pnt_integrity/ClockBiasCheck.hpp File Reference

AssurancCheck class defined for the clock bias check.

```
#include <cstring>
#include "pnt_integrity/AssuranceCheck.hpp"
```

## Classes

- struct pnt_integrity::ClockBiasCheckDiagnostics

  *Structure used to publish diagnostic data.*
- class pnt_integrity::ClockBiasCheck

  *Class implementation for the position velocity check.*

### Namespaces

- pnt_integrity

  *Namespace for all pnt_integrity applications.*

### Variables

- const std::string pnt_integrity::INTEGRITY_CLOCK_BIAS_DIAGNOSTICS

  *String ID for the clock-bias check diagnostic data.*
- const std::string pnt_integrity::INTEGRITY_CLOCK_BIAS_DIAG_EXP_DRIFT

  *String ID for the clock-bias check expected drift.*
- const std::string pnt_integrity::INTEGRITY_CLOCK_BIAS_DIAG_EXP_DRIFT_VAR

  *String ID for the clock-bias check drift variance.*
- const std::string pnt_integrity::INTEGRITY_CLOCK_BIAS_DIAG_PROP_OFFSET

  *String ID for the clock-bias check propagation offset.*
- const std::string pnt_integrity::INTEGRITY_CLOCK_BIAS_DIAG_ACTUAL_OFFSET

  *String ID for the clock-bias check actual offset.*
- const std::string pnt_integrity::INTEGRITY_CLOCK_BIAS_DIAG_OFFSET_ERROR

  *String ID for the clock-bias check offset error.*
- const std::string pnt_integrity::INTEGRITY_CLOCK_BIAS_DIAG_DRIFT_RATE_BOUND

  *String ID for the clock-bias check drift rate bound.*
- const std::string pnt_integrity::INTEGRITY_CLOCK_BIAS_DIAG_DRIFT_RATE_VAR_BOUND

  *String ID for the clock-bias check drift rate var bound.*

### 9.5.1 Detailed Description

AssurancCheck class defined for the clock bias check.

**Author**

Josh Clanton `josh.clanton@is4s.com`
John David Sprunger `jss0027@tigermail.auburn.edu`

**Date**

December 17, 2019

## 9.6 include/pnt_integrity/CnoCheck.hpp File Reference

Class defined for the carrier-to-noise ratio (Cno) checks.

`#include "pnt_integrity/AssuranceCheck.hpp"`

## Classes

- struct pnt_integrity::CnoCheckDiagnostics

  *Diagnostic data for the check.*
- class pnt_integrity::CnoCheck

  *Class implementation of the carrier-to-noise (CnO) assurance check. The check analyzes the CnO values for abnormalities.*

## Namespaces

- pnt_integrity

  *Namespace for all pnt_integrity applications.*

## Variables

- const std::string pnt_integrity::INTEGRITY_CN0_DIAGNOSTICS = "INTEGRITY_CN0_DIAGNOSTICS"

  *String ID for the CNO check diagnostic data.*
- const std::string pnt_integrity::INTEGRITY_CN0_DIAG_AVG_COUNT = "INTEGRITY_CN0_DIAG_AVG_CO↩ UNT"

  *String ID for the CNO check average count.*
- const std::string pnt_integrity::INTEGRITY_CN0_DIAG_ITHRESH = "INTEGRITY_CN0_DIAG_ITHRESH"

  *String ID for the CNO check survey inconsistent thresh.*
- const std::string pnt_integrity::INTEGRITY_CN0_DIAG_UTHRESH = "INTEGRITY_CN0_DIAG_UTHRESH"

  *String ID for the CNO check survey unassured thresh.*

### 9.6.1 Detailed Description

Class defined for the carrier-to-noise ratio (Cno) checks.

**Author**

Josh Clanton `josh.clanton@is4s.com`

**Date**

October 23, 2019

## 9.7 include/pnt_integrity/GPSAlmanac.hpp File Reference

Stores a set of GPS almanac data and computes satellite pos.

```
#include <cstdint>
#include <string>
#include "pnt_integrity/GPSNavDataCommon.hpp"
```

## Classes

- struct pnt_integrity::SVAlmHealth
- struct pnt_integrity::AlmanacParameters
- union pnt_integrity::AlmanacSubframeFaults
- struct pnt_integrity::AlmanacSubframeFaults::FaultType
- class pnt_integrity::GpsAlmanac

  *Class to parse and store almanac data for a GPS Satellite.*

## Namespaces

- pnt_integrity

  *Namespace for all pnt_integrity applications.*

## Enumerations

- enum pnt_integrity::SVNavHealth : uint8_t {
  **AllDataOK** = 0, **ParityFailure** = 1, **TlmHowFormatProblem** = 2, **ZCountInHowBad** = 3,
  **Subframe_1_2_or_3_Bad** = 4, **Subframe_4_or_5_Bad** = 5, **AllUploadedDataBad** = 6, **AllDataBad** = 7 }

### 9.7.1 Detailed Description

Stores a set of GPS almanac data and computes satellite pos.

#### Author

David Hodo `david.hodo@is4s.com`

#### Date

January 2015

## 9.8 include/pnt_integrity/GPSEphemeris.hpp File Reference

Stores a set of GPS ephemeris data and computes satellite pos.

```
#include <stdint.h>
#include <limits>
#include <string>
#include <utility>
#include "pnt_integrity/GPSNavDataCommon.hpp"
```

### Classes

- struct pnt_integrity::SVHealth

  *Structure to hold the SV health status from subframe 1, word 3, bits 17-22.*
- struct pnt_integrity::EphemerisParameters
- struct pnt_integrity::Subframe1Fault
- struct pnt_integrity::Subframe2Fault
- struct pnt_integrity::Subframe3Fault
- class pnt_integrity::GpsEphemeris

### Namespaces

- pnt_integrity

  *Namespace for all pnt_integrity applications.*

### Enumerations

- enum pnt_integrity::AntiSpoofFlag { **Off** = 0, **On** = 1 }
- enum pnt_integrity::L2CodeType { **Reserved** = 0, **PCodeOn** = 1, **CACodeOn** = 2, **CAAndPCodeOn** = 3 }
- enum pnt_integrity::FitInterval { **FourHrs** = 0, **GreaterThanFourHrs** = 1 }
- enum pnt_integrity::AlertFlag { **ALERT_OFF** = 0, **ALERT_RAISED** = 1 }
- enum pnt_integrity::L2NavDataFlag { **On** = 0, **Off** = 1 }

### 9.8.1 Detailed Description

Stores a set of GPS ephemeris data and computes satellite pos.

#### Author

William Travis `william.travis@is4s.com`
David Hodo `david.hodo@is4s.com`

#### Date

August 2013

## 9.9 include/pnt_integrity/GPSNavDataCommon.hpp File Reference

Common structures and functions used in processing GPS LNAV data.

```
#include <cstdint>
#include <string>
```

## Namespaces

- pnt_integrity

  *Namespace for all pnt_integrity applications.*

## Enumerations

- enum pnt_integrity::SVSignalHealth : uint8_t {
**AllSignalsOk** = 0, **AllSignalsWeak** = 1, **AllSignalsDead** = 2, **AllSignalsHaveNoDataModulation** = 3,
**L1PSignalWeak** = 4, **L1PSignalDead** = 5, **L1PSignalHasNoDataModulation** = 6, **L2PSignalWeak** = 7,
**L2PSignalDead** = 8, **L2PSignalHasNoDataModulation** = 9, **L1CSignalWeak** = 10, **L1CSignalDead** = 11,
**L1CSignalHasNoDataModulation** = 12, **L2CSignalWeak** = 13, **L2CSignalDead** = 14, **L2CSignalHasNoDataModulation** = 15,
**L1AndL2PSignal_Weak** = 16, **L1AndL2PSignal_Dead** = 17, **L1AndL2PSignal_HasNoDataModulation** = 18,
**L1AndL2CSignal_Weak** = 19,
**L1AndL2CSignal_Dead** = 20, **L1AndL2CSignal_HasNoDataModulation** = 21, **L1SignalWeak** = 22, **L1SignalDead** = 23,
**L1SignalHasNoDataModulation** = 24, **L2SignalWeak** = 25, **L2SignalDead** = 26, **L2SignalHasNoDataModulation** = 27,
**SVIsTemporarilyOutDoNotUse** = 28, **SVWillBeTemporarilyOutUseWithCaution** = 29, **OneOrMoreSignalsDeforedURAStillValid** = 30, **MoreThanOneCombinationNeededToDescribeAnomalies** = 31 }
- enum pnt_integrity::NavDataTimeOfArrival { **Older**, **Same**, **Newer** }

  *Enumeration to define the relative time between multiple LNAV data sets.*

## Functions

- void pnt_integrity::fromHex (const std::string &in, void ∗const data)

  *Convert hexadecimal string to char array.*
- void pnt_integrity::toHex (unsigned char ∗const byteData, const size_t dataLength, std::string &dest)

  *Convert char array to hexadecimal string.*
- void pnt_integrity::convertSubframeFrom10To30Word (const uint32_t(&sfIn)[10], uint8_t(&sfOut)[30])

  *Converts uint32_t[10] subframe to uint8_t[30] array.*
- void pnt_integrity::convertSubframeFrom30To10Word (const uint8_t(&sfIn)[30], uint32_t(&sfOut)[10])

  *Converts uint8_t[30] subframe to uint32_t[10] array.*
- void pnt_integrity::removeSubframeParity (const uint32_t(&subframeWordsIn)[10], uint32_t(&subframeWordsOut)[10])

  *Remove parity bits from subframe.*
- uint16_t pnt_integrity::parseSubframeID (const uint8_t(&subframe)[30])

  *Parse a subframe and return its ID number.*
- void pnt_integrity::parseSubframeID (const uint8_t(&subframe)[30], uint16_t &subframeID)

  *Parse a subframe and return its ID number.*
- double pnt_integrity::parseTimeOfWeek (const uint8_t(&subframe)[30])

  *Parse a subframe and return the time of week.*
- void pnt_integrity::parseTimeOfWeek (const uint8_t(&subframe)[30], double &tow)

  *Parse a subframe and return the time of week.*

### Variables

- const double pnt_integrity::gpsPi = 3.1415926535898

  *PI as defined in IS-GPS-200 (30.3.3.1.3)*
- const double pnt_integrity::twoGpsPi = 2.0 ∗ gpsPi

  *2 ∗ PI as defined in IS-GPS-200 (convenience constant)*
- const double pnt_integrity::speedOfLight = 2.99792458e8

  *Speed of light as defined in IS-GPS-200 (20.3.4.3) [m/s].*
- const double pnt_integrity::gpsGM = 3.986005e14

  *Earth gravitational constant as defined in IS-GPS-200 (Tbl. 30-II) [m^3/s^2].*
- const double pnt_integrity::gpsF = -4.442807633e-10

  *Flattening constant as defined in IS-GPS-200 (20.3.3.3.3) [sec/meter^0.5].*
- const double pnt_integrity::gpsEarthRotationRate = 7.2921151467e-5
- const double pnt_integrity::secondsInWeek = 604800.0

  *Number of GPS seconds in a week.*
- const double pnt_integrity::secondsInHalfWeek = secondsInWeek / 2.0

  *Number of GPS seconds in a half week.*

### 9.9.1 Detailed Description

Common structures and functions used in processing GPS LNAV data.

**Author**

David Hodo `david.hodo@is4s.com`

**Date**

April 2021

## 9.10 include/pnt_integrity/IntegrityData.hpp File Reference

Defines all data types and structure definitions.

```
#include <cmath>
#include <cstdint>
#include <iostream>
#include <limits>
#include <map>
#include <string>
#include <vector>
```

## Classes

- struct pnt_integrity::data::Timestamp

  *A timestamp used in all headers.*
- struct pnt_integrity::data::GNSSTime

  *A GNSS time.*
- struct pnt_integrity::data::Header

  *The header used for all associated data types.*
- struct pnt_integrity::data::ClockOffset

  *A structure for measuring the offset between two clocks.*
- class pnt_integrity::data::AssuranceState

  *A structure to hold an AssuranceLevel and value.*
- struct pnt_integrity::data::AssuranceReport

  *A structure to hold a single assurance report.*
- struct pnt_integrity::data::AssuranceReports

  *A structure to hold assurance data for all registered checks.*
- struct pnt_integrity::data::GNSSObservable

  *A structure for GNSS observables (pseudorange, carrier, doppler, etc)*
- struct pnt_integrity::data::GNSSObservables

  *The GNSSObservables message.*
- struct pnt_integrity::data::GNSSSubframe

  *GNSS Subframe data.*
- struct pnt_integrity::data::GeodeticPosition3d

  *A structure to represent 3D geodetic position.*
- struct pnt_integrity::data::PositionVelocity

  *A structure to represent a Position / Velocity message.*
- struct pnt_integrity::data::AccumulatedDistranceTraveled

  *A structure that represents a distance traveled over a time period.*
- struct pnt_integrity::data::IMU

  *A structure that represents IMU measurement data.*
- struct pnt_integrity::data::MeasuredRange

  *A structure that represents a distance measurement to a known point.*
- struct pnt_integrity::data::RfSpectrum

  *A structure that represents an RF spectrum measurement.*
- struct pnt_integrity::data::AgcValue

  *A structure to represent an AGC measurement.*

## Namespaces

- pnt_integrity

  *Namespace for all pnt_integrity applications.*
- pnt_integrity::data

  *Namespace for all integrity data definitions.*

### Typedefs

- using pnt_integrity::data::GNSSObservableMap = std::map< uint64_t, GNSSObservable >

  *A map to relate a GNSSObservable to a PRN.*

### Enumerations

- enum pnt_integrity::data::TimeSystem { **GLO** = 0, **GPS**, **GAL**, **BDT** }

  *Enumeration for all available satellite-based time system sources.*
- enum pnt_integrity::data::SatelliteSystem : uint8_t {
  **GPS** = 0, **Glonass**, **Galileo**, **QZSS**,
  **BeiDou**, **IRNSS**, **SBAS**, **Mixed**,
  **Other** }

  *Enumeration for satellite system identification.*
- enum pnt_integrity::data::FrequencyBand : uint8_t {
  **Band1** = 0, **Band2**, **Band5**, **Band6**,
  **Band7**, **Band8**, **Band9**, **Band0**,
  **Band10** }

  *Defines all possible frequency types.*
- enum pnt_integrity::data::CodeType : uint8_t {
  **SigP** = 0, **SigC**, **SigD**, **SigY**,
  **SigM**, **SigN**, **SigA**, **SigB**,
  **SigI**, **SigQ**, **SigS**, **SigL**,
  **SigX**, **SigW**, **SigZ**, **SigBLANK** }

  *Defines all possible code types.*
- enum pnt_integrity::data::AssuranceLevel : int8_t { **Unavailable** = 0, **Unassured**, **Inconsistent**, **Assured** }

  *Defines all available assurance level values.*

### 9.10.1 Detailed Description

Defines all data types and structure definitions.

**Author**

Josh Clanton `josh.clanton@is4s.com`

**Date**

May 28, 2019

## 9.11 include/pnt_integrity/IntegrityDataRepository.hpp File Reference

Defines the IntegrityDataRepository class in pnt_integrity.

```
#include <atomic>
#include <deque>
#include <iostream>
#include <mutex>
#include <sstream>
#include <vector>
#include "logutils/logutils.hpp"
#include "pnt_integrity/RepositoryEntry.hpp"
```

## Classes

- struct pnt_integrity::TimeEntry

  *Structure for a time entry into the repository.*
- class pnt_integrity::IntegrityDataRepository

  *Class definition for the history of data at a single PNT node.*

## Namespaces

- pnt_integrity

  *Namespace for all pnt_integrity applications.*

## Typedefs

- using pnt_integrity::RemoteRepoEntries = std::map< std::string, RepositoryEntry >

  *A type to map remote entries to their node name / device id.*
- using pnt_integrity::TimeEntryHistory = std::map< double, TimeEntry >

### 9.11.1 Detailed Description

Defines the IntegrityDataRepository class in pnt_integrity.

**Author**

Josh Clanton `josh.clanton@is4s.com`

**Date**

May 28, 2019

## 9.12 include/pnt_integrity/IntegrityMonitor.hpp File Reference

Defines the IntegrityMonitor class in pnt_integrity.

```
#include "logutils/logutils.hpp"
#include "pnt_integrity/AssuranceCheck.hpp"
#include <iomanip>
#include <memory>
#include <mutex>
#include <shared_mutex>
#include <sstream>
#include <vector>
```

## Classes

- class pnt_integrity::IntegrityMonitor

  *Class implementation of integrity monitoring using AssuranceChecks and IntegrityData.*

## Namespaces

- pnt_integrity

  *Namespace for all pnt_integrity applications.*

## Typedefs

- using pnt_integrity::AssuranceChecks = std::map$<$ std::string, AssuranceCheck $*$ $>$

  *A vector type for a collection of AssuranceChecks.*

### 9.12.1 Detailed Description

Defines the IntegrityMonitor class in pnt_integrity.

**Author**

Josh Clanton `josh.clanton@is4s.com`

**Date**

May 28, 2019

## 9.13 include/pnt_integrity/NavigationDataCheck.hpp File Reference

AssuranceCheck class for checking broadcast navigation data.

```
#include "pnt_integrity/AssuranceCheck.hpp"
#include "pnt_integrity/GPSAlmanac.hpp"
#include "pnt_integrity/GPSEphemeris.hpp"
#include <future>
#include <thread>
```

## Classes

- struct pnt_integrity::NavDataCheckDiagnostics

  *Structure for check diagnostics.*
- class pnt_integrity::NavigationDataCheck

  *Class implementation for the navigation data check.*

## Namespaces

- pnt_integrity

    *Namespace for all pnt_integrity applications.*

## Variables

- const std::string pnt_integrity::INTEGRITY_NAV_DATA_DIAGNOSTICS

    *String ID for the nav data check diagnostic data.*
- const std::string pnt_integrity::INTEGRITY_NAV_DATA_VALID = "INTEGRITY_NAV_DATA_VALID"

    *String ID for the nav data check data valid flag.*
- const std::string pnt_integrity::INTEGRITY_NAV_DATA_VALID_MSG = "INTEGRITY_NAV_DATA_VALID_M←SG"

    *String ID for the nav data check data valid msg.*
- const std::string pnt_integrity::INTEGRITY_NAV_DATA_TOW_VALID = "INTEGRITY_NAV_DATA_TOW_VA←LID"

    *String ID for the nav data check tow valid flag.*
- const std::string pnt_integrity::INTEGRITY_NAV_DATA_TOW_VALID_MSG

    *String ID for the nav data check tow valid flag msg.*
- const std::string pnt_integrity::INTEGRITY_NAV_DATA_WN_VALID = "INTEGRITY_NAV_DATA_WN_VALID"

    *String ID for the nav data check week number valid flag.*
- const std::string pnt_integrity::INTEGRITY_NAV_DATA_WN_VALID_MSG

    *String ID for the nav data check week number valid flag msg.*

### 9.13.1 Detailed Description

AssuranceCheck class for checking broadcast navigation data.

**Author**

David Hodo `david.hodo@is4s.com`

**Date**

May 14, 2021

## 9.14 include/pnt_integrity/PositionJumpCheck.hpp File Reference

AssuranceCheck class defined for the position jump check.

```
#include "pnt_integrity/AssuranceCheck.hpp"
#include "pnt_integrity/GeodeticConverter.hpp"
```

### Classes

- struct pnt_integrity::PosJumpCheckDiagnostics

  *Structure for check diagnostics.*
- class pnt_integrity::PositionJumpCheck

  *Class implementation for the position-jump check.*

### Namespaces

- pnt_integrity

  *Namespace for all pnt_integrity applications.*

### Variables

- const std::string pnt_integrity::INTEGRITY_POS_JUMP_DIAGNOSTICS

  *String ID for the position-jump check diagnostic data.*
- const std::string pnt_integrity::INTEGRITY_POS_JUMP_DIAG_BOUND

  *String ID for the position-jump check bound.*
- const std::string pnt_integrity::INTEGRITY_POS_JUMP_DIAG_DIST = "INTEGRITY_POS_JUMP_DIAG_DIST"

  *String ID for the position-jump check distance.*

### 9.14.1 Detailed Description

AssuranceCheck class defined for the position jump check.

**Author**

Will Travis `will.travis@is4s.com`
Josh Clanton `josh.clanton@is4s.com`

**Date**

November 27, 2019

## 9.15 include/pnt_integrity/PositionVelocityConsistencyCheck.hpp File Reference

AssurancCheck class defined for the position velocity consistency check.

```
#include <cstring>
#include "pnt_integrity/AssuranceCheck.hpp"
```

### Classes

- struct pnt_integrity::PosVelConsCheckDiagnostics

  *Structure used to publish diagnostic data.*
- class pnt_integrity::PositionVelocityConsistencyCheck

  *Class implementation for the position velocity check.*

### Namespaces

- pnt_integrity

  *Namespace for all pnt_integrity applications.*

### Variables

- const std::string pnt_integrity::INTEGRITY_PVC_DIAGNOSTICS = "INTEGRITY_PVC_DIAGNOSTICS"

  *String ID for the position-velocity consistent check diagnostics data.*
- const std::string pnt_integrity::INTEGRITY_PVC_DIAG_PB = "INTEGRITY_PVC_DIAG_PB"

  *String ID for PVC diagnostic key for the "percent bad" variable.*
- const std::string pnt_integrity::INTEGRITY_PVC_DIAG_ITHRESH = "INTEGRITY_PVC_DIAG_ITHRESH"

  *String ID for the PVC diagnostic key for the inconsistent threshold.*
- const std::string pnt_integrity::INTEGRITY_PVC_DIAG_UTHRESH = "INTEGRITY_PVC_DIAG_UTHRESH"

  *String ID for the PVC diagnostic key for the unassured threshold.*
- const std::string pnt_integrity::INTEGRITY_PVC_DIAG_ERR_VAL = "INTEGRITY_PVC_DIAG_ERR_VAL"

  *String ID for the PVC diagnostic key for error values.*
- const std::string pnt_integrity::INTEGRITY_PVC_DIAG_ERR_THRESH

  *String ID for the PVC diagnostic key for error thresh values.*

### 9.15.1 Detailed Description

AssurancCheck class defined for the position velocity consistency check.

**Author**

Josh Clanton `josh.clanton@is4s.com`
John David Sprunger `jss0027@tigermail.auburn.edu`

**Date**

September 18, 2019

## 9.16 include/pnt_integrity/RangePositionCheck.hpp File Reference

AssurancCheck class defined for the range / position check.

```
#include "pnt_integrity/AssuranceCheck.hpp"
```

## Classes

- struct pnt_integrity::RngPosCheckNodeDiagnostic

  *Structure for check diagnostics.*
- class pnt_integrity::RangePositionCheck

  *Class implementation for the range / position check.*

## Namespaces

- pnt_integrity

  *Namespace for all pnt_integrity applications.*

## Typedefs

- using pnt_integrity::RngPosCheckDiagnostics = std::map< std::string, RngPosCheckNodeDiagnostic >

  *Defined type for check diagnostics.*

## Variables

- const std::string pnt_integrity::INTEGRITY_RNG_POS_DIAGNOSTICS

  *String ID for the range-position check diagnostic data.*
- const std::string pnt_integrity::INTEGRITY_RNG_POS_DIAG_MAX_CALC

  *String ID for the range-position check max calculated range.*
- const std::string pnt_integrity::INTEGRITY_RNG_POS_DIAG_MIN_CALC

  *String ID for the range-position check min calculated range.*
- const std::string pnt_integrity::INTEGRITY_RNG_POS_DIAG_MAX_MEAS

  *String ID for the range-position check max measured range.*
- const std::string pnt_integrity::INTEGRITY_RNG_POS_DIAG_MIN_MEAS

  *String ID for the range-position check min measured range.*

### 9.16.1 Detailed Description

AssurancCheck class defined for the range / position check.

**Author**

Josh Clanton `josh.clanton@is4s.com`

**Date**

June 11, 2019

## 9.17 include/pnt_integrity/RepositoryEntry.hpp File Reference

Defines the RepositoryEntry class in pnt_integrity.

```
#include <cmath>
#include <functional>
#include <map>
#include <string>
#include <vector>
#include "logutils/logutils.hpp"
#include "pnt_integrity/IntegrityData.hpp"
```

### Classes

- class pnt_integrity::RepositoryEntry

  *Class definition for an entry into the repository.*

### Namespaces

- pnt_integrity

  *Namespace for all pnt_integrity applications.*

### Enumerations

- enum pnt_integrity::DataLocaleType { **Local** = 0, **Remote** = 1 }

  *Defines the possible observable types.*

### 9.17.1 Detailed Description

Defines the RepositoryEntry class in pnt_integrity.

**Author**

Josh Clanton `josh.clanton@is4s.com`

**Date**

May 28, 2019

## 9.18 include/pnt_integrity/StaticPositionCheck.hpp File Reference

AssurancCheck class defined for the static position check.

```
#include <cstring>
#include "pnt_integrity/AssuranceCheck.hpp"
```

## Classes

- struct pnt_integrity::StaticPosCheckDiagnostics

  *Structure used to publish diagnostic data.*
- class pnt_integrity::StaticPositionCheck

  *Class implementation for the static-position check.*

## Namespaces

- pnt_integrity

  *Namespace for all pnt_integrity applications.*

## Variables

- const std::string pnt_integrity::INTEGRITY_STATIC_POS_DIAGNOSTICS

  *String ID for the static position check diagnostic data.*
- const std::string pnt_integrity::INTEGRITY_STAIC_POS_DIAG_POS_LAT

  *String ID for the static position check survey latitude.*
- const std::string pnt_integrity::INTEGRITY_STAIC_POS_DIAG_POS_LON

  *String ID for the static position check survey longitdue.*
- const std::string pnt_integrity::INTEGRITY_STAIC_POS_DIAG_POS_ALT

  *String ID for the static position check survey altitude.*
- const std::string pnt_integrity::INTEGRITY_STAIC_POS_DIAG_POS_CHNG_THRESH

  *String ID for the static position check change threshold.*
- const std::string pnt_integrity::INTEGRITY_STAIC_POS_DIAG_PERCENT_OVER

  *String ID for the static position check percentage threshold.*
- const std::string pnt_integrity::INTEGRITY_STAIC_POS_DIAG_ITHRESH

  *String ID for the static position check survey inconsistent thresh.*
- const std::string pnt_integrity::INTEGRITY_STAIC_POS_DIAG_UTHRESH

  *String ID for the static position check survey unassured thresh.*

### 9.18.1 Detailed Description

AssurancCheck class defined for the static position check.

#### Author

Josh Clanton `josh.clanton@is4s.com`
John David Sprunger `jss0027@tigermail.auburn.edu`

#### Date

September 3, 2019

# Index