Massless API

Generated by Doxygen 1.8.17

Chapter 1

The Massless Pen API

The API, and this guide in html form can be found at massless.io/Developers.

Massless Pen API Developers Guide

We provide an API for developers to include input from the Massless Pen in their own applications. This takes the form of a C-API in a native DLL built for 64bit Windows. We also include the corresponding header (.h) and library (.lib) files to conveniently link against this API.

This guide will provide information on how to get started using the API, and what exactly it will output.

Basic Usage

To use the API, it is recommended to first register the notification and error callbacks using PenAttachNotification↩ Listener and PenAttachEventListener. This will enable you to receive any log messages and error notifications that may occur during PenStart.

To start tracking you would need to call PenStart with your assigned integration key, which would start the camera and tracking algorithms running, and create the data structures to hold all the settings. If you want the API to return the pose with length units other than the default mm, call the PenSetUnits function.

You need to get the tracker's pose relative to the attached VR HMD coordinate system. This is found by calling Pen↩ GetTrackerPose specifying the correct type enum value that corresponds to the type of tracking reference device that you have attached to the Massless Tracker. This function will then provide you with the offsets in both linear and rotation.

We currently support a number of devices as tracking references. This includes Vive Lighthouse Base Stations, both version 1 and version 2; Oculus Sensor, for the Oculus Rift CV1; Vive Tracker, designed for tracking 3rd party devices, this is the recommended method for the Vive Lighthouse based systems; and the Oculus Touch controllers (version 2) that come with the Rift S and Quest.

Then you would register the pose callback by calling PenAttachFullPoseListener with whichever function you have defined to be your full pose callback. If you want information about the pen's capacitive sensor, you would register the callback PenAttachFullStateListener with whichever function you have defined to be your full state callback. At the moment we have disabled the surface sensing capabilities of the pen, so please ignore this in the returned data.

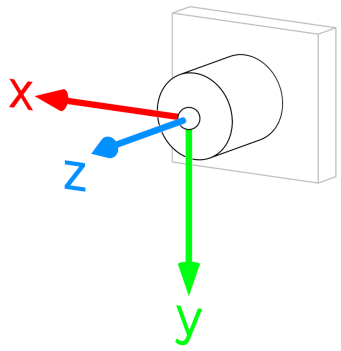
To provide haptic feedback you can call PenSimpleBuzz at any time to vibrate the pen's haptic feedback motor for the specified duration (in ms).

When you have finished you would call PenStop for the program to release its resources and close. The PenStop also deregisters your notification callback in preparation for closing down the whole API. However, if you want to start it up again without unloading the DLL it is fine to just call PenAttachNotificationListener again and then PenStart to start the tracking up again.

Coordinate System

Massless Trackers

The Massless API follows the computer vision convention for coordinate systems, having the right-handed coordinate system defined by the camera.



This figure shows the convention that we follow for the cameras' coordinate systems.

Origin

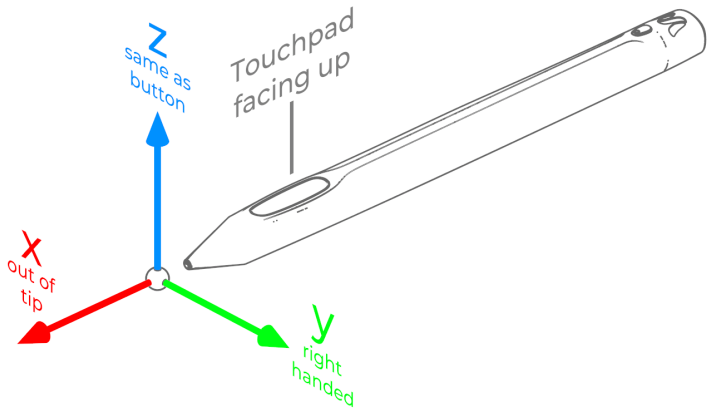
The origin of the Massless coordinate system is that of our tracker. Typically this tracker must be attached to one of the headset system's trackers. We then provide the transformation between the coordinate system of our tracker and that of an Oculus Tracker, or a Vive Lighthouse that is attached to our tracker via one of the brackets we supply. You must decide which headset system you are using and call the function to get the Massless Tracker's pose offset from this system.

Offset Modifications For the Rift S & Quest

There are two sets of offsets for the Rift S & Quest controllers. This is because the offsets we give, compare the Rift S Touch controller's coordinate system with the Massless Tracker's coordinate system. By default, in Unity (using the XRInput system) the coordinate system lines up with the Touch controller in one way, and in OpenVR, the Touch controller's coordinate system lines up with the physical controller in another way (approximately 45° different) so you need to work out which coordinate system of the Touch controller you have, and therefore which set of offsets to use from the Massless API.

Massless Pen

The Massless Pen has a coordinate system itself. Its pose is represented as a transformation between this coordinate system and the Massless Tracker's coordinate system. The origin of the Massless Pen's coordinate system is the tip, with the x-axis pointing away from the tip, forwards, in the direction the pen is pointing, and the z-axis out of the same side as the power button. The y-axis is then fully defined by those two and the fact that it is a right-handed system.



This figure shows the coordinate system associated with the Massless Pen.

Transformations

We provide the "pose" as a transformation, comprised of a displacement vector and a quaternion. Both of these are actually object-transformations rather than straight descriptions of position and orientation. This is illustrated with an example.

Take a model of the Massless Pen including a marker on it. Apply the pen pose transformation according to the following equation, to find the position of the marker in the Massless Tracker's coordinate system:

$$\underline{x}_t = P_{tp}\underline{x}_p \tag{1.1}$$

where $\underline{x}_p$ is the position of the marker in the Massless Pen's coordinate system and the pose transformation returned by the API is

$$P_{tp}\underline{x} = R_{tp}\underline{x} + T_{tp} \tag{1.2}$$

the rotation is actually provided as a quaternion, so the rotation operator here is not a matrix but represents the application of the unit quaternion's rotation to the vector.

Use of the Coordinate systems in Unity

Unity uses a left-handed coordinate system, we take Y as the axis being flipped, which brings our cameras into line with how their camera coordinate systems function.

On bringing our pose into Unity, we recommend the following transformations.

$$y_{unity} = -y_{Massless} \tag{1.3}$$

then divide the whole thing by 1000 to go from mm to m, for the translation part of the pose. The quaternion needs the following,

$$qx_{unity} = -qz_{Massless} \tag{1.4}$$
$$qz_{unity} = -qx_{Massless} \tag{1.5}$$

This will then be in proper Unity units, so we can just set the resulting quaternion and position to the transform of a game object.

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

Chapter 4

Class Index

4.1 Class List

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

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

Chapter 6

Namespace Documentation

6.1 Massless Namespace Reference

Namespaces

- Events

6.2 Massless::Events Namespace Reference

Classes

- struct BaseEvent
- struct ErrorEvent
- struct PenBatteryEvent
- struct PenTappedEvent
- struct TouchPadHeldEvent
- struct TouchPadMultiTapNewEvent
- struct TouchPadMultiTapTotalEvent
- struct TouchPadPressedEvent
- struct TouchPadReleasedEvent
- struct TouchPadSwipeEvent

Enumerations

- enum EventType : uint16_t {
  Error = 0, PenBattery = 1, CameraOK = 2, TedOK = 3,
  PenOK = 4, PenTapped = 5, PenConnected = 6, PenDisconnected = 7,
  TouchPadPressed = 8, TouchPadReleased = 9, TouchPadHeld = 10, TouchPadMultiTapNew = 11,
  TouchPadMultiTapTotal = 12, TouchPadSwipe = 13 }
- enum PenTapFrom : uint8_t {
  XPlus = 0, XMinus = 1, YPlus = 2, YMinus = 3,
  ZPlus = 4, ZMinus = 5 }

Variables

- static const unordered_map< uint16_t, int32_t > EventLength

6.2.1 Enumeration Type Documentation

6.2.1.1 EventType

```
enum Massless::Events::EventType :  uint16_t
```

Enum to cover the different event types that we send from the API

This event type will define what payload is attached and what else needs to be read. Most are separate events, but there are a whole group of TouchPad events depending on what has ahappened with the TouchPad.

Enumerator

Error	The system has encountered an error, the payload will include details.
PenBattery	The pen has a battery event, which one is in the payload.
CameraOK	The system has reported that the camera has started correctly.
TedOK	The system has reported that the tracker extension (serial comms and IMU) has started correctly.
PenOK	The system has reported that the pen has started and connected correctly.
PenTapped	The pen has been physically tapped, measured by the IMU, (please do not use, it is not reliable).
PenConnected	The pen has been connected to the tracker over BLE.
PenDisconnected	The pen has been disconnected from the tracker over BLE.
TouchPadPressed	The touchpad has been pressed, details in the payload.
TouchPadReleased	The touchpad has been released, details in the payload.
TouchPadHeld	A touchpad held gesture has been recognised, details in the payload.
TouchPadMultiTapNew	A new tap has been recognised close enough to be considered a new tap in a multitap gesture, details in the payload.
TouchPadMultiTapTotal	A new multitap has been recognised with a final total number of taps available, details in the payload.
TouchPadSwipe	A touchpad swipe gesture has been recognised, details in the payload.

6.2.1.2 PenTapFrom

```
enum Massless::Events::PenTapFrom :  uint8_t
```

Enum defining the pen tap directions, not recommended to use because it is unreliable.

Enumerator

XPlus	

Enumerator

XMinus	
YPlus	
YMinus	
ZPlus	
ZMinus	

6.2.2 Variable Documentation

6.2.2.1 EventLength

```
const unordered_map<uint16_t, int32_t> Massless::Events::EventLength  [static]
```

Initial value:

```
= {
            {EventType::Error, 4},
            {EventType::PenBattery, 3},
            {EventType::CameraOK, 0},
            {EventType::TedOK, 0},
            {EventType::PenOK, 0},
            {EventType::PenTapped, 2},
            {EventType::PenConnected, 0},
            {EventType::PenDisconnected, 0},
            {EventType::TouchPadPressed, 1},
            {EventType::TouchPadReleased, 6},
            {EventType::TouchPadHeld, 4},
            {EventType::TouchPadMultiTapNew, 5},
            {EventType::TouchPadMultiTapTotal, 5},
            {EventType::TouchPadSwipe, 4}
        }
```

Map to tell you the payload size sent with each event

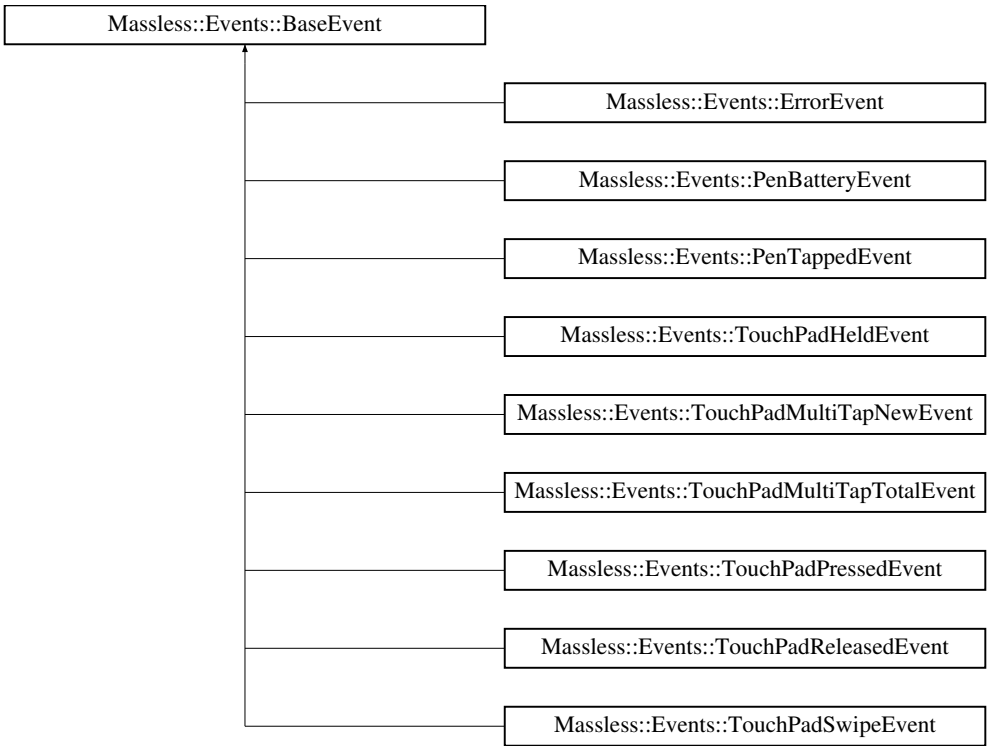
Chapter 7

Class Documentation

7.1 Massless::Events::BaseEvent Struct Reference

`#include <MasslessCallbackEvents.h>`

Inheritance diagram for Massless::Events::BaseEvent:



7.1.1 Detailed Description

Nothing in the base event, but stored so we have the base class to inherit from

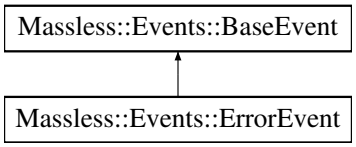
The documentation for this struct was generated from the following file:

- MasslessCallbackEvents.h

7.2 Massless::Events::ErrorEvent Struct Reference

`#include <MasslessCallbackEvents.h>`

Inheritance diagram for Massless::Events::ErrorEvent:



Public Attributes

- uint32_t ErrorNumber

  Error code as uint32, check MasslessErrors.h for details of each code.

7.2.1 Detailed Description

Structure defining the payload of the Error Event.

7.2.2 Member Data Documentation

7.2.2.1 ErrorNumber

`uint32_t Massless::Events::ErrorEvent::ErrorNumber`

Error code as uint32, check MasslessErrors.h for details of each code.

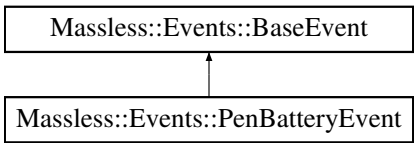
The documentation for this struct was generated from the following file:

- MasslessCallbackEvents.h

7.3 Massless::Events::PenBatteryEvent Struct Reference

`#include <MasslessCallbackEvents.h>`

Inheritance diagram for Massless::Events::PenBatteryEvent:



Public Attributes

- bool Low

  Boolean to say if the battery is low.
- bool Critical

  Boolean to say if the battery is critically low.
- bool Charging

  Boolean to say if the battery is charging.

7.3.1 Detailed Description

Structure defining the payload of the PenBattery Event.

7.3.2 Member Data Documentation

7.3.2.1 Charging

```
bool Massless::Events::PenBatteryEvent::Charging
```

Boolean to say if the battery is charging.

7.3.2.2 Critical

```
bool Massless::Events::PenBatteryEvent::Critical
```

Boolean to say if the battery is critically low.

7.3.2.3 Low

```
bool Massless::Events::PenBatteryEvent::Low
```

Boolean to say if the battery is low.

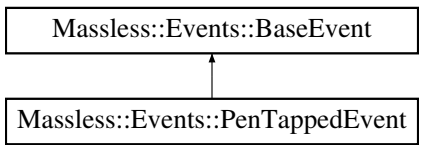
The documentation for this struct was generated from the following file:

- MasslessCallbackEvents.h

7.4 Massless::Events::PenTappedEvent Struct Reference

`#include <MasslessCallbackEvents.h>`

Inheritance diagram for Massless::Events::PenTappedEvent:



Public Attributes

- bool DoubleTap

  Is the tap a double-tap?
- PenTapFrom TapDirection

  What direction the tap came from.

7.4.1 Detailed Description

Structure defining the payload of the PenTapped Event. Not recommended for use.

7.4.2 Member Data Documentation

7.4.2.1 DoubleTap

`bool Massless::Events::PenTappedEvent::DoubleTap`

Is the tap a double-tap?

7.4.2.2 TapDirection

`PenTapFrom Massless::Events::PenTappedEvent::TapDirection`

What direction the tap came from.

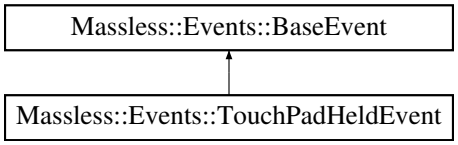
The documentation for this struct was generated from the following file:

- MasslessCallbackEvents.h

7.5 Massless::Events::TouchPadHeldEvent Struct Reference

`#include <MasslessCallbackEvents.h>`

Inheritance diagram for Massless::Events::TouchPadHeldEvent:



Public Attributes

- float PositionAverage

  The average position of the finger during the time between touched and the system recognising a hold gesture.

7.5.1 Detailed Description

Structure defining the payload of the TouchPadHeld Event.

The touchpad held event is fired as soon as the gesture recognition system detects a "hold" gesture, this is after a certain duration that the touchpad has been touched while moving little enough to be considered stationary.

7.5.2 Member Data Documentation

7.5.2.1 PositionAverage

`float Massless::Events::TouchPadHeldEvent::PositionAverage`

The average position of the finger during the time between touched and the system recognising a hold gesture.

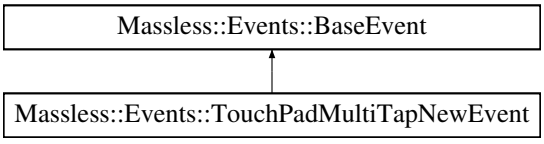
The documentation for this struct was generated from the following file:

- MasslessCallbackEvents.h

7.6 Massless::Events::TouchPadMultiTapNewEvent Struct Reference

`#include <MasslessCallbackEvents.h>`

Inheritance diagram for Massless::Events::TouchPadMultiTapNewEvent:



Public Attributes

- float PositionAverage

  The average position of the finger for each of the press events in the multitap.
- uint8_t CountCurrent

  The count of the current press, and therefore the running total of how many taps so far.

7.6.1 Detailed Description

Structure defining the payload of the TouchPadMultiTapNew Event.

Each new tap that is detected as part of a multitap is announced with this event.

7.6.2 Member Data Documentation

7.6.2.1 CountCurrent

```
uint8_t Massless::Events::TouchPadMultiTapNewEvent::CountCurrent
```

The count of the current press, and therefore the running total of how many taps so far.

7.6.2.2 PositionAverage

```
float Massless::Events::TouchPadMultiTapNewEvent::PositionAverage
```

The average position of the finger for each of the press events in the multitap.

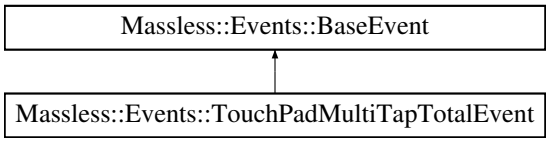
The documentation for this struct was generated from the following file:

- MasslessCallbackEvents.h

7.7 Massless::Events::TouchPadMultiTapTotalEvent Struct Reference

```
#include <MasslessCallbackEvents.h>
```

Inheritance diagram for Massless::Events::TouchPadMultiTapTotalEvent:



Public Attributes

- float PositionAverage

  The average position of all the taps in the multitap gesture.
- uint8_t CountTotal

  The final count of number of taps included in the multitap gesture.

7.7.1 Detailed Description

Structure defining the payload of the TouchPadMultiTapTotal Event.

When the multitap detection times out and decides that the multiple tap gesture has finished it will fire this event to let the client know the final tap count. There will be a slight delay after the final tap to ensure no further taps could be recorded and counted in the same gesture.

7.7.2 Member Data Documentation

7.7.2.1 CountTotal

```
uint8_t Massless::Events::TouchPadMultiTapTotalEvent::CountTotal
```

The final count of number of taps included in the multitap gesture.

7.7.2.2 PositionAverage

```
float Massless::Events::TouchPadMultiTapTotalEvent::PositionAverage
```

The average position of all the taps in the multitap gesture.

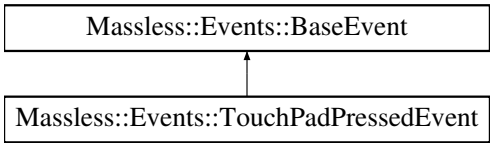
The documentation for this struct was generated from the following file:

- MasslessCallbackEvents.h

7.8 Massless::Events::TouchPadPressedEvent Struct Reference

```
#include <MasslessCallbackEvents.h>
```

Inheritance diagram for Massless::Events::TouchPadPressedEvent:



Public Attributes

- uint8_t PositionPressed

  The position where the touchpad is touched.

7.8.1 Detailed Description

Structure defining the payload of the TouchPadPressed Event.

The touchpad pressed event is fired immediately as it is touched. This is effectievly raw, before the gesture recognition system gets hold of the data and starts recognising gestures.

7.8.2 Member Data Documentation

7.8.2.1 PositionPressed

```
uint8_t Massless::Events::TouchPadPressedEvent::PositionPressed
```

The position where the touchpad is touched.

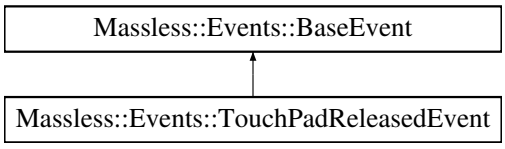
The documentation for this struct was generated from the following file:

- MasslessCallbackEvents.h

7.9 Massless::Events::TouchPadReleasedEvent Struct Reference

```
#include <MasslessCallbackEvents.h>
```

Inheritance diagram for Massless::Events::TouchPadReleasedEvent:



Public Attributes

- float PositionAverage

  The average position of the finger throughout the entire gesture.
- uint8_t DurationPressed

  The duration of the hold.
- uint8_t PositionReleased

  The instantateous position when it was released.

7.9.1 Detailed Description

Structure defining the payload of the TouchPadReleased Event.

The touchpad pressed event is fired immediately as it is released. This is effectively raw data before the gesture recognition system gets hold of the data and starts recognising gestures. However, a lot of gestures can be ended with the released event, so this can often be used to "end" the previous gesture. Eg. If you have previously had "pressed", then "held" then "released", this means that the touchpad held gesture has finished.

7.9.2 Member Data Documentation

7.9.2.1 DurationPressed

```
uint8_t Massless::Events::TouchPadReleasedEvent::DurationPressed
```

The duration of the hold.

7.9.2.2 PositionAverage

```
float Massless::Events::TouchPadReleasedEvent::PositionAverage
```

The average position of the finger throughout the entire gesture.

7.9.2.3 PositionReleased

```
uint8_t Massless::Events::TouchPadReleasedEvent::PositionReleased
```

The instantateous position when it was released.

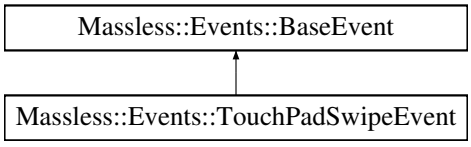
The documentation for this struct was generated from the following file:

- MasslessCallbackEvents.h

7.10 Massless::Events::TouchPadSwipeEvent Struct Reference

```
#include <MasslessCallbackEvents.h>
```

Inheritance diagram for Massless::Events::TouchPadSwipeEvent:



Public Attributes

- float Velocity

  The velocity of the swipe, positive being swiping towards the tip.

7.10.1 Detailed Description

Structure defining the payload of the TouchPadSwipe Event.

This is fired when the gesture recognition detects that the touchpad is being held but the contact point moved significantly and was then released again. It is a touch-move-release gesture.

7.10.2 Member Data Documentation

7.10.2.1 Velocity

```
float Massless::Events::TouchPadSwipeEvent::Velocity
```

The velocity of the swipe, positive being swiping towards the tip.

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

- MasslessCallbackEvents.h

Chapter 8

File Documentation

8.1 DocumentationSummary.h File Reference

8.2 MasslessCallbackEvents.h File Reference

```
#include <cstdint>
#include <unordered_map>
```

Classes

- struct Massless::Events::BaseEvent
- struct Massless::Events::ErrorEvent
- struct Massless::Events::PenBatteryEvent
- struct Massless::Events::PenTappedEvent
- struct Massless::Events::TouchPadPressedEvent
- struct Massless::Events::TouchPadReleasedEvent
- struct Massless::Events::TouchPadHeldEvent
- struct Massless::Events::TouchPadMultiTapNewEvent
- struct Massless::Events::TouchPadMultiTapTotalEvent
- struct Massless::Events::TouchPadSwipeEvent

Namespaces

- Massless
- Massless::Events

Enumerations

- enum Massless::Events::EventType : uint16_t {
  Massless::Events::Error = 0, Massless::Events::PenBattery = 1, Massless::Events::CameraOK = 2,
  Massless::Events::TedOK = 3,
  Massless::Events::PenOK = 4, Massless::Events::PenTapped = 5, Massless::Events::PenConnected = 6,
  Massless::Events::PenDisconnected = 7,
  Massless::Events::TouchPadPressed = 8, Massless::Events::TouchPadReleased = 9, Massless::Events::TouchPadHeld = 10, Massless::Events::TouchPadMultiTapNew = 11,
  Massless::Events::TouchPadMultiTapTotal = 12, Massless::Events::TouchPadSwipe = 13 }
- enum Massless::Events::PenTapFrom : uint8_t {
  Massless::Events::XPlus = 0, Massless::Events::XMinus = 1, Massless::Events::YPlus = 2, Massless::Events::YMinus = 3,
  Massless::Events::ZPlus = 4, Massless::Events::ZMinus = 5 }

Variables

- static const unordered_map< uint16_t, int32_t > Massless::Events::EventLength

8.3 MASSLESSErrors.h File Reference

Macros

- #define M_OK EXIT_SUCCESS

  Return a zero when everything is fine. Defined in terms of stdlib's exit success macro.
- #define ERROR_Unknown_DLL_loading_error 0x1000

  Unknown error occurred during DLL loading.
- #define ERROR_DLL_Failed_Load 0x1001

  Failed to load the DLL, MASSLESSPen.dll.
- #define ERROR_DLL_Function_Not_Found 0x1002

  Failed to find the function #1 in the DLL.
- #define ERROR_UnsupportedThirdPartyTrackerType 0x1003

  Unsupported tracker type for third party tracking.
- #define ERROR_NoTrackerConnectedYet 0x1004

  No tracker is connected yet, cannot define pose.
- #define ERROR_NoTrackerOrientationReadingYet 0x1005

  No measurement from the tracker yet, orientation is not valid.
- #define ERROR_Logger_failed_to_open_folder 0x1006

  Logger failed to open the Massless AppData folder, logging disabled.
- #define WARNING_Not_logged_as_did_not_meet_minimum_level 0x1007

  Warning: Logger did not log message as it was lower than the minimum level.
- #define WARNING_Repeated_logger_message_ignored 0x1008

  Warning: Logger has ignored a repeated message.
- #define WARNING_NotSendingNotificationAsNoListenersRegistered 0x1009

  Warning: There are no listeners so we are skipping sending out to notification listeners.
- #define WARNING_NotSendingEventAsNoListenersRegistered 0x100A

  Warning: There are no listeners so we are skipping sending out to event listeners.
- #define ERROR_RequestedDeviceDetailsNotAvailable 0x100B

  Error: The requested device details are not available, perhaps this device has not yet responded over serial.
- #define ERROR_TrackingAPINotStarted 0x100C

  Error: You must call PenStart before this function, as the system needs to be created and initialised.
- #define WARNING_PenStopCalledPenAlreadyStopped 0x100D

  Warning: PenStop skipping as pen not started or already stopped.
- #define ERROR_SystemConfigurationFileFailedCreationFromDefaults 0x100E

  Error: Failed to create the default system configuration file from defaults, check file/folder permissions.
- #define ERROR_SystemConfigurationFileFailedOpenForRead 0x100F

  Error: Failed to open the system configuration file for reading, check file/folder permissions.
- #define ERROR_SystemConfigurationFileFailedOpenForWrite 0x1010

  Error: Failed to open the system configuration file for writing, check file/folder permissions.
- #define ERROR_SystemConfigurationPropertyNotRecognised 0x1011

  Error: Requested system configuration property not recognised.
- #define ERROR_SystemConfigurationPropertyNotFoundInFile 0x1012

  Error: Requested system configuration property not found in configuration file.
- #define ERROR_Unknown_Camera_Setup_error 0x2000

Unknown error occurred during camera setup.

- #define ERROR_Pen_System_Already_Started 0x2001

  Failed to start the pen system, pen already loaded.
- #define ERROR_Camera_Setup_Failed 0x2002

  Unknown failure during camera setup.
- #define ERROR_CouldNotStartPenAlreadyStarted 0x2003

  Could not initialise pen. Pen object already exists.
- #define ERROR_No_Tracker_Found 0x2004

  No Tara camera found, cannot start tracking, aborting.
- #define ERROR_CameraIsNotOpenAnyMore 0x2005

  ERROR: The camera is no longer open.
- #define WARNING_UsingDefaultCameraProperties 0x2006

  Warning: Using default camera properties.
- #define ERROR_FailedToReadExtrinsicAndIntrinsicFiles 0x2007

  ERROR: Failed to read the camera intrinsic and extrinsic files.
- #define ERROR_InvalidIntrinsicOrExtrinsicFileLength 0x2008

  ERROR: Invalid intrinsic or extrinsic file length, causing read failure.
- #define ERROR_FailedToInitialiseCameraExtensionUnit 0x2009

  ERROR: Failed to initialise the camera extension unit.
- #define ERROR_Unknown_error_in_tracking 0x3000

  Unknown error occurred during tracking.
- #define ERROR_Unknown_error_location_tracking 0x3001

  Unknown error occurred with the location tracking.
- #define ERROR_Unknown_error_orientation_tracking 0x3002

  Unknown error occurred with the orientation tracking.
- #define ERROR_PoseUpdate_Callbacks_Long_duration 0x3003

  Error: PoseUpdate callbacks took more than half of the loop timing. This may lead to a loss of tracking precision.
- #define ERROR_Cannot_recieve_IMU_data_fusion_thread_not_running 0x3004

  Error: Cannot recieve IMU data when the fusion thread is not running.
- #define ERROR_Cannot_recieve_Optical_data_fusion_thread_not_running 0x3005

  Error: Cannot recieve optical data when the fusion thread is not running.
- #define WARNING_KalmanFilter_skipping_predict_at_dt_0 0x3006

  Warning: Kalman Filter cannot perform predict step at the same timestamp as the last one. Skipping the predict step.
- #define ERROR_Kalman_Filter_reject_invalid_measurement 0x3007

  ERROR: Kalman Filter cannot update using an invalid measurement, with a negative timestamp.
- #define ERROR_Kalman_Filter_cannot_predict_until_initialised 0x3008

  ERROR: Kalman Filter cannot predict the state until it has been fully initialised with a correct state.
- #define ERROR_Kalman_Filter_cannot_correct_until_initialised 0x3009

  ERROR: Kalman Filter cannot apply a measured state (correct step), until it has been initialised.
- #define ERROR_OrderOfSGFilterInvalid 0x300A

  ERROR: Attempted to set an invalid Savitsky-Golay filter order. Only 5,7 and 9 are allowed.
- #define WARNING_Kalman_Filter_rejected_measurement_for_being_an_outlier 0x300B

  WARNING: Kalman filter has rejected the supplied measurement for being an outlier.
- #define ERROR_TrackingAlgorithmFoundTooManyPotentialMarkers 0x300C

  ERROR: Tracking algorithm found too many potential markers in the image. Please check your camera's shutter speed and gain.
- #define WARNING_Kalman_Filter_Initialised_By_Correct_Step 0x3010

  Warning: Kalman Filter was initialised by providing a correct step measurement.
- #define ERROR_Kalman_Filter_cannot_predict_into_past 0x3011

  ERROR: Kalman Filter cannot 'predict' to a point in the past.
- #define ERROR_Kalman_Filter_cannot_predict_long_future 0x3012

ERROR: Kalman Filter cannot predict too far into the future.

- #define ERROR_Unknown_Buzzer_error 0x4000

  Unknown error occurred with the buzzer.
- #define ERROR_Buzzer_not_yet_implemented 0x4001

  Buzzer interface is not implemented yet, please do not use.
- #define ERROR_Unknown_Serial_error 0x5000

  Unknown error occurred with the serial communication.
- #define ERROR_Handle_unattached_COM_Port_Unavailable 0x5001

  ERROR: Handle was not attached. Reason: s not available.
- #define ERROR_Failed_to_get_serial_parameters 0x5002

  Error: Failed to get the current serial parameters.
- #define ERROR_Could_not_set_serial_parameters 0x5003

  Error: Failed to set the serial parameters.
- #define ERROR_Serial_port_already_closed 0x5004

  Error: Could not close, serial port not open.
- #define ERROR_Serial_port_write_error 0x5005

  Error: Could not write to serial port, error number returned lu.
- #define ERROR_Serial_command_sending_busy 0x5006

  Could not send command, as previous command still in progress.
- #define ERROR_Serial_Unknown_command_sent 0x5007

  Error: Command not recognised, unable to send via serial.
- #define WARNING_Radio_Comms_Inactive 0x5008

  Warning: Radio comms are inactive, this action will not be performed.
- #define WARNING_Serial_Interface_address_already_set 0x5009

  Warning: This serial interface has already had an address set, being overwritten.
- #define WARNING_UnknownSerialCommandRecieved 0x500A

  Warning: Unknown serial command received.
- #define ERROR_SerialPacketRecievedWithIncorrectLength 0x500B

  Error: Serial packet receieved with incorrect length advertised.
- #define WARNING_SerialMessageReceivedWithIncorrectCRC 0x500C

  Warning: Serial packet received with incorrect CRC.
- #define ERROR_FailedToOpenCOMPortAccessDenied 0x500D

  Error: Access to COM port denied.
- #define WARNING_SerialSendBufferFullRemovingOldestCommand 0x500E

  Warning: Serial send buffer is full, removing oldest command of type s.
- #define WARNING_SerialPortHandshakeTimeout 0x500F

  Warning: Serial port handshake has timed out, device not a Massless device, or not correctly working or connected.
- #define ERROR_No_pen_comms_known 0x5010

  Error: The communication with the Pen is unknown.
- #define ERROR_Failed_to_find_Massless_devices_on_serial 0x5011

  Error: During serial start up, failed to find any Massless Devices.
- #define ERROR_AttemptingToSendCommandToPenWhileNotAttached 0x5012

  Error: Attempting to send command to the pen via a port that doesn't currently have the pen attached.
- #define ERROR_PenDataReportedByDeviceThatIsNotPen 0x6000

  Device that is not a pen trying to report pen sensor data.
- #define WARNING_PenBatteryLow 0x6001

  Warning: Pen battery is getting low, please charge it.
- #define WARNING_ThisPenExperiencedAtLeastOneHardFault 0x6002
- #define ERROR_PenBatteryCritical 0x6003

  Error: Pen battery is critical, pen shutdown is imminent.
- #define WARNING_PenSurfaceSensorHasBackground 0x6004

Warning: The pen's surface sensor has too much background.

- #define ERROR_FirmwareTooLow 0x6005

  Error: The firmware is too low on one of your devices, please update.
- #define ERROR_FirmwareTooHigh 0x6006

  Error: Your firmware is too high on one of your devices, please downgrade, or upgrade this tracking API.
- #define ERROR_FirmwareCheckingDeviceTypeInvalid 0x6007

  Error: Unable to check the firmware requested, because the device type was incorrect.
- #define ERROR_UnknownMetricTrackingError 0x7000

  Unknown error occurred with the metrics tracking.
- #define WARNING_MetricsTrackingFailedOpeningCacheFile 0x7001

  Warning: Could not open the cache, sending unknowns until we receive values from the pen.
- #define WARNING_MetricsTrackingFailedOpeningRetryFile 0x7002

  Warning: Could not open the retry file.
- #define WARNING_NoMetricsToRetrySending 0x7003

  Warning: No metrics to retry sending, none in the failed log.
- #define ERROR_CouldNotPutRemainingToRetryBackInFailed 0x7004

  Error: Interrupted while retrying, but could not replace the retry pending messages back in the failed log.
- #define ERROR_Unknown 0xF000

  Unknown error occurred.
- #define ERROR_ImageProcessingThreadNotRunning 0xF001

  Failed to stop the pen system correctly, as the image processing thread was not running.
- #define ERROR_FailedToCreateMasslessFolderInAppData 0xF002

  Failed to create MASSLESS folder in the user's AppData directory.
- #define ERROR_ReturnParameterAddressInvalid 0xF003

  Returning a variable as a pointer to the output parameter, but pointer address is invalid.
- #define WARNING_NotImplementedYet 0xF004

  Warning: This function has not been implemented yet.
- #define ERROR_Event_Sent_Without_Payload 0xF005

  Error: An event was attempted to be sent without a payload, ignoring event:
- #define ERROR_Failed_to_get_username 0xF006

  Failed to retrieve the username of the current user.
- #define ERROR_FailedToCreateVideoFolder 0xF007

  Failed to create video folder in the user's Documents directory.
- #define ERROR_FailedToStoreVideoData 0xF008

  Failed to store the video data, intrinsics.yml extrinsics.yml or PenName.Pen.

8.3.1 Macro Definition Documentation

8.3.1.1 ERROR_AttemptingToSendCommandToPenWhileNotAttached

```
#define ERROR_AttemptingToSendCommandToPenWhileNotAttached 0x5012
```

Error: Attempting to send command to the pen via a port that doesn't currently have the pen attached.

8.3.1.2 ERROR_Buzzer_not_yet_implemented

```
#define ERROR_Buzzer_not_yet_implemented 0x4001
```

Buzzer interface is not implemented yet, please do not use.

8.3.1.3 ERROR_Camera_Setup_Failed

```
#define ERROR_Camera_Setup_Failed 0x2002
```

Unknown failure during camera setup.

8.3.1.4 ERROR_CameraIsNotOpenAnyMore

```
#define ERROR_CameraIsNotOpenAnyMore 0x2005
```

ERROR: The camera is no longer open.

8.3.1.5 ERROR_Cannot_recieve_IMU_data_fusion_thread_not_running

```
#define ERROR_Cannot_recieve_IMU_data_fusion_thread_not_running 0x3004
```

Error: Cannot recieve IMU data when the fusion thread is not running.

8.3.1.6 ERROR_Cannot_recieve_Optical_data_fusion_thread_not_running

```
#define ERROR_Cannot_recieve_Optical_data_fusion_thread_not_running 0x3005
```

Error: Cannot recieve optical data when the fusion thread is not running.

8.3.1.7 ERROR_Could_not_set_serial_parameters

```
#define ERROR_Could_not_set_serial_parameters 0x5003
```

Error: Failed to set the serial parameters.

8.3.1.8 ERROR_CouldNotPutRemainingToRetryBackInFailed

```
#define ERROR_CouldNotPutRemainingToRetryBackInFailed 0x7004
```

Error: Interrupted while retrying, but could not replace the retry pending messages back in the failed log.

8.3.1.9 ERROR_CouldNotStartPenAlreadyStarted

```
#define ERROR_CouldNotStartPenAlreadyStarted 0x2003
```

Could not initialise pen. Pen object already exists.

8.3.1.10 ERROR_DLL_Failed_Load

```
#define ERROR_DLL_Failed_Load 0x1001
```

Failed to load the DLL, MASSLESSPen.dll.

8.3.1.11 ERROR_DLL_Function_Not_Found

```
#define ERROR_DLL_Function_Not_Found 0x1002
```

Failed to find the function #1 in the DLL.

8.3.1.12 ERROR_Event_Sent_Without_Payload

```
#define ERROR_Event_Sent_Without_Payload 0xF005
```

Error: An event was attempted to be sent without a payload, ignoring event:

8.3.1.13 ERROR_Failed_to_find_Massless_devices_on_serial

```
#define ERROR_Failed_to_find_Massless_devices_on_serial 0x5011
```

Error: During serial start up, failed to find any Massless Devices.

8.3.1.14 ERROR_Failed_to_get_serial_parameters

```
#define ERROR_Failed_to_get_serial_parameters 0x5002
```

Error: Failed to get the current serial parameters.

8.3.1.15 ERROR_Failed_to_get_username

```
#define ERROR_Failed_to_get_username 0xF006
```

Failed to retrieve the username of the current user.

8.3.1.16 ERROR_FailedToCreateMasslessFolderInAppData

```
#define ERROR_FailedToCreateMasslessFolderInAppData 0xF002
```

Failed to create MASSLESS folder in the user's AppData directory.

8.3.1.17 ERROR_FailedToCreateVideoFolder

```
#define ERROR_FailedToCreateVideoFolder 0xF007
```

Failed to create video folder in the user's Documents directory.

8.3.1.18 ERROR_FailedToInitialiseCameraExtensionUnit

```
#define ERROR_FailedToInitialiseCameraExtensionUnit 0x2009
```

ERROR: Failed to initialise the camera extension unit.

8.3.1.19 ERROR_FailedToOpenCOMPortAccessDenied

```
#define ERROR_FailedToOpenCOMPortAccessDenied 0x500D
```

Error: Access to COM port denied.

8.3.1.20 ERROR_FailedToReadExtrinsicAndIntrinsicFiles

```
#define ERROR_FailedToReadExtrinsicAndIntrinsicFiles 0x2007
```

ERROR: Failed to read the camera intrinsic and extrinsic files.

8.3.1.21 ERROR_FailedToStoreVideoData

```
#define ERROR_FailedToStoreVideoData 0xF008
```

Failed to store the video data, intrinsics.yml extrinsics.yml or PenName.Pen.

8.3.1.22 ERROR_FirmwareCheckingDeviceTypeInvalid

```
#define ERROR_FirmwareCheckingDeviceTypeInvalid 0x6007
```

Error: Unable to check the firmware requested, because the device type was incorrect.

8.3.1.23 ERROR_FirmwareTooHigh

```
#define ERROR_FirmwareTooHigh 0x6006
```

Error: Your firmware is too high on one of your devices, please downgrade, or upgrade this tracking API.

8.3.1.24 ERROR_FirmwareTooLow

```
#define ERROR_FirmwareTooLow 0x6005
```

Error: The firmware is too low on one of your devices, please update.

8.3.1.25 ERROR_Handle_unattached_COM_Port_Unavailable

```
#define ERROR_Handle_unattached_COM_Port_Unavailable 0x5001
```

ERROR: Handle was not attached. Reason: s not available.

8.3.1.26 ERROR_ImageProcessingThreadNotRunning

```
#define ERROR_ImageProcessingThreadNotRunning 0xF001
```

Failed to stop the pen system correctly, as the image processing thread was not running.

8.3.1.27 ERROR_InvalidIntrinsicOrExtrinsicFileLength

```
#define ERROR_InvalidIntrinsicOrExtrinsicFileLength 0x2008
```

ERROR: Invalid intrinsic or extrinsic file length, causing read failure.

8.3.1.28 ERROR_Kalman_Filter_cannot_correct_until_initialised

```
#define ERROR_Kalman_Filter_cannot_correct_until_initialised 0x3009
```

ERROR: Kalman Filter cannot apply a measured state (correct step), until it has been initialised.

8.3.1.29 ERROR_Kalman_Filter_cannot_predict_into_past

```
#define ERROR_Kalman_Filter_cannot_predict_into_past 0x3011
```

ERROR: Kalman Filter cannot 'predict' to a point in the past.

8.3.1.30 ERROR_Kalman_Filter_cannot_predict_long_future

```
#define ERROR_Kalman_Filter_cannot_predict_long_future 0x3012
```

ERROR: Kalman Filter cannot predict too far into the future.

8.3.1.31 ERROR_Kalman_Filter_cannot_predict_until_initialised

```
#define ERROR_Kalman_Filter_cannot_predict_until_initialised 0x3008
```

ERROR: Kalman Filter cannot predict the state until it has been fully initialised with a correct state.

8.3.1.32 ERROR_Kalman_Filter_reject_invalid_measurement

```
#define ERROR_Kalman_Filter_reject_invalid_measurement 0x3007
```

ERROR: Kalman Filter cannot update using an invalid measurement, with a negative timestamp.

8.3.1.33 ERROR_Logger_failed_to_open_folder

```
#define ERROR_Logger_failed_to_open_folder 0x1006
```

Logger failed to open the Massless AppData folder, logging disabled.

8.3.1.34 ERROR_No_pen_comms_known

```
#define ERROR_No_pen_comms_known 0x5010
```

Error: The communication with the Pen is unknown.

8.3.1.35 ERROR_No_Tracker_Found

```
#define ERROR_No_Tracker_Found 0x2004
```

No Tara camera found, cannot start tracking, aborting.

8.3.1.36 ERROR_NoTrackerConnectedYet

```
#define ERROR_NoTrackerConnectedYet 0x1004
```

No tracker is connected yet, cannot define pose.

8.3.1.37 ERROR_NoTrackerOrientationReadingYet

```
#define ERROR_NoTrackerOrientationReadingYet 0x1005
```

No measurement from the tracker yet, orientation is not valid.

8.3.1.38 ERROR_OrderOfSGFilterInvalid

```
#define ERROR_OrderOfSGFilterInvalid 0x300A
```

ERROR: Attempted to set an invalid Savitsky-Golay filter order. Only 5,7 and 9 are allowed.

8.3.1.39 ERROR_Pen_System_Already_Started

```
#define ERROR_Pen_System_Already_Started 0x2001
```

Failed to start the pen system, pen already loaded.

8.3.1.40 ERROR_PenBatteryCritical

```
#define ERROR_PenBatteryCritical 0x6003
```

Error: Pen battery is critical, pen shutdown is imminent.

8.3.1.41 ERROR_PenDataReportedByDeviceThatIsNotPen

```
#define ERROR_PenDataReportedByDeviceThatIsNotPen 0x6000
```

Device that is not a pen trying to report pen sensor data.

8.3.1.42 ERROR_PoseUpdate_Callbacks_Long_duration

```
#define ERROR_PoseUpdate_Callbacks_Long_duration 0x3003
```

Error: PoseUpdate callbacks took more than half of the loop timing. This may lead to a loss of tracking precision.

8.3.1.43 ERROR_RequestedDeviceDetailsNotAvailable

```
#define ERROR_RequestedDeviceDetailsNotAvailable 0x100B
```

Error: The requested device details are not available, perhaps this device has not yet responded over serial.

8.3.1.44 ERROR_ReturnParameterAddressInvalid

```
#define ERROR_ReturnParameterAddressInvalid 0xF003
```

Returning a variable as a pointer to the output parameter, but pointer address is invalid.

8.3.1.45 ERROR_Serial_command_sending_busy

```
#define ERROR_Serial_command_sending_busy 0x5006
```

Could not send command, as previous command still in progress.

8.3.1.46 ERROR_Serial_port_already_closed

```
#define ERROR_Serial_port_already_closed 0x5004
```

Error: Could not close, serial port not open.

8.3.1.47 ERROR_Serial_port_write_error

```
#define ERROR_Serial_port_write_error 0x5005
```

Error: Could not write to serial port, error number returned lu.

8.3.1.48 ERROR_Serial_Unknown_command_sent

```
#define ERROR_Serial_Unknown_command_sent 0x5007
```

Error: Command not recognised, unable to send via serial.

8.3.1.49 ERROR_SerialPacketRecievedWithIncorrectLength

```
#define ERROR_SerialPacketRecievedWithIncorrectLength 0x500B
```

Error: Serial packet receieved with incorrect length advertised.

8.3.1.50 ERROR_SystemConfigurationFileFailedCreationFromDefaults

```
#define ERROR_SystemConfigurationFileFailedCreationFromDefaults 0x100E
```

Error: Failed to create the default system configuration file from defaults, check file/folder permissions.

8.3.1.51 ERROR_SystemConfigurationFileFailedOpenForRead

```
#define ERROR_SystemConfigurationFileFailedOpenForRead 0x100F
```

Error: Failed to open the system configuration file for reading, check file/folder permissions.

8.3.1.52 ERROR_SystemConfigurationFileFailedOpenForWrite

```
#define ERROR_SystemConfigurationFileFailedOpenForWrite 0x1010
```

Error: Failed to open the system configuration file for writing, check file/folder permissions.

8.3.1.53 ERROR_SystemConfigurationPropertyNotFoundInFile

```
#define ERROR_SystemConfigurationPropertyNotFoundInFile 0x1012
```

Error: Requested system configuration property not found in configuration file.

8.3.1.54 ERROR_SystemConfigurationPropertyNotRecognised

```
#define ERROR_SystemConfigurationPropertyNotRecognised 0x1011
```

Error: Requested system configuration property not recognised.

8.3.1.55 ERROR_TrackingAlgorithmFoundTooManyPotentialMarkers

```
#define ERROR_TrackingAlgorithmFoundTooManyPotentialMarkers 0x300C
```

ERROR: Tracking algorithm found too many potential markers in the image. Please check your camera's shutter speed and gain.

8.3.1.56 ERROR_TrackingAPINotStarted

```
#define ERROR_TrackingAPINotStarted 0x100C
```

Error: You must call PenStart before this function, as the system needs to be created and initialised.

8.3.1.57 ERROR_Unknown

```
#define ERROR_Unknown 0xF000
```

Unknown error occurred.

8.3.1.58 ERROR_Unknown_Buzzer_error

```
#define ERROR_Unknown_Buzzer_error 0x4000
```

Unknown error occurred with the buzzer.

8.3.1.59 ERROR_Unknown_Camera_Setup_error

```
#define ERROR_Unknown_Camera_Setup_error 0x2000
```

Unknown error occurred during camera setup.

8.3.1.60 ERROR_Unknown_DLL_loading_error

```
#define ERROR_Unknown_DLL_loading_error 0x1000
```

Unknown error occurred during DLL loading.

8.3.1.61 ERROR_Unknown_error_in_tracking

```
#define ERROR_Unknown_error_in_tracking 0x3000
```

Unknown error occurred during tracking.

8.3.1.62 ERROR_Unknown_error_location_tracking

```
#define ERROR_Unknown_error_location_tracking 0x3001
```

Unknown error occurred with the location tracking.

8.3.1.63 ERROR_Unknown_error_orientation_tracking

```
#define ERROR_Unknown_error_orientation_tracking 0x3002
```

Unknown error occurred with the orientation tracking.

8.3.1.64 ERROR_Unknown_Serial_error

```
#define ERROR_Unknown_Serial_error 0x5000
```

Unknown error occurred with the serial communication.

8.3.1.65 ERROR_UnknownMetricTrackingError

```
#define ERROR_UnknownMetricTrackingError 0x7000
```

Unknown error occurred with the metrics tracking.

8.3.1.66 ERROR_UnsupportedThirdPartyTrackerType

```
#define ERROR_UnsupportedThirdPartyTrackerType 0x1003
```

Unsupported tracker type for third party tracking.

8.3.1.67 M_OK

```
#define M_OK EXIT_SUCCESS
```

Return a zero when everything is fine. Defined in terms of stdlib's exit success macro.

8.3.1.68 WARNING_Kalman_Filter_Initialised_By_Correct_Step

```
#define WARNING_Kalman_Filter_Initialised_By_Correct_Step 0x3010
```

Warning: Kalman Filter was initialised by providing a correct step measurement.

8.3.1.69 WARNING_Kalman_Filter_rejected_measurement_for_being_an_outlier

```
#define WARNING_Kalman_Filter_rejected_measurement_for_being_an_outlier 0x300B
```

WARNING: Kalman filter has rejected the supplied measurement for being an outlier.

8.3.1.70 WARNING_KalmanFilter_skipping_predict_at_dt_0

```
#define WARNING_KalmanFilter_skipping_predict_at_dt_0 0x3006
```

Warning: Kalman Filter cannot perform predict step at the same timestamp as the last one. Skipping the predict step.

8.3.1.71 WARNING_MetricsTrackingFailedOpeningCacheFile

```
#define WARNING_MetricsTrackingFailedOpeningCacheFile 0x7001
```

Warning: Could not open the cache, sending unknowns until we receive values from the pen.

8.3.1.72 WARNING_MetricsTrackingFailedOpeningRetryFile

```
#define WARNING_MetricsTrackingFailedOpeningRetryFile 0x7002
```

Warning: Could not open the retry file.

8.3.1.73 WARNING_NoMetricsToRetrySending

```
#define WARNING_NoMetricsToRetrySending 0x7003
```

Warning: No metrics to retry sending, none in the failed log.

8.3.1.74 WARNING_Not_logged_as_did_not_meet_minimum_level

```
#define WARNING_Not_logged_as_did_not_meet_minimum_level 0x1007
```

Warning: Logger did not log message as it was lower than the minimum level.

8.3.1.75 WARNING_NotImplementedYet

```
#define WARNING_NotImplementedYet 0xF004
```

Warning: This function has not been implemented yet.

8.3.1.76 WARNING_NotSendingEventAsNoListenersRegistered

```
#define WARNING_NotSendingEventAsNoListenersRegistered 0x100A
```

Warning: There are no listeners so we are skipping sending out to event listeners.

8.3.1.77 WARNING_NotSendingNotificationAsNoListenersRegistered

```
#define WARNING_NotSendingNotificationAsNoListenersRegistered 0x1009
```

Warning: There are no listeners so we are skipping sending out to notification listeners.

8.3.1.78 WARNING_PenBatteryLow

```
#define WARNING_PenBatteryLow 0x6001
```

Warning: Pen battery is getting low, please charge it.

8.3.1.79 WARNING_PenStopCalledPenAlreadyStopped

```
#define WARNING_PenStopCalledPenAlreadyStopped 0x100D
```

Warning: PenStop skipping as pen not started or already stopped.

8.3.1.80 WARNING_PenSurfaceSensorHasBackground

```
#define WARNING_PenSurfaceSensorHasBackground 0x6004
```

Warning: The pen's surface sensor has too much background.

8.3.1.81 WARNING_Radio_Comms_Inactive

```
#define WARNING_Radio_Comms_Inactive 0x5008
```

Warning: Radio comms are inactive, this action will not be performed.

8.3.1.82 WARNING_Repeated_logger_message_ignored

```
#define WARNING_Repeated_logger_message_ignored 0x1008
```

Warning: Logger has ignored a repeated message.

8.3.1.83 WARNING_Serial_Interface_address_already_set

```
#define WARNING_Serial_Interface_address_already_set 0x5009
```

Warning: This serial interface has already had an address set, being overwritten.

8.3.1.84 WARNING_SerialMessageReceivedWithIncorrectCRC

```
#define WARNING_SerialMessageReceivedWithIncorrectCRC 0x500C
```

Warning: Serial packet received with incorrect CRC.

8.3.1.85 WARNING_SerialPortHandshakeTimeout

```
#define WARNING_SerialPortHandshakeTimeout 0x500F
```

Warning: Serial port handshake has timed out, device not a Massless device, or not correctly working or connected.

8.3.1.86 WARNING_SerialSendBufferFullRemovingOldestCommand

```
#define WARNING_SerialSendBufferFullRemovingOldestCommand 0x500E
```

Warning: Serial send buffer is full, removing oldest command of type s.

8.3.1.87 WARNING_ThisPenExperiencedAtLeastOneHardFault

```
#define WARNING_ThisPenExperiencedAtLeastOneHardFault 0x6002
```

8.3.1.88 WARNING_UnknownSerialCommandRecieved

```
#define WARNING_UnknownSerialCommandRecieved 0x500A
```

Warning: Unknown serial command received.

8.3.1.89 WARNING_UsingDefaultCameraProperties

```
#define WARNING_UsingDefaultCameraProperties 0x2006
```

Warning: Using default camera properties.

8.4 MASSLESSPenSystem.h File Reference

```
#include <cstdint>
#include "MASSLESSPenSystemCallbacks.h"
```

Functions

- int PenStart (uint8_t IntegrationKey[16])
- int PenStop ()
- int PenSetUnits (float units)
- int PenAttachListener (PoseUpdate updateClient)
- int PenGetTrackerPose (int typeInt, float *x=0, float *y=0, float *z=0, float *qr=0, float *qx=0, float *qy=0, float *qz=0)
- int PenAttachStateListener (InputStateUpdate updateClient)
- int PenAttachFullPoseListener (FullPoseUpdate updateClient)
- int PenAttachSGSmoothedPoseListener (int SGOrder, PoseUpdate updateClient)
- int PenAttachFullStateListener (FullStateUpdate updateClient)
- int PenAttachNotificationListener (NotificationCallback updateClient)
- int PenAttachEventListener (EventCallback client)
- int PenSimpleBuzz (uint16_t DurationMs=208)
- int32_t PenGetSelectedTrackerID (uint8_t pointingType=0)
- int PenSetOriginTracker (int32_t ID)
- uint64_t PenGetDllVersionNumber ()
- int PenSwitchOff ()
- int GetDeviceOK (uint8_t *AskedPenOK, uint8_t *AskedTrackerOK, uint8_t *AskedCameraOK)
- int GetSpecificTrackerOK (uint8_t *AskedTrackerOK, uint8_t *AskedCameraOK, int TrackerIndex)
- int GetPenDetails (uint16_t *AskedHWVersion, uint16_t *AskedSWVersion, uint8_t *AskedType, uint8_t *AskedMAC, uint8_t *AskedSerial, int32_t *AskedHardFaultCount)
- int GetTrackerDetails (uint16_t *AskedHWVersion, uint16_t *AskedSWVersion, uint8_t *AskedType, uint8_t *AskedMAC, uint8_t *AskedSerial, int32_t *AskedHardFaultCount, int TrackerIndex=0)
- int GetSingleTrackerDetails (uint16_t *AskedHWVersion, uint16_t *AskedSWVersion, uint8_t *AskedType, uint8_t *AskedMAC, uint8_t *AskedSerial, int32_t *AskedHardFaultCount)
- int PenChangeIntegrationKey (uint8_t IntegrationKey[16])
- int PenSetErrorDiscretisationFactor (float ScalingFactor)
- int PenRecordBackground (float Duration=10.0)

Variables

- const uint16_t dllVersionMajor = 0
- const uint16_t dllVersionMinor = 8
- const uint32_t dllVersionPatch = 0

8.4.1 Function Documentation

8.4.1.1 GetDeviceOK()

```
int GetDeviceOK (
            uint8_t * AskedPenOK,
            uint8_t * AskedTrackerOK,
            uint8_t * AskedCameraOK )
```

Returns the OK message for each type of device

This will return an OK message for each type of device. If you only have one device, this will be the only one. If you have multiple trackers, it will check that at least one Camera, and at least one Tracker board are working. We return a 0 for not ok, and a 1 for OK.

Parameters

AskedPenOK	uint8_t* pointer to the uint8_t to fill with the pen's OK value
AskedTrackerOK	uint8_t* pointer to the uint8_t to fill with the tracker's OK value
AskedCameraOK	uint8_t* pointer to the uint8_t to fill with the camera's OK value

Returns

ErrorCode int 0 on success.

8.4.1.2 GetPenDetails()

```
int GetPenDetails (
            uint16_t * AskedHWVersion,
            uint16_t * AskedSWVersion,
            uint8_t * AskedType,
            uint8_t * AskedMAC,
            uint8_t * AskedSerial,
            int32_t * AskedHardFaultCount )
```

Returns the device details for the pen

This will return the full device details for the pen. It will ignore any specific details that you supply a null pointer or a zero for. Any that have a pointer will be filled with the data.

The serial number is deterministically generated from the MAC address. The hard fault count is the number of times the device has had a hard fault, where the watchdog fires and forces the reboot into bootloader.

The data that it returns is:

- Hardware version.
- Software version.
- Type of device (this should be a 2 for a pen).
- MAC address.
- Serial number.
- Number of hard faults (lifetime count).

WARNING: The calling code is responsible for memory management and therefore must pre-allocated the correct amount of memory for each of the properties requested. If you do not want to allocate memory for a property, pass in a nullptr or 0 and it will be ignored.

Sizes: Hardware version requires an array of three uint16_t to be allocated. Software version requires an array of three uint16_t to be allocated. Type of device requires a single uint8_t to be allocated. MAC address requires an array of six uint8_t to be allocated. Serial number requires an array of sixteen uint8_t to be allocated. Number of hard faults requires a single int32_t to be allocated.

Parameters

AskedHWVersion	uint16_t[3] Pointer to an array of 3 uint16_t to fill with the hardware version.
AskedSWVersion	uint16_t[3] Pointer to an array of 3 uint16_t to fill with the hardware version.
AskedType	uint8_t Pointer to a single uint8_t to fill with the type of device (uint8_t enum).
AskedMAC	uint8_t[6] Pointer to an array of 6 uint8_t to fill with the MAC address.
AskedSerial	uint8_t[16] Pointer to an array of 16 uint8_t to fill with the serial number.
AskedHardFaultCount	int32_t Pointer to a single int32_t to fill with the lifetime number of hard faults.

Returns

ErrorCode int 0 on success.

8.4.1.3 GetSingleTrackerDetails()

```
int GetSingleTrackerDetails (
            uint16_t * AskedHWVersion,
            uint16_t * AskedSWVersion,
            uint8_t * AskedType,
            uint8_t * AskedMAC,
            uint8_t * AskedSerial,
            int32_t * AskedHardFaultCount )
```

Returns the device details for the only tracker

This will return the full device details for the only tracker. If you have more than one tracker it will return the details for the first in the list (arbitary, based on timing during program startup). It will ignore any specific properties that you supply a null pointer or a zero for. Any that have a pointer will be filled with the data.

The serial number is deterministically generated from the MAC address. The hard fault count is the number of times the device has had a hard fault, where the watchdog fires and forces the reboot into bootloader.

The data that it returns is:

- Hardware version.
- Software version.
- Type of device (this should be a 1 for a tracker).
- MAC address.
- Serial number.
- Number of hard faults (lifetime count).

WARNING: The calling code is responsible for memory management and therefore must pre-allocated the correct amount of memory for each of the properties requested. If you do not want to allocate memory for a property, pass in a nullptr or 0 and it will be ignored.

Sizes: Hardware version requires an array of three uint16_t to be allocated. Software version requires an array of three uint16_t to be allocated. Type of device requires a single uint8_t to be allocated. MAC address requires an array of six uint8_t to be allocated. Serial number requires an array of sixteen uint8_t to be allocated. Number of hard faults requires a single int32_t to be allocated.

Parameters

AskedHWVersion	uint16_t[3] Pointer to an array of 3 uint16_t to fill with the hardware version.
AskedSWVersion	uint16_t[3] Pointer to an array of 3 uint16_t to fill with the hardware version.
AskedType	uint8_t Pointer to a single uint8_t to fill with the type of device (uint8_t enum).
AskedMAC	uint8_t[6] Pointer to an array of 6 uint8_t to fill with the MAC address.
AskedSerial	uint8_t[16] Pointer to an array of 16 uint8_t to fill with the serial number.
AskedHardFaultCount	int32_t Pointer to a single int32_t to fill with the lifetime number of hard faults.

Returns

ErrorCode int 0 on success.

8.4.1.4 GetSpecificTrackerOK()

```
int GetSpecificTrackerOK (
            uint8_t * AskedTrackerOK,
            uint8_t * AskedCameraOK,
            int TrackerIndex )
```

Returns the OK message for a specified tracker and camera

This will return an OK message for the tracker and camera that you request. It will check the pair of devices tracker and camera that are associated with the requested tracker index. We return a 0 for not ok, and a 1 for OK.

Parameters

AskedTrackerOK	uint8_t* pointer to the uint8_t to fill with the tracker's OK value
AskedCameraOK	uint8_t* pointer to the uint8_t to fill with the camera's OK value
TrackerIndex	int index to the tracker & camera pair that you want to query.

Returns

ErrorCode int 0 on success.

8.4.1.5 GetTrackerDetails()

```
int GetTrackerDetails (
            uint16_t * AskedHWVersion,
            uint16_t * AskedSWVersion,
            uint8_t * AskedType,
            uint8_t * AskedMAC,
            uint8_t * AskedSerial,
            int32_t * AskedHardFaultCount,
            int TrackerIndex = 0 )
```

Returns the device details for the chosen tracker

This will return the full device details for the chosen tracker. It will ignore any specific details that you supply a null pointer or a zero for. Any that have a pointer will be filled with the data.

The serial number is deterministically generated from the MAC address. The hard fault count is the number of times the device has had a hard fault, where the watchdog fires and forces the reboot into bootloader.

The data that it returns is:

- Hardware version.
- Software version.
- Type of device (this should be a 1 for a tracker).
- MAC address.
- Serial number.
- Number of hard faults (lifetime count).

WARNING: The calling code is responsible for memory management and therefore must pre-allocated the correct amount of memory for each of the properties requested. If you do not want to allocate memory for a property, pass in a nullptr or 0 and it will be ignored.

Sizes:

- Hardware version requires an array of three uint16_t to be allocated.
- Software version requires an array of three uint16_t to be allocated.
- Type of device requires a single uint8_t to be allocated.
- MAC address requires an array of six uint8_t to be allocated.
- Serial number requires an array of sixteen uint8_t to be allocated.
- Number of hard faults requires a single int32_t to be allocated.

Parameters

AskedHWVersion	uint16_t[3] Pointer to an array of 3 uint16_t to fill with the hardware version.
AskedSWVersion	uint16_t[3] Pointer to an array of 3 uint16_t to fill with the hardware version.
AskedType	uint8_t Pointer to a single uint8_t to fill with the type of device (uint8_t enum).
AskedMAC	uint8_t[6] Pointer to an array of 6 uint8_t to fill with the MAC address.
AskedSerial	uint8_t[16] Pointer to an array of 16 uint8_t to fill with the serial number.
AskedHardFaultCount	int32_t Pointer to a single int32_t to fill with the lifetime number of hard faults.
TrackerIndex	int Index of the tracker you want the details for. Optional, if ommitted, index 0 will be used.

Returns

ErrorCode int 0 on success.

8.4.1.6 PenAttachEventListener()

```
int PenAttachEventListener (
            EventCallback client )
```

Registers a listener callback to the Event stream

This will register a listener callback to the event stream. To receive specific events from the pen. This is things like "Camera OK", or "Pen low on battery", or "Pen was tapped". See the callback details for the format of what it returns. See the Callback Events for specific event details. This callback can be attached before PenStart is called. It is even recommended to do this so that you recieve any events generated during the PenStart function.

Parameters

client	EventCallback callback to receive the events.

Returns

ErrorCode int 0 on success.

8.4.1.7 PenAttachFullPoseListener()

```
int PenAttachFullPoseListener (
            FullPoseUpdate updateClient )
```

Registers a listener callback to the FullPose stream

This will register a listener callback to the full pose stream that is versioned, and contains, position, orientation, gyro data, error on position, and more (see callback details). PenStart must have been called already to attach this callback.

Parameters

updateClient	FullPoseUpdate callback to receive the updates.

Returns

ErrorCode int 0 on success.

8.4.1.8 PenAttachFullStateListener()

```
int PenAttachFullStateListener (
            FullStateUpdate updateClient )
```

Registers a listener callback to the FullStateUpdate stream

This will register a listener callback to the full input state stream that is versioned, and contains, cap sense, surface sense, raw surface data, and more (see callback details). PenStart must have been called already to attach this callback.

Parameters

updateClient	FullStateUpdate callback to receive the updates.

Returns

ErrorCode int 0 on success.

8.4.1.9 PenAttachListener()

```
int PenAttachListener (
            PoseUpdate updateClient )
```

Registers a listener callback to the PoseUpdate stream

This will register a listener callback to the simplest PoseUpdate stream that only receives position and orientation. PenStart must have been called already to attach this callback.

Parameters

updateClient	PoseUpdate callback to receive the updates.

Returns

ErrorCode int 0 on success.

8.4.1.10 PenAttachNotificationListener()

```
int PenAttachNotificationListener (
            NotificationCallback updateClient )
```

Registers a listener callback to the NotificationCallback stream

This will register a listener callback to the notification callbacks. It will be called for every log message that we have. You can filter on log type, and only process errors, please see the callback details (in their own header) for a full description. This callback can be attached before PenStart is called. It is even recommended to do this so that you recieve any notifications generated during the PenStart function.

Parameters

updateClient	NotificationCallback callback to receive the logs.

Returns

ErrorCode int 0 on success.

8.4.1.11 PenAttachSGSmoothedPoseListener()

```
int PenAttachSGSmoothedPoseListener (
            int SGOrder,
            PoseUpdate updateClient )
```

Registers a listener callback to one of the Savitsky-Golay smoothed pose updates

This will register a listener callback to one of the smoothed streams. It will take an integer as the order of the Savitsky-Golay filter. We only accept 5, 7 or 9 as the order, anything else will return an error.

Parameters

SGOrder	int The order of the Savitsky-Golay filter to connect to.
updateClient	PoseUpdate callback to receive the updates.

Returns

ErrorCode int 0 on success.

8.4.1.12 PenAttachStateListener()

```
int PenAttachStateListener (
            InputStateUpdate updateClient )
```

Registers a listener to the pen's basic input state

This will register a listener callback to the simplest InputStateUpdate stream that only receives CapSense and surface sense. PenStart must have been called already to attach this callback.

Parameters

updateClient	InputStateUpdate callback to receive the updates.

Returns

ErrorCode int 0 on success.

8.4.1.13 PenChangeIntegrationKey()

```
int PenChangeIntegrationKey (
            uint8_t IntegrationKey[16] )
```

Swaps the integration key, starting a new metrics tracking session

This is when we need to change the integration key because a new program has been launched it is better to do this without completely shutting down and restarting the whole tracking.

You must give it the integration key of which integration you are calling it from. These can be requested by messaging Dom the CTO of Massless, dom@massless.io they will be unique to your integration.

Parameters

IntegrationKey	uint8_t[16] this is the 16 byte UUID of your particular integration.

Returns

ErrorCode int 0 on success.

8.4.1.14 PenGetDllVersionNumber()

```
uint64_t PenGetDllVersionNumber ( )
```

Returns the DLL version number

This will return the version number of the DLL. It packs the data into a single uint64_t. The format is given by the example packing code below:

```
uint64_t version = 0;
version |= ((uint64_t)dllVersionMajor << (16 * 3));
version |= ((uint64_t)dllVersionMinor << (16 * 2));
version |= ((uint64_t)dllVersionPatch);
```

Returns

VersionNumber uint64_t Packed version of the DLL.

8.4.1.15 PenGetSelectedTrackerID()

```
int32_t PenGetSelectedTrackerID (
            uint8_t pointingType = 0 )
```

Returns the ID of the selected tracker

Not relevant until we support multiple trackers. This will select a tracker based on the pen's position relative to all of them. There are two types, pointing and proximity.

Parameters

pointingType	uint8_t number denoting the type of selection.

Returns

TrackerID int32_t ID of the selected tracker.

8.4.1.16 PenGetTrackerPose()

```
int PenGetTrackerPose (
            int typeInt,
            float * x = 0,
            float * y = 0,
            float * z = 0,
            float * qr = 0,
            float * qx = 0,
            float * qy = 0,
            float * qz = 0 )
```

Returns the pose of the tracker relative to the tracking reference.

This will use the direction of gravity to decide which way around the Massless tracker is relative to the tracking reference, it will then report the pose relative to this tracking reference, depending on what type of device the tracking reference is. Often you are calling this at the start of the program, before we receive the orientation of the tracker, so receiving an error from this function is fairly likely.

Parameters

typeInt	int an integer representing the type of tracking reference used.
x	float* A pointer to a float, ready to fill with the x value.
y	float* A pointer to a float, ready to fill with the y value.
z	float* A pointer to a float, ready to fill with the z value.
qr	float* A pointer to a float, ready to fill with the qr value.
qx	float* A pointer to a float, ready to fill with the qx value.
qy	float* A pointer to a float, ready to fill with the qy value.
qz	float* A pointer to a float, ready to fill with the qz value.

Returns

ErrorCode int 0 on success.

8.4.1.17 PenRecordBackground()

```
int PenRecordBackground (
            float Duration = 10.0 )
```

Tells the system to record the background markers

This sets off the process to record the background, and any markers it finds in this time will be noted down as "not the pen". This will help with elimination of things like the vive base stations, and also lights present in the room. It allows you to set an optional duration for how long you want the background recorded for. If you do not set a duration, then the default of 10s will be used.

Parameters

Duration	float The number of seconds to record the background for.

Returns

ErrorCode int 0 on success.

8.4.1.18 PenSetErrorDiscretisationFactor()

```
int PenSetErrorDiscretisationFactor (
            float ScalingFactor )
```

Sets the distcretization from error factor

This is used for setting the scaling factor from the error to set the discretization size. Separately on all three exes, it will only register a new position if that position is outside the region defied by the error multiplied by the scaling factor. This function sets the scaling factor. By default it is 1.0 which is exactly the error.

To turn this off completely, set the scaling factor to 0 (or negative).

Parameters

ScalingFactor	float The scaling factor parameter to set.

Returns

ErrorCode int 0 on success.

8.4.1.19 PenSetOriginTracker()

```
int PenSetOriginTracker (
            int32_t ID )
```

Sets the chosen tracker as the origin.

Not relevant until we support multiple trackers. This will tell the tracking library which tracker to use as the origin tracker, and report all measurements relative to it.

Parameters

ID	int32_t ID of the selected tracker to use

Returns

ErrorCode int 0 on success.

8.4.1.20 PenSetUnits()

```
int PenSetUnits (
            float units )
```

Sets the unit system for the tracking

Sets up the units for the system to report its measurements in. You need to tell it the number of your requested units that are in 1 metre. For example, 1.09361 for reporting in yards.

Parameters

units	float number of units that are in a metre.

Returns

ErrorCode int 0 on success.

8.4.1.21 PenSimpleBuzz()

```
int PenSimpleBuzz (
            uint16_t DurationMs = 208 )
```

Sends a command to tell the pen to buzz

This will send a command telling the pen to buzz. There is a default duration, but you can specify if you want.

Parameters

DurationMs	uint16_t duration in milliseconds of the buzz. (Optional)

Returns

ErrorCode int 0 on success.

8.4.1.22 PenStart()

```
int PenStart (
            uint8_t IntegrationKey[16] )
```

Starts the pen tracking system

This starts up the whole pen tracking system. Including the camera, the serial communications, and everything. You must give it the integration key of which integration you are calling it from. These can be requested by messaging Dom, the CTO of Massless, dom@massless.io. They will be unique to your integration. This must be called before most other functions, the only ones that can be called before this one, is PenAttachEventListener and PenAttachNotificationListener.

Parameters

IntegrationKey	uint8_t[16] this is the 16 byte UUID of your particular integration.

Returns

ErrorCode int 0 on success.

8.4.1.23 PenStop()

```
int PenStop ( )
```

Shuts down and stops the tracking releasing resources

Stops the tracking system by going through and releasing all resources that it needed. This includes the camera, and all serial ports. It also does cleanup sending final logs. It clears all the listeners including those listening to the event and notification channels. This is because when the system closes, we expect the caller to change.

Returns

ErrorCode int 0 on success.

8.4.1.24 PenSwitchOff()

```
int PenSwitchOff ( )
```

Sends a command to the pen to tell it to turn off

This will send the pen a command telling it to turn off.

Returns

ErrorCode int 0 on success.

8.4.2 Variable Documentation

8.4.2.1 dllVersionMajor

```
const uint16_t dllVersionMajor = 0
```

8.4.2.2 dllVersionMinor

```
const uint16_t dllVersionMinor = 8
```

8.4.2.3 dllVersionPatch

```
const uint32_t dllVersionPatch = 0
```

8.5 MASSLESSPenSystemCallbacks.h File Reference

```
#include <cstdint>
```

Typedefs

- typedef int(__stdcall * PoseUpdate) (uint32_t Length, uint8_t *PenPose)
- typedef int(__stdcall * InputStateUpdate) (uint32_t Length, uint8_t *InputState)
- typedef int(__stdcall * FullPoseUpdate) (uint32_t Length, uint8_t *Data)
- typedef int(__stdcall * FullStateUpdate) (uint32_t Length, uint8_t *Data)
- typedef int(__stdcall * NotificationCallback) (uint32_t Length, uint8_t *Data)
- typedef int(__stdcall * EventCallback) (uint32_t Length, uint8_t *Data)

Enumerations

- enum XRTrackingNodeType {
  OculusTrackerLegacy = 0, ViveLighthouseLegacy = 1, ViveTrackerLegacy = 2, OculusSensor = 3,
  ViveBaseStationV1 = 4, ViveTracker = 5, ViveBaseStationV2 = 6, None = 7,
  OculusTouchV2Right = 8, OculusTouchV2Left = 9, OculusTouchV2RightUnity = 10, OculusTouchV2LeftUnity
  = 11 }

8.5.1 Typedef Documentation

8.5.1.1 EventCallback

```
typedef int(__stdcall* EventCallback) (uint32_t Length, uint8_t *Data)
```

Function prototype for listener callbacks to recieve notifications of events that occur

For example tap events, and other Computer readable events. This is not for ascii messages to the user, but events to the program. We will document the schema for these messages in a separate document.

This is a message system, where we send events of a particular type, and we can ignore events that we don't understand. This should allow back compatibility where a newer version of the library works with older versions of Studio. It should also ensure back compatibility where a newer version of studio works with older versions of the library (because it will just not be sending the events that are being looked for).

Parameters

Length	uint32_t parameter detailing the number of bytes present in the payload.
Data	uint8_t* pointer to a message with the following format: • 0: uint16_t type of event being called • 2-n: binary payload with the data from the event.

8.5.1.2 FullPoseUpdate

```
typedef int(__stdcall* FullPoseUpdate) (uint32_t Length, uint8_t *Data)
```

Function prototype for listener callbacks to recieve the latest pose update information

This is an evolving packet, you check the version number, and if it's above the version where a certain piece of data was introduced then it should be present in the message. Best to also check the length is long enough to contain it. This should allow back compatibility where a newer version of the library works with older versions of Studio (because they just ignore newer data). It should also ensure back compatibility where a newer version of studio works with older versions of the library (because they will just not look for and default the data they might want).

Parameters

Length	uint32_t parameter detailing the number of bytes present in the main message.
Data	uint8_t* pointer to a message with the following format: • 0: uint8_t version of the message, newer versions will typically only append things to the message. • 1 - 28: data same as the PoseUpdate message (version 0 and above) • 29: uint8_t, flags to denote status format tbc (present but fixed to 0x01 for optically tracked in version 0) • 30: float*3 X,Y,Z of the gyro readings (version 1 and above, not implemented yet) • 42: float*3 X,Y,Z of the position error (version 2 and above, not implemented yet)

8.5.1.3 FullStateUpdate

```
typedef int(__stdcall* FullStateUpdate) (uint32_t Length, uint8_t *Data)
```

Function prototype for listener callbacks to recieve the latest status update information

This is an evolving packet, you check the version number, and if it's above the version where a certain piece of data was introduced then it should be present in the message. Best to also check the length is long enough to contain it. This should allow back compatibility where a newer version of the library works with older versions of Studio (because they just ignore newer data). It should also ensure back compatibility where a newer version of studio works with older versions of the library (because they will just not look for and default the data they might want).

Parameters

Length	uint32_t parameter detailing the number of bytes present in the main message.
Data	uint8_t* pointer to a message with the following format: • 0: uint8_t version of the message, newer versions will typically only append things to the message. • 1: float, surface sense reading (version 0 and above). • 5: uint8_t, capsense reading (version 0 and above). • 6: uint8_t, boolean surface or not surface (version 1 and above). • 7: uint8_t, tap detector reading set of bitfields (format tbc) (possibly version 2, may use separate event callback for taps and above).

8.5.1.4 InputStateUpdate

```
typedef int(__stdcall * InputStateUpdate) (uint32_t Length, uint8_t *InputState)
```

Function prototype for listener callbacks to receive input state information

Parameters

Length	uint32_t parameter detailing the number of bytes present in the main message.
PenPose	uint8_t* pointer to a message with the following format: • 0: float, proximity to a surface (of the tip). • 4: uint8_t, value of the capsense. 0xFF = "not touched"; 0x00 - 0xFE = linear slider where 0x00 is tip end and 0xFE is USBC end.

Returns

int Currently ignored, intended for future error reporting.

8.5.1.5 NotificationCallback

```
typedef int(__stdcall* NotificationCallback) (uint32_t Length, uint8_t *Data)
```

Function prototype for listener callbacks to recieve the latest notifications

This is intended for notifications and error messages. It is primarily to send ascii messages all the way to the user. Things like "no tracker connected" or "pen is low on battery".

This is an evolving packet, you check the version number, and if it's above the version where a certain piece of data was introduced then it should be present in the message. Best to also check the length is long enough to contain it. This should allow back compatibility where a newer version of the library works with older versions of Studio (because they just ignore newer data). It should also ensure back compatibility where a newer version of studio works with older versions of the library (because they will just not look for and default the data they might want).

Parameters

Length	uint32_t parameter detailing the number of bytes present in the main message.
Data	uint8_t* pointer to a message with the following format: • 0: uint8_t version of the message, newer versions will typically only append things to the message. • 1: uint8_t, type of message (error 3, warning 2, notification 1). (version 0 and above) • 2: Depending on the type, error and warning will have an integer here, uint16_t. (version 0 and above) • 2 or 4: There will also be an ascii string passed through, start byte address dependent on the type. Will be a null terminated ascii encoded C-string. (version 0 and above)

8.5.1.6 PoseUpdate

```
typedef int(__stdcall * PoseUpdate) (uint32_t Length, uint8_t *PenPose)
```

Function prototype for listener callbacks to receive pose information

Parameters

Length	uint32_t parameter detailing the number of bytes present in the main message.

Parameters

PenPose	uint8_t* pointer to a message with the following format: • 0: float, X position of the tip. • 4: float, Y position of the tip. • 8: float, Z position of the tip. • 12: float, Pen's orientation quaternion, real part. • 16: float, Pen's orientation quaternion, vector part, X component. • 20: float, Pen's orientation quaternion, vector part, Y component. • 24: float, Pen's orientation quaternion, vector part, Z component.

Returns

int Currently ignored, intended for future error reporting.

8.5.2 Enumeration Type Documentation

8.5.2.1 XRTrackingNodeType

```
enum XRTrackingNodeType
```

Declaration for each type of tracking reference. Depending on the tracking type you are using, you will need to ask for a different one of these to get the correct offsets for the Massless Tracker.

Enumerator

OculusTrackerLegacy	Please disregard the legacy options unless you are already using them.
ViveLighthouseLegacy	Please disregard the legacy options unless you are already using them.
ViveTrackerLegacy	Please disregard the legacy options unless you are already using them.
OculusSensor	OculusSensor is used to connect to an Oculus Sensor (with the Rift CV1).
ViveBaseStationV1	ViveBaseStationV1 is used to connect to the V1 Base Stations, (original generation Vive Base Station).
ViveTracker	ViveTraker is used to connect to a Vive Tracker, this is the recommended method for use with the Lighthouse tracking system.
ViveBaseStationV2	ViveBaseStationV2 is used to connect to the V2 Base Stations, (as shipped with Vive Pro and Valve Index).
None	None is used when you are not connecting to a VR reference, it will only tell you which way up the Massless Tracker is.
OculusTouchV2Right	OculusTouchV2Right is used to connect with an Oculus Touch (right hand) as shipped with a Rift S or Quest. This is for OpenVR based applications as the Touch's coordinate system is different to the others.
OculusTouchV2Left	OculusTouchV2Right is used to connect with an Oculus Touch (left hand) as shipped with a Rift S or Quest. This is for OpenVR based applications as the Touch's coordinate system is different to the others.

Enumerator

OculusTouchV2RightUnity	OculusTouchV2Right is used to connect with an Oculus Touch (right hand) as shipped with a Rift S or Quest. This is for Unity based applications as the Touch's coordinate system is different to the others.
OculusTouchV2LeftUnity	OculusTouchV2Right is used to connect with an Oculus Touch (left hand) as shipped with a Rift S or Quest. This is for Unity based applications as the Touch's coordinate system is different to the others.

Index