

spp\_profinet

Generated by Doxygen 1.8.6

Mon Feb 1 2016 08:33:44



# Contents

<b>1</b>	<b>Data Structure Index</b>	<b>1</b>
1.1	Data Structures . . . . .	1
<b>2</b>	<b>File Index</b>	<b>3</b>
2.1	File List . . . . .	3
<b>3</b>	<b>Data Structure Documentation</b>	<b>5</b>
3.1	Buffy Struct Reference . . . . .	5
3.1.1	Detailed Description . . . . .	5
3.1.2	Field Documentation . . . . .	5
3.1.2.1	initialized . . . . .	5
3.1.2.2	ops . . . . .	5
3.1.2.3	p . . . . .	5
3.2	Buffy_ops Struct Reference . . . . .	5
3.2.1	Detailed Description . . . . .	6
3.2.2	Field Documentation . . . . .	6
3.2.2.1	Buffy_free . . . . .	6
3.2.2.2	Buffy_get_bits16 . . . . .	6
3.2.2.3	Buffy_get_bits32 . . . . .	6
3.2.2.4	Buffy_get_bits64 . . . . .	7
3.2.2.5	Buffy_get_bits8 . . . . .	7
3.3	Dissector Struct Reference . . . . .	7
3.3.1	Detailed Description . . . . .	8
3.3.2	Field Documentation . . . . .	8
3.3.2.1	calling . . . . .	8
3.3.2.2	initialized . . . . .	8
3.3.2.3	ops . . . . .	8
3.4	Dissector_ops Struct Reference . . . . .	8
3.4.1	Detailed Description . . . . .	9
3.4.2	Field Documentation . . . . .	9
3.4.2.1	Dissector_dissect . . . . .	9
3.4.2.2	Dissector_free . . . . .	9

3.4.2.3	Dissector_getSub	9
3.4.2.4	Dissector_lower	9
3.4.2.5	Dissector_registerSub	9
3.4.2.6	Dissector_size	10
3.4.2.7	Dissector_upper	10
3.5	DissectorRegister Struct Reference	10
3.5.1	Detailed Description	10
3.5.2	Field Documentation	10
3.5.2.1	initialized	10
3.5.2.2	ops	11
3.6	DissectorRegister_ops Struct Reference	11
3.6.1	Detailed Description	11
3.6.2	Member Function Documentation	11
3.6.2.1	DissectorRegister_insert	11
3.6.3	Field Documentation	12
3.6.3.1	DissectorRegister_get	12
3.6.3.2	DissectorRegister_size	13
3.7	EtherHeader Struct Reference	13
3.7.1	Detailed Description	13
3.8	Frame Struct Reference	13
3.8.1	Detailed Description	14
3.9	HeaderInfo Struct Reference	14
3.9.1	Detailed Description	14
3.10	PNRTDissector Struct Reference	14
3.10.1	Detailed Description	14
3.11	ProtocolTree Struct Reference	15
3.11.1	Detailed Description	15
3.11.2	Field Documentation	15
3.11.2.1	branches	15
3.11.2.2	hInfo	15
3.11.2.3	initialized	15
3.11.2.4	ops	15
3.11.2.5	parent	15
3.12	ProtocolTree_ops Struct Reference	15
3.12.1	Detailed Description	16
3.12.2	Field Documentation	16
3.12.2.1	ProtocolTree_branch	16
3.12.2.2	ProtocolTree_findBranch	16
3.12.2.3	ProtocolTree_free	16
3.12.2.4	ProtocolTree_new	17

3.13	Sender Struct Reference	17
3.13.1	Detailed Description	17
3.13.2	Field Documentation	17
3.13.2.1	initialized	17
3.13.2.2	ops	17
3.14	Sender_ops Struct Reference	17
3.14.1	Detailed Description	18
3.14.2	Field Documentation	18
3.14.2.1	Sender_free	18
3.14.2.2	Sender_send	18
3.15	Truffle Struct Reference	18
3.15.1	Detailed Description	19
3.16	UnixSocketSender Struct Reference	19
3.16.1	Detailed Description	19
3.16.2	Field Documentation	19
3.16.2.1	sender	19
<b>4</b>	<b>File Documentation</b>	<b>21</b>
4.1	src/Profinet/Bufy-int.h File Reference	21
4.1.1	Detailed Description	21
4.2	src/Profinet/Bufy.h File Reference	21
4.2.1	Detailed Description	22
4.2.2	Function Documentation	22
4.2.2.1	Bufy_free	22
4.2.2.2	Bufy_get_bits16	22
4.2.2.3	Bufy_get_bits32	22
4.2.2.4	Bufy_get_bits64	22
4.2.2.5	Bufy_get_bits8	23
4.2.2.6	Bufy_new	23
4.3	src/Profinet/Dissector-int.h File Reference	23
4.3.1	Detailed Description	23
4.4	src/Profinet/Dissector.h File Reference	24
4.4.1	Detailed Description	24
4.4.2	Function Documentation	24
4.4.2.1	Dissector_dissect	24
4.4.2.2	Dissector_free	24
4.4.2.3	Dissector_getSub	24
4.4.2.4	Dissector_new	25
4.4.2.5	Dissector_registerSub	25
4.5	src/Profinet/DissectorRegister-int.h File Reference	25

4.5.1	Detailed Description	26
4.6	src/Profinet/DissectorRegister.h File Reference	26
4.6.1	Detailed Description	26
4.6.2	Function Documentation	26
4.6.2.1	DissectorRegister_get	26
4.6.2.2	DissectorRegister_insert	26
4.6.2.3	DissectorRegister_new	27
4.7	src/Profinet/PNRTDissector.c File Reference	27
4.7.1	Detailed Description	27
4.7.2	Function Documentation	27
4.7.2.1	PNRTDissector_dissect	27
4.7.2.2	PNRTDissector_free	28
4.7.2.3	PNRTDissector_new	28
4.8	src/Profinet/ProtocolTree-int.h File Reference	28
4.8.1	Detailed Description	28
4.9	src/Profinet/ProtocolTree.h File Reference	28
4.9.1	Detailed Description	29
4.9.2	Function Documentation	29
4.9.2.1	ProtocolTree_branch	29
4.9.2.2	ProtocolTree_findBranch	29
4.9.2.3	ProtocolTree_free	29
4.9.2.4	ProtocolTree_new	30
4.10	src/Profinet/Sender-int.h File Reference	30
4.10.1	Detailed Description	30
4.11	src/Profinet/Sender.h File Reference	30
4.11.1	Detailed Description	31
4.11.2	Function Documentation	31
4.11.2.1	Sender_free	31
4.11.2.2	Sender_new	31
4.11.2.3	Sender_send	31
4.12	src/Profinet/Truffle.h File Reference	31
4.12.1	Detailed Description	32
4.13	src/Profinet/UnixSocketSender.c File Reference	32
4.13.1	Detailed Description	32
4.13.2	Function Documentation	32
4.13.2.1	UnixSocketSender_free	32
4.13.2.2	UnixSocketSender_new	32
4.13.2.3	UnixSocketSender_send	33
4.14	src/spp_profinet.c File Reference	33
4.14.1	Detailed Description	33

4.14.2	Function Documentation	33
4.14.2.1	DissectorInit	33
4.14.2.2	SetupProfiNet	33
4.14.3	Variable Documentation	33
4.14.3.1	sender	33
4.14.3.2	tlRegister	34
4.15	src/spp_profinet.h File Reference	34
4.15.1	Detailed Description	34
4.15.2	Function Documentation	34
4.15.2.1	SetupProfiNet	34
<b>Index</b>		<b>35</b>





# Chapter 1

## Data Structure Index

### 1.1 Data Structures

Here are the data structures with brief descriptions:

<a href="#">Buffy</a>	Buffer for dissecting packages in the profinet plugin . . . . .	5
<a href="#">Buffy_ops</a>	The operations that can be called by a <a href="#">Buffy</a> buffer . . . . .	5
<a href="#">Dissector</a>	Used to dissect certain data ranges within a package . . . . .	7
<a href="#">Dissector_ops</a>	The operations that can be called by a <a href="#">Dissector</a> . . . . .	8
<a href="#">DissectorRegister</a>	The datastructure for registering Dissectors on their specific intervals . . . . .	10
<a href="#">DissectorRegister_ops</a>	The operations that can be called by a <a href="#">DissectorRegister</a> . . . . .	11
<a href="#">EtherHeader</a>	Houses specific information about the ether header . . . . .	13
<a href="#">Frame</a>	Houses specific information about the frame . . . . .	13
<a href="#">HeaderInfo</a>	Info that can be inserte into a protocol tree as new branch . . . . .	14
<a href="#">PNRTDissector</a>	The <a href="#">Dissector</a> for Profi Real Time IO 0x8892 . . . . .	14
<a href="#">ProtocolTree</a>	Buffer for dissecting packages in the profinet plugin . . . . .	15
<a href="#">ProtocolTree_ops</a>	The operations that can be called by a <a href="#">ProtocolTree</a> . . . . .	15
<a href="#">Sender</a>	<a href="#">Sender</a> for sending Truffles to a specified port/socket/mq/sma . . . . .	17
<a href="#">Sender_ops</a>	The operations that can be called by a <a href="#">Sender</a> . . . . .	17
<a href="#">Truffle</a>	The datastructure for sending relevant information to another process . . . . .	18
<a href="#">UnixSocketSender</a>	Sends Truffles to a unix socket a client is reading from . . . . .	19



## Chapter 2

# File Index

### 2.1 File List

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

src/ <a href="#">spp_profinet.c</a>	
Snort Preprocessor Plugin Source File ProfiNet Purpose: . . . . .	33
src/ <a href="#">spp_profinet.h</a>	
Snort Preprocessor Plugin Header . . . . .	34
src/Profinet/ <a href="#">Buffy-int.h</a>	
The internal structure of <a href="#">Buffy</a> . . . . .	21
src/Profinet/ <a href="#">Buffy.h</a>	
The interface for <a href="#">Buffy</a> . . . . .	21
src/Profinet/ <a href="#">Dissector-int.h</a>	
This Header describes the internal structure of the <a href="#">Dissector</a> type, it defines the basic interface for operations . . . . .	23
src/Profinet/ <a href="#">Dissector.h</a>	
The Basic <a href="#">Dissector</a> abstraction (Interface) . . . . .	24
src/Profinet/ <a href="#">DissectorRegister-int.h</a>	
The internal structure of a dissector register. Including the operation structure and fields . . . . .	25
src/Profinet/ <a href="#">DissectorRegister.h</a>	
The interface for dissector registers . . . . .	26
src/Profinet/ <a href="#">PNRTDissector.c</a>	
<a href="#">PNRTDissector</a> implementation . . . . .	27
src/Profinet/ <a href="#">ProtocolTree-int.h</a>	
The internal sturcture of <a href="#">ProtocolTree</a> . . . . .	28
src/Profinet/ <a href="#">ProtocolTree.h</a>	
The interface for <a href="#">ProtocolTree</a> . . . . .	28
src/Profinet/ <a href="#">Sender-int.h</a>	
The internal structure of <a href="#">Sender</a> . . . . .	30
src/Profinet/ <a href="#">Sender.h</a>	
The sender interface . . . . .	30
src/Profinet/ <a href="#">Truffle.h</a>	
The structure of a <a href="#">Truffle</a> that is send via ipc . . . . .	31
src/Profinet/ <a href="#">UnixSocketSender.c</a>	
This file houses the operations that are specific for a <a href="#">UnixSocketSender</a> . . . . .	32



## Chapter 3

# Data Structure Documentation

### 3.1 Buffy Struct Reference

Buffer for dissecting packages in the profinet plugin.

```
#include <Buffy-int.h>
```

#### Data Fields

- bool [initialized](#)
- Packet \* [p](#)
- const struct [Buffy\\_ops](#) \* [ops](#)

#### 3.1.1 Detailed Description

Buffer for dissecting packages in the profinet plugin.

#### 3.1.2 Field Documentation

##### 3.1.2.1 bool Buffy::initialized

Whether this buffer was initialized.

##### 3.1.2.2 const struct [Buffy\\_ops](#)\* Buffy::ops

The buffer operations.

##### 3.1.2.3 Packet\* Buffy::p

Pointer to the snort package this buffer was created from

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

- src/Profinet/[Buffy-int.h](#)

### 3.2 Buffy\_ops Struct Reference

The operations that can be called by a [Buffy](#) buffer.

```
#include <Buffy-int.h>
```

## Data Fields

- `void(* Buffy\_free )(Buffy_t *buffy)`  
*Frees the given buffer from memory.*
- `uint8_t(* Buffy\_get\_bits8 )(Buffy_t *this, unsigned int bit_offset, const int no_of_bits)`  
*Get 1 - 8 bits returned in a uint8.*
- `uint16_t(* Buffy\_get\_bits16 )(Buffy_t *this, unsigned int bit_offset, const int no_of_bits, const unsigned int encoding)`  
*Get 1 - 16 bits returned in a uint16.*
- `uint32_t(* Buffy\_get\_bits32 )(Buffy_t *this, unsigned int bit_offset, const int no_of_bits, const unsigned int encoding)`  
*Get 1 - 32 bits returned in a uint32.*
- `uint64_t(* Buffy\_get\_bits64 )(Buffy_t *this, unsigned int bit_offset, const int no_of_bits, const unsigned int encoding)`  
*Get 1 - 64 bits returned in a uint64.*

### 3.2.1 Detailed Description

The operations that can be called by a [Buffy](#) buffer.

### 3.2.2 Field Documentation

#### 3.2.2.1 `void(* Buffy\_ops::Buffy\_free )(Buffy_t *buffy)`

Frees the given buffer from memory.

Parameters

<i>buffy</i>	the buffer to be freed
--------------	------------------------

#### 3.2.2.2 `uint16_t(* Buffy\_ops::Buffy\_get\_bits16 )(Buffy_t *this, unsigned int bit_offset, const int no_of_bits, const unsigned int encoding)`

Get 1 - 16 bits returned in a uint16.

Parameters

<i>this</i>	the calling buffer
<i>bit_offset</i>	the offset for from the currenty buffer position
<i>the</i>	number of bits to be read

Returns

unsigned 16 bit value representing the specified bit range

#### 3.2.2.3 `uint32_t(* Buffy\_ops::Buffy\_get\_bits32 )(Buffy_t *this, unsigned int bit_offset, const int no_of_bits, const unsigned int encoding)`

Get 1 - 32 bits returned in a uint32.

## Parameters

<i>this</i>	the calling buffer
<i>bit_offset</i>	the offset for from the currenty buffer position
<i>the</i>	number of bits to be read Gu

## Returns

unsigned 32 bit value representing the specified bit range

3.2.2.4 `uint64_t(* Buffy_ops::Buffy_get_bits64)(Buffy_t* this, unsigned int bit_offset, const int no_of_bits, const unsigned int encoding)`

Get 1 - 64 bits returned in a uint64.

## Parameters

<i>this</i>	the calling buffer
<i>bit_offset</i>	the offset for from the currenty buffer position
<i>the</i>	number of bits to be read

## Returns

unsigned 64 bit value representing the specified bit range

3.2.2.5 `uint8_t(* Buffy_ops::Buffy_get_bits8)(Buffy_t* this, unsigned int bit_offset, const int no_of_bits)`

Get 1 - 8 bits returned in a uint8.

## Parameters

<i>this</i>	the calling buffer
<i>bit_offset</i>	the offset for from the currenty buffer position
<i>the</i>	number of bits to be read

## Returns

unsigned 8 bit value representing the specified bit range

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

- [src/Profinet/Buffy-int.h](#)

## 3.3 Dissector Struct Reference

Used to dissect certain data ranges within a package.

```
#include <Dissector-int.h>
```

## Data Fields

- bool [initialized](#)
- const struct [Dissector\\_ops](#) \* [ops](#)
- [Dissector\\_t](#) \* [calling](#)

### 3.3.1 Detailed Description

Used to dissect certain data ranges within a package.

[Dissector](#) are used to dissect certain ranges of data in a network package, while having the possibility to link to further dissectors when the dissection of the desired range is complete. Further Dissectors are linked by using an internal [DissectorRegister](#).

-> It is possible to link several Dissectors together building a tree of dissectors and subdissectors that call each other when their dissection part is completed.

### 3.3.2 Field Documentation

#### 3.3.2.1 `Dissector_t*` `Dissector::calling`

The dissector this dissector has been called from.

#### 3.3.2.2 `bool` `Dissector::initialized`

Whether this dissector was initialized.

#### 3.3.2.3 `const struct Dissector_ops*` `Dissector::ops`

The dissectors operations.

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

- `src/Profinet/Dissector-int.h`

## 3.4 Dissector\_ops Struct Reference

The operations that can be called by a [Dissector](#).

```
#include <Dissector-int.h>
```

### Data Fields

- `size_t` [Dissector\\_size](#)  
*Returns the number of subdissectors in this dissector.*
- `uint64_t` [Dissector\\_lower](#)  
*Returns the lower bound this subdissector is being called upon.*
- `uint64_t` [Dissector\\_upper](#)  
*Returns the upper bound this subdissector is being called upon.*
- `void(* Dissector_free)(Dissector_t *dissector)`  
*Returns the number of subdissectors in this dissector.*
- `Dissector_t *(* Dissector_registerSub)(Dissector_t *this, Dissector_t *subDissector, Interval interval)`  
*Registers a given sub dissector on this dissector.*
- `Dissector_t *(* Dissector_getSub)(Dissector_t *this, uint64_t data)`  
*Returns the sub dissector that is register for the given unsigned long.*
- `int(* Dissector_dissect)(Dissector_t *this, Buffer_t *buf, ProtocolTree_t *tree)`  
*Dissects the package the given buffer is pointing to.*



### 3.4.1 Detailed Description

The operations that can be called by a [Dissector](#).

### 3.4.2 Field Documentation

#### 3.4.2.1 `int(* Dissector_ops::Dissector_dissect)(Dissector_t *this, Buffer_t *buf, ProtocolTree_t *tree)`

Dissects the package the given buffer is pointing to.

##### Parameters

<i>this</i>	the calling <a href="#">Dissector</a>
<i>buf</i>	the buffer pointing to the package data currently being processed
<i>tree</i>	the tree strcture to save the package data in

##### Returns

0 if the dissection was successful wihtout any failures, -1 if it was a faulty package. The fault flag will be set in the [ProtocolTree](#) accordingly

#### 3.4.2.2 `void(* Dissector_ops::Dissector_free)(Dissector_t *dissector)`

Returns the number of subdissectors in this dissector.

##### Returns

the number of sub-dissectors in this dissector

#### 3.4.2.3 `Dissector_t*(* Dissector_ops::Dissector_getSub)(Dissector_t *this, uint64_t data)`

Returns the sub dissector that is register for the given unsigned long.

##### Parameters

<i>this</i>	the dissector calling <code>Dissector_getSub</code>
<i>data</i>	the value for looking up in the dissector register

##### Returns

the registered sub dissector if any, NULL otherwise

#### 3.4.2.4 `uint64_t Dissector_ops::Dissector_lower`

Returns the lower bound this subdissector is being called upon.

##### Returns

the lower bound this subdissector is being called upon

#### 3.4.2.5 `Dissector_t*(* Dissector_ops::Dissector_registerSub)(Dissector_t *this, Dissector_t *subDissector, Interval interval)`

Registers a given sub dissector on this dissector.

**Parameters**

<i>this</i>	the dissector to register the subDissector on
<i>subDissector</i>	the dissector to be registered as sub

**Returns**

NULL if there was no other dissector registered for the given interval otherwise the existing [Dissector](#) will be overwritten and returned.

**3.4.2.6 size\_t Dissector\_ops::Dissector\_size**

Returns the number of subdissectors in this dissector.

**Returns**

the number of sub-dissectors in this dissector

**3.4.2.7 uint64\_t Dissector\_ops::Dissector\_upper**

Returns the upper bound this subdissector is being called upon.

**Returns**

the upper bound this subdissector is being called upon

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

- [src/Profinet/Dissector-int.h](#)

## 3.5 DissectorRegister Struct Reference

The datastructure for registering Dissectors on their specific intervals.

```
#include <DissectorRegister-int.h>
```

**Data Fields**

- bool [initialized](#)
- const struct [DissectorRegister\\_ops](#) \* [ops](#)

### 3.5.1 Detailed Description

The datastructure for registering Dissectors on their specific intervals.

The dissector register is used to register dissectors to intervals. Thereby making it possible to dissect a package while using certain data ranges for calling a next dissector that is mapped to the given data.

### 3.5.2 Field Documentation

**3.5.2.1 bool DissectorRegister::initialized**

Whether this dissector register is initialized.

3.5.2.2 `const struct DissectorRegister_ops* DissectorRegister::ops`

The dissector register operations.

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

- `src/Profinet/DissectorRegister-int.h`

## 3.6 DissectorRegister\_ops Struct Reference

The operations that can be called by a [DissectorRegister](#).

```
#include <DissectorRegister-int.h>
```

### Public Member Functions

- [Dissector\\_t](#) \* [DissectorRegister\\_insert](#) ([DissectorRegister\\_t](#) \*this, [Dissector\\_t](#) \*dissector)  
*Inserts a new [Dissector](#).*

### Data Fields

- `size_t` [DissectorRegister\\_size](#)  
*Returns the number dissectors registered.*
- `void` \*(\* [DissectorRegister\\_free](#) )([DissectorRegister\\_t](#) \*this)  
*Frees the given [DissectorRegister](#).*
- [Dissector\\_t](#) \*(\* [DissectorRegister\\_get](#) )([DissectorRegister\\_t](#) \*this, `uint64_t` data)  
*Returns the [Dissector](#) that is registered for the given unsigned long.*

### 3.6.1 Detailed Description

The operations that can be called by a [DissectorRegister](#).

### 3.6.2 Member Function Documentation

#### 3.6.2.1 `Dissector_t* DissectorRegister_ops::DissectorRegister_insert ( DissectorRegister_t * this, Dissector_t * dissector )`

Inserts a new [Dissector](#).

The new dissector will be inserted into the [DissectorRegister](#) by obtaining its lower and upper identifier bounds and mapping it accordingly.

#### Parameters

<i>this</i>	the calling register
<i>dissector</i>	the dissector to be inserted

#### Returns

NULL if there is no previous dissector registered within its interval, otherwise overwrites the old dissector and returns it

### 3.6.3 Field Documentation

#### 3.6.3.1 `Dissector_t>(* DissectorRegister_ops::DissectorRegister_get)(DissectorRegister_t *this, uint64_t data)`

Returns the [Dissector](#) that is registered for the given unsigned long.

## Parameters

<i>this</i>	the <a href="#">DissectorRegister</a> calling
<i>data</i>	the value for looking up in the <a href="#">DissectorRegister</a>

## Returns

the registered [Dissector](#) if any, NULL otherwise

3.6.3.2 `size_t DissectorRegister_ops::DissectorRegister_size`

Returns the number dissectors registered.

## Returns

the number of dissectors in this register

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

- `src/Profinet/DissectorRegister-int.h`

## 3.7 EtherHeader Struct Reference

Houses specific information about the ether header.

```
#include <Truffle.h>
```

## Data Fields

- `uint64_t sourceMacAddress`
- `uint64_t destMacAddress`
- `uint16_t etherType`

### 3.7.1 Detailed Description

Houses specific information about the ether header.

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

- `src/Profinet/Truffle.h`

## 3.8 Frame Struct Reference

Houses specific information about the frame.

```
#include <Truffle.h>
```

## Data Fields

- `uint16_t frameID`
- `char destName [30]`
- `char srcName [30]`
- `long long cycleCounter`

### 3.8.1 Detailed Description

Houses specific information about the frame.

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

- src/Profinet/[Truffle.h](#)

## 3.9 HeaderInfo Struct Reference

Info that can be inserte into a protocol tree as new branch.

```
#include <ProtocolTree.h>
```

### Data Fields

- char [caption](#) [256]  
*The caption of this info field.*
- uint64\_t [bitmask](#)  
*Interesting bits that can be set.*
- char [infofield](#) [256]  
*Infofield, can contain any information in char format for specific size.*
- long long [value](#)  
*A value that can be put for information.*
- int [type](#)  
*Specifies the type of information.*

### 3.9.1 Detailed Description

Info that can be inserte into a protocol tree as new branch.

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

- src/Profinet/[ProtocolTree.h](#)

## 3.10 PNRTDissector Struct Reference

The [Dissector](#) for Profi Real Time IO 0x8892.

### Data Fields

- struct [Dissector](#) [dissector](#)  
*Houses a [Dissector](#) internally for safe type casting.*

### 3.10.1 Detailed Description

The [Dissector](#) for Profi Real Time IO 0x8892.

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

- src/Profinet/[PNRTDissector.c](#)

## 3.11 ProtocolTree Struct Reference

Buffer for dissecting packages in the profinet plugin.

```
#include <ProtocolTree-int.h>
```

### Data Fields

- bool [initialized](#)
- struct [HeaderInfo](#) \* [hInfo](#)
- ProtocolTree\_t \* [parent](#)
- ProtocolTree\_t \*\* [branches](#)
- const struct [ProtocolTree\\_ops](#) \* [ops](#)

### 3.11.1 Detailed Description

Buffer for dissecting packages in the profinet plugin.

### 3.11.2 Field Documentation

#### 3.11.2.1 ProtocolTree\_t\*\* ProtocolTree::branches

Pointing to the branching protocol trees of this root node

#### 3.11.2.2 struct HeaderInfo\* ProtocolTree::hInfo

The Info field of this Subtree

#### 3.11.2.3 bool ProtocolTree::initialized

Whether this protocol Subtree was initialized.

#### 3.11.2.4 const struct ProtocolTree\_ops\* ProtocolTree::ops

The operations that can be called by a [ProtocolTree](#)

#### 3.11.2.5 ProtocolTree\_t\* ProtocolTree::parent

Pointer to the parent subtree

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

- src/Profinet/[ProtocolTree-int.h](#)

## 3.12 ProtocolTree\_ops Struct Reference

The operations that can be called by a [ProtocolTree](#).

```
#include <ProtocolTree-int.h>
```

## Data Fields

- `ProtocolTree_t *(* ProtocolTree_new )()`  
*Creates a new [ProtocolTree](#).*
- `void(* ProtocolTree_free )(ProtocolTree_t *proto)`  
*Frees the given [ProtocolTree](#) from memory.*
- `ProtocolTree_t *(* ProtocolTree_branch )(ProtocolTree_t *this, struct HeaderInfo *info)`  
*Creates a new branch with the given info field from the current root pointer of this [ProtocolTree](#).*
- `ProtocolTree_t *(* ProtocolTree_findBranch )(ProtocolTree_t *this, char *caption)`  
*Searches and returns the branch with the given caption.*

### 3.12.1 Detailed Description

The operations that can be called by a [ProtocolTree](#).

### 3.12.2 Field Documentation

#### 3.12.2.1 `ProtocolTree_t>(* ProtocolTree_ops::ProtocolTree_branch)(ProtocolTree_t *this, struct HeaderInfo *info)`

Creates a new branch with the given info field from the current root pointer of this [ProtocolTree](#).

##### Parameters

<i>this</i>	the calling <a href="#">ProtocolTree</a>
<i>info</i>	the header info to be inserted for the new subtree

##### Returns

A pointer to a Subtree with the newly created branch as its root pointer.

#### 3.12.2.2 `ProtocolTree_t>(* ProtocolTree_ops::ProtocolTree_findBranch)(ProtocolTree_t *this, char *caption)`

Searches and returns the branch with the given caption.

##### Parameters

<i>this</i>	the calling <a href="#">ProtocolTree</a>
<i>the</i>	caption to be searched for

##### Returns

the [ProtocolTree](#) starting at the found branch, NULL if there is no such branch.

#### 3.12.2.3 `void(* ProtocolTree_ops::ProtocolTree_free)(ProtocolTree_t *proto)`

Frees the given [ProtocolTree](#) from memory.

##### Parameters

<i>proto</i>	the <a href="#">ProtocolTree</a> to be freed
--------------	--



#### 3.12.2.4 ProtocolTree\_t>(\* ProtocolTree\_ops::ProtocolTree\_new)()

Creates a new [ProtocolTree](#).

##### Returns

the instantiated Tree

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

- src/Profinet/[ProtocolTree-int.h](#)

## 3.13 Sender Struct Reference

[Sender](#) for sending Truffles to a specified port/socket/mq/sma.

```
#include <Sender-int.h>
```

### Data Fields

- bool [initialized](#)
- const struct [Sender\\_ops](#) \* [ops](#)

#### 3.13.1 Detailed Description

[Sender](#) for sending Truffles to a specified port/socket/mq/sma.

#### 3.13.2 Field Documentation

##### 3.13.2.1 bool Sender::initialized

Whether this sender was initialized.

##### 3.13.2.2 const struct Sender\_ops\* Sender::ops

The sender operations.

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

- src/Profinet/[Sender-int.h](#)

## 3.14 Sender\_ops Struct Reference

The operations that can be called by a [Sender](#).

```
#include <Sender-int.h>
```

### Data Fields

- int(\* [Sender\\_free](#))([Sender\\_t](#) \*sender)  
*Frees the given sender.*
- int(\* [Sender\\_send](#))([Sender\\_t](#) \*this, [Truffle\\_t](#) \*truffle)

### 3.14.1 Detailed Description

The operations that can be called by a [Sender](#).

### 3.14.2 Field Documentation

#### 3.14.2.1 `int(* Sender_ops::Sender_free)(Sender_t *sender)`

Frees the given sender.

##### Parameters

<i>sender</i>	the sender to be freed
---------------	------------------------

##### Returns

0 if the freeing was successful, -1 otherwise

#### 3.14.2.2 `int(* Sender_ops::Sender_send)(Sender_t *this, Truffle_t *truffle)`

Sends the given truffle to the specified ipc

##### Parameters

<i>this</i>	the calling sender
<i>truffle</i>	the truffle to be send

##### Returns

0 if the sending was successful, -1 if no client is detected for receiving, or on other errors.

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

- `src/Profinet/Sender-int.h`

## 3.15 Truffle Struct Reference

The datastructure for sending relevant information to another process.

```
#include <Truffle.h>
```

### Data Fields

- `uint64_t flags`

*Flags are used for specific boolean states that are relevant for the whole package.*

- `struct EtherHeader eh`

*The Etherheader holds information from the etherheader of the network package.*

- `struct Frame frame`

*The [Frame](#) structure encapsulates information about the [Frame](#) within the network package.*

### 3.15.1 Detailed Description

The datastructure for sending relevant information to another process.

The [Truffle](#) is the datastructure that encapsulates all necessary and important information about a processed Network Packet. The structure of the [Truffle](#) is also known by the clients that want to receive information about the network package.

Like this clients are able to cast incoming data to this data type and immediately read out the relevant data.

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

- [src/Profinet/Truffle.h](#)

## 3.16 UnixSocketSender Struct Reference

Sends Truffles to a unix socket a client is reading from.

### Data Fields

- struct [Sender sender](#)

### 3.16.1 Detailed Description

Sends Truffles to a unix socket a client is reading from.

### 3.16.2 Field Documentation

#### 3.16.2.1 struct [Sender UnixSocketSender::sender](#)

The encapsulated sender type for save casting.

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

- [src/Profinet/UnixSocketSender.c](#)



## Chapter 4

# File Documentation

### 4.1 src/Profinet/Bufgy-int.h File Reference

The internal structure of [Bufgy](#).

#### Data Structures

- struct [Bufgy\\_ops](#)  
*The operations that can be called by a [Bufgy](#) buffer.*
- struct [Bufgy](#)  
*Buffer for dissecting packages in the profinet plugin.*

#### Functions

- [Bufgy\\_t](#) \* **Bufgy\_new** (Packet \*p)

#### 4.1.1 Detailed Description

The internal structure of [Bufgy](#).

### 4.2 src/Profinet/Bufgy.h File Reference

The interface for [Bufgy](#).

#### Functions

- [Bufgy\\_t](#) \* [Bufgy\\_new](#) (Packet \*p)  
*Creates a new buffer from the given snort package.*
- void [Bufgy\\_free](#) ([Bufgy\\_t](#) \*bufgy)  
*Frees the given buffer from memory.*
- [uint8\\_t](#) [Bufgy\\_get\\_bits8](#) ([Bufgy\\_t](#) \*this, unsigned int bit\_offset, const int no\_of\_bits)  
*Get 1 - 8 bits returned in a uint8.*
- [uint16\\_t](#) [Bufgy\\_get\\_bits16](#) ([Bufgy\\_t](#) \*this, unsigned int bit\_offset, const int no\_of\_bits, const unsigned int encoding)  
*Get 1 - 16 bits returned in a uint16.*

- uint32\_t [Buffy\\_get\\_bits32](#) (Buffy\_t \*this, unsigned int bit\_offset, const int no\_of\_bits, const unsigned int encoding)  
Get 1 - 32 bits returned in a uint32.
- uint64\_t [Buffy\\_get\\_bits64](#) (Buffy\_t \*this, unsigned int bit\_offset, const int no\_of\_bits, const unsigned int encoding)  
Get 1 - 64 bits returned in a uint64.

#### 4.2.1 Detailed Description

The interface for [Buffy](#).

#### 4.2.2 Function Documentation

##### 4.2.2.1 void Buffy\_free ( Buffy\_t \* buffy )

Frees the given buffer from memory.

Parameters

<i>buffy</i>	the buffer to be freed
--------------	------------------------

##### 4.2.2.2 uint16\_t Buffy\_get\_bits16 ( Buffy\_t \* this, unsigned int bit\_offset, const int no\_of\_bits, const unsigned int encoding )

Get 1 - 16 bits returned in a uint16.

Parameters

<i>this</i>	the calling buffer
<i>bit_offset</i>	the offset for from the currenty buffer position
<i>the</i>	number of bits to be read

Returns

unsigned 16 bit value representing the specified bit range

##### 4.2.2.3 uint32\_t Buffy\_get\_bits32 ( Buffy\_t \* this, unsigned int bit\_offset, const int no\_of\_bits, const unsigned int encoding )

Get 1 - 32 bits returned in a uint32.

Parameters

<i>this</i>	the calling buffer
<i>bit_offset</i>	the offset for from the currenty buffer position
<i>the</i>	number of bits to be read

Returns

unsigned 32 bit value representing the specified bit range

##### 4.2.2.4 uint64\_t Buffy\_get\_bits64 ( Buffy\_t \* this, unsigned int bit\_offset, const int no\_of\_bits, const unsigned int encoding )

Get 1 - 64 bits returned in a uint64.

## Parameters

<i>this</i>	the calling buffer
<i>bit_offset</i>	the offset for from the currenty buffer position
<i>the</i>	number of bits to be read

## Returns

unsigned 64 bit value representing the specified bit range

4.2.2.5 uint8\_t Buffy\_get\_bits8 ( Buffy\_t \* *this*, unsigned int *bit\_offset*, const int *no\_of\_bits* )

Get 1 - 8 bits returned in a uint8.

## Parameters

<i>this</i>	the calling buffer
<i>bit_offset</i>	the offset for from the currenty buffer position
<i>the</i>	number of bits to be read

## Returns

unsigned 8 bit value representing the specified bit range

4.2.2.6 Buffy\_t\* Buffy\_new ( Packet \* *p* )

Creates a new buffer from the given snort package.

## Parameters

<i>p</i>	the packet as defined by snort
----------	--------------------------------

## Returns

the instantiated Buffer

## 4.3 src/Profinet/Dissector-int.h File Reference

This Header describes the internal structure of the [Dissector](#) type, it defines the basic interface for operations.

### Data Structures

- struct [Dissector\\_ops](#)  
*The operations that can be called by a [Dissector](#).*
- struct [Dissector](#)  
*Used to dissect certain data ranges within a package.*

### Functions

- [Dissector\\_t](#) \* **Dissector\_new** (const struct [Dissector\\_ops](#) \*ops)

#### 4.3.1 Detailed Description

This Header describes the internal structure of the [Dissector](#) type, it defines the basic interface for operations.

## 4.4 src/Profinet/Dissector.h File Reference

The Basic [Dissector](#) abstraction (Interface).

### Typedefs

- typedef struct [Dissector](#) **Dissector\_t**

### Functions

- [Dissector\\_t](#) \* [Dissector\\_new](#) (const struct dissector\_ops \*ops)  
*Creates a new [Dissector](#) with the given operations.*
- void [Dissector\\_free](#) ([Dissector\\_t](#) \*dissector)
- [Dissector\\_t](#) \* [Dissector\\_registerSub](#) ([Dissector\\_t](#) \*this, [Dissector\\_t](#) \*subDissector)  
*Registers a given sub dissector on this dissector.*
- [Dissector\\_t](#) \* [Dissector\\_getSub](#) ([Dissector\\_t](#) \*this, uint64\_t data)  
*Returns the sub dissector that is register for the given unsigned long.*
- int [Dissector\\_dissect](#) ([Dissector\\_t](#) \*this, [Buffer\\_t](#) \*buf, [ProtocolTree\\_t](#) \*tree)  
*Dissects the package the given buffer is pointing to.*

#### 4.4.1 Detailed Description

The Basic [Dissector](#) abstraction (Interface). The Base [Dissector](#) abstraction. Every implementation of a [Dissector](#) will use and implement the operations described in this interface. [Dissector](#) are used to dissect certain ranges of data in a network package, while having the possibility to link to further dissectors when the dissection of the desired range is complete.

-> It is possible to link several Dissectors together building a tree of dissectors and subdissectors that call each other when their dissection part is completed.

#### 4.4.2 Function Documentation

##### 4.4.2.1 int Dissector\_dissect ( [Dissector\\_t](#) \* this, [Buffer\\_t](#) \* buf, [ProtocolTree\\_t](#) \* tree )

Dissects the package the given buffer is pointing to.

##### Parameters

<i>this</i>	the calling <a href="#">Dissector</a>
<i>buf</i>	the buffer pointing to the package data currently being processed
<i>tree</i>	the tree strcture to save the package data in

##### Returns

0 if the dissection was successful wihtout any failures, -1 if it was a faulty package. The fault flag will be set in the [ProtocolTree](#) accordingly

##### 4.4.2.2 void Dissector\_free ( [Dissector\\_t](#) \* dissector )

Frees the given dissector.

##### 4.4.2.3 [Dissector\\_t](#)\* Dissector\_getSub ( [Dissector\\_t](#) \* this, uint64\_t data )

Returns the sub dissector that is register for the given unsigned long.



## Parameters

<i>this</i>	the dissector calling Dissector_getSub
<i>data</i>	the value for looking up in the dissector register

## Returns

the registered sub dissector if any, NULL otherwise

## 4.4.2.4 Dissector\_t\* Dissector\_new ( const struct dissector\_ops \* ops )

Creates a new [Dissector](#) with the given operations.

This Function is the interface constructor for every [Dissector](#) implementation. Calling this function will initialize the dissector correctly and fill the needed data within the [Dissector](#) structure.

## Parameters

<i>ops</i>	the pointer to the operations used for this dissector
------------	---

## Returns

a pointer to the created dissector

## 4.4.2.5 Dissector\_t\* Dissector\_registerSub ( Dissector\_t \* this, Dissector\_t \* subDissector )

Registers a given sub dissector on this dissector.

## Parameters

<i>this</i>	the dissector to register the subDissector on
<i>subDissector</i>	the dissector to be registered as sub

## Returns

NULL if there was no other dissector registered for the given interval otherwise the existing [Dissector](#) will be overwritten and returned.

## 4.5 src/Profinet/DissectorRegister-int.h File Reference

The internal structure of a dissector register. Including the operation structure and fields.

### Data Structures

- struct [DissectorRegister\\_ops](#)  
The operations that can be called by a [DissectorRegister](#).
- struct [DissectorRegister](#)  
The datastructure for registering Dissectors on their specific intervals.

### Functions

- [Dissector\\_t](#) \* [DissectorRegister\\_new](#) (const struct [DissectorRegister\\_ops](#) \*ops)

### 4.5.1 Detailed Description

The internal structure of a dissector register. Including the operation structure and fields.

## 4.6 src/Profinet/DissectorRegister.h File Reference

The interface for dissector registers.

```
#include "Dissector.h"
```

### Typedefs

- typedef struct [DissectorRegister](#) **DissectorRegister\_t**

### Functions

- [DissectorRegister\\_t](#) \* [DissectorRegister\\_new](#) (const struct [DissectorRegister\\_ops](#) \*ops)  
*Creates a new [DissectorRegister](#) with the given operations.*
- void [DissectorRegister\\_free](#) ([DissectorRegister\\_t](#) \*this)  
*Frees the given [DissectorRegister](#).*
- [Dissector\\_t](#) \* [DissectorRegister\\_insert](#) ([DissectorRegister\\_t](#) \*this, [Dissector\\_t](#) \*dissector)  
*Inserts a new [Dissector](#).*
- [Dissector\\_t](#) \* [DissectorRegister\\_get](#) ([DissectorRegister\\_t](#) \*this, uint64\_t data)  
*Returns the [Dissector](#) that is registered for the given unsigned long.*

### 4.6.1 Detailed Description

The interface for dissector registers. The dissector register is used to register dissectors to intervals. Thereby making it possible to dissect a package while using certain data ranges for calling a next dissector that is mapped to the given data.

### 4.6.2 Function Documentation

#### 4.6.2.1 [Dissector\\_t](#)\* [DissectorRegister\\_get](#) ( [DissectorRegister\\_t](#) \* *this*, uint64\_t *data* )

Returns the [Dissector](#) that is registered for the given unsigned long.

#### Parameters

<i>this</i>	the <a href="#">DissectorRegister</a> calling
<i>data</i>	the value for looking up in the <a href="#">DissectorRegister</a>

#### Returns

the registered [Dissector](#) if any, NULL otherwise

#### 4.6.2.2 [Dissector\\_t](#)\* [DissectorRegister\\_insert](#) ( [DissectorRegister\\_t](#) \* *this*, [Dissector\\_t](#) \* *dissector* )

Inserts a new [Dissector](#).

The new dissector will be inserted into the [DissectorRegister](#) by obtaining its lower and upper identifier bounds and mapping it accordingly.

## Parameters

<i>this</i>	the calling register
<i>dissector</i>	the dissector to be inserted

## Returns

NULL if there is no previous dissector registered within its interval, otherwise overwrites the old dissector and returns it

## 4.6.2.3 DissectorRegister\_t\* DissectorRegister\_new ( const struct DissectorRegister\_ops \* ops )

Creates a new [DissectorRegister](#) with the given operations.

This Function is the interface constructor for every [DissectorRegister](#) implementation. By calling this function a new dissector register will be stored in heap memory and initialized correctly.

## Parameters

<i>ops</i>	the pointer to the operations used for this <a href="#">DissectorRegister</a>
------------	---

## Returns

a pointer to the created [DissectorRegister](#)

## 4.7 src/Profinet/PNRTDissector.c File Reference

[PNRTDissector](#) implementation.

## Data Structures

- struct [PNRTDissector](#)  
The [Dissector](#) for Profi Real Time IO 0x8892.

## Functions

- [Dissector\\_t](#) \* [PNRTDissector\\_new](#) ()
- void [PNRTDissector\\_free](#) ([Dissector\\_t](#) \*dissector)
- int [PNRTDissector\\_dissect](#) ([Dissector\\_t](#) \*this, [Buffer\\_t](#) \*buf, [ProtocolTree\\_t](#) \*tree)

## 4.7.1 Detailed Description

[PNRTDissector](#) implementation. This [Dissector](#) is the 0x8892 toplevel dissector, which will be followed by frame and block dissectors.

## 4.7.2 Function Documentation

## 4.7.2.1 int PNRTDissector\_dissect ( Dissector\_t \* this, Buffer\_t \* buf, ProtocolTree\_t \* tree )

## See Also

[Dissector\\_dissect](#)

#### 4.7.2.2 void PNRTDissector\_free ( Dissector\_t \* dissector )

See Also

[Dissector\\_free](#)

#### 4.7.2.3 Dissector\_t\* PNRTDissector\_new ( )

See Also

[Dissector\\_new](#)

## 4.8 src/Profinet/ProtocolTree-int.h File Reference

The internal sturcture of [ProtocolTree](#).

### Data Structures

- struct [ProtocolTree\\_ops](#)  
*The operations that can be called by a [ProtocolTree](#).*
- struct [ProtocolTree](#)  
*Buffer for dissecting packages in the profinet plugin.*

### Functions

- ProtocolTree\_t \* [ProtocolTree\\_new](#) (Packet \*p)

#### 4.8.1 Detailed Description

The internal sturcture of [ProtocolTree](#).

## 4.9 src/Profinet/ProtocolTree.h File Reference

The interface for [ProtocolTree](#).

### Data Structures

- struct [HeaderInfo](#)  
*Info that can be inserte into a protocol tree as new branch.*

### Functions

- struct [HeaderInfo](#) [ProtocolTree\\_new](#) ()  
*Creates a new [ProtocolTree](#).*
- void [ProtocolTree\\_free](#) (ProtocolTree\_t \*proto)  
*Frees the given [ProtocolTree](#) from memory.*
- ProtocolTree\_t \* [ProtocolTree\\_branch](#) (ProtocolTree\_t \*this, struct [HeaderInfo](#) \*info)  
*Creates a new branch with the given info field from the current root pointer of this [ProtocolTree](#).*
- ProtocolTree\_t \* [ProtocolTree\\_findBranch](#) (ProtocolTree\_t \*this, char \*caption)  
*Searches and returns the branch with the given caption.*

## Variables

- char [caption](#) [256]  
*The caption of this info field.*
- uint64\_t [bitmask](#)  
*Interesting bits that can be set.*
- char [infofield](#) [256]  
*Infofield, can contain any information in char format for specific size.*
- long long [value](#)  
*A value that can be put for information.*
- int [type](#)  
*Specifies the type of information.*

### 4.9.1 Detailed Description

The interface for [ProtocolTree](#).

### 4.9.2 Function Documentation

#### 4.9.2.1 [ProtocolTree\\_t\\*](#) [ProtocolTree\\_branch](#) ( [ProtocolTree\\_t](#) \* *this*, struct [HeaderInfo](#) \* *info* )

Creates a new branch with the given info field from the current root pointer of this [ProtocolTree](#).

##### Parameters

<i>this</i>	the calling <a href="#">ProtocolTree</a>
<i>info</i>	the header info to be inserted for the new subtree

##### Returns

A pointer to a Subtree with the newly created branch as its root pointer.

#### 4.9.2.2 [ProtocolTree\\_t\\*](#) [ProtocolTree\\_findBranch](#) ( [ProtocolTree\\_t](#) \* *this*, char \* *caption* )

Searches and returns the branch with the given caption.

##### Parameters

<i>this</i>	the calling <a href="#">ProtocolTree</a>
<i>the</i>	caption to be searched for

##### Returns

the [ProtocolTree](#) starting at the found branch, NULL if there is no such branch.

#### 4.9.2.3 void [ProtocolTree\\_free](#) ( [ProtocolTree\\_t](#) \* *proto* )

Frees the given [ProtocolTree](#) from memory.

## Parameters

<i>proto</i>	the <a href="#">ProtocolTree</a> to be freed
--------------	--

## 4.9.2.4 struct HeaderInfo ProtocolTree\_new ( )

Creates a new [ProtocolTree](#).

## Returns

the instantiated Tree

## 4.10 src/Profinet/Sender-int.h File Reference

The internal structure of [Sender](#).

## Data Structures

- struct [Sender\\_ops](#)  
*The operations that can be called by a [Sender](#).*
- struct [Sender](#)  
*[Sender](#) for sending Truffles to a specified port/socket/mq/sma.*

## Functions

- [Sender\\_t](#) \* **Sender\_new** (const struct sender\_ops \*ops)

## Variables

- struct [Sender\\_ops](#) \* **ProtocolTree\_new**

## 4.10.1 Detailed Description

The internal structure of [Sender](#).

## 4.11 src/Profinet/Sender.h File Reference

The sender interface.

## Typedefs

- typedef struct [Sender](#) **Sender\_t**

## Functions

- [Sender\\_t](#) \* **Sender\_new** (const struct sender\_ops \*ops)
- int **Sender\_free** ([Sender\\_t](#) \*sender)  
*Frees the given sender.*
- int **Sender\_send** ([Sender\\_t](#) \*this, [Truffle\\_t](#) \*truffle)

### 4.11.1 Detailed Description

The sender interface. The basic [Sender](#) abstraction. Every implementation of a [Sender](#) will use and implement the operations described in this interface. A [Sender](#) is used to send truffles to a certain port, socket, or messagequeue, depending on the implementation.

### 4.11.2 Function Documentation

#### 4.11.2.1 `int Sender_free ( Sender_t * sender )`

Frees the given sender.

##### Parameters

<i>sender</i>	the sender to be freed
---------------	------------------------

##### Returns

0 if the freeing was successful, -1 otherwise

#### 4.11.2.2 `Sender_t* Sender_new ( const struct sender_ops * ops )`

Creates a new [Dissector](#) with the given operations. This Function is the interface constructor for every [Dissector](#) implementation.

##### Parameters

<i>ops</i>	the pointer to the operations used for this dissector
------------	---

##### Returns

a pointer to the created dissector

#### 4.11.2.3 `int Sender_send ( Sender_t * this, Truffle_t * truffle )`

Sends the given truffle to the specified ipc

##### Parameters

<i>this</i>	the calling sender
<i>truffle</i>	the truffle to be send

##### Returns

0 if the sending was successful, -1 if no client is detected for receiving, or on other errors.

## 4.12 src/Profinet/Truffle.h File Reference

The structure of a [Truffle](#) that is send via ipc.

### Data Structures

- struct [EtherHeader](#)

*Houses specific information about the ether header.*

- struct [Frame](#)  
*Houses specific information about the frame.*
- struct [Truffle](#)  
*The datastructure for sending relevant information to another process.*

## Typedefs

- typedef struct [Truffle](#) **Truffle\_t**

### 4.12.1 Detailed Description

The structure of a [Truffle](#) that is send via ipc.

## 4.13 src/Profinet/UnixSocketSender.c File Reference

This file houses the operations that are specific for a [UnixSocketSender](#).

## Data Structures

- struct [UnixSocketSender](#)  
*Sends Truffles to a unix socket a client is reading from.*

## Functions

- [Sender\\_t](#) \* [UnixSocketSender\\_new](#) ()
- int [UnixSocketSender\\_free](#) ([Sender\\_t](#) \*sender)
- int [UnixSocketSender\\_send](#) ([Sender\\_t](#) \*this, [Truffle\\_t](#) \*truffle)

### 4.13.1 Detailed Description

This file houses the operations that are specific for a [UnixSocketSender](#). [UnixSocketSender](#) uses Unix sockets for sending a [Truffle](#) to a listening client.

### 4.13.2 Function Documentation

#### 4.13.2.1 int [UnixSocketSender\\_free](#) ( [Sender\\_t](#) \* sender )

See Also

[Sender\\_free](#)

#### 4.13.2.2 [Sender\\_t](#)\* [UnixSocketSender\\_new](#) ( )

See Also

[Sender\\_new](#)



4.13.2.3 `int UnixSocketSender_send ( Sender_t * this, Truffle_t * truffle )`

See Also

[Sender\\_send](#)

## 4.14 src/spp\_profinet.c File Reference

Snort Preprocessor Plugin Source File ProfiNet Purpose:

### Functions

- void [SetupProfiNet](#) ()
- void [DissectorInit](#) ()

### Variables

- [DissectorRegister\\_t](#) \* [tlRegister](#)
- [Sender\\_t](#) \* [sender](#)

#### 4.14.1 Detailed Description

Snort Preprocessor Plugin Source File ProfiNet Purpose: `$Id$` Preprocessors perform some function *once* for *each* packet. This is different from detection plugins, which are accessed depending on the standard rules. When adding a plugin to the system, be sure to add the "Setup" function to the `InitPreprocessors()` function call in `plugbase.c`!

Arguments:

This is the list of arguments that the plugin can take at the "preprocessor" line in the rules file

Effect:

What the preprocessor does. Check out some of the default ones (e.g. `spp_frag2`) for a good example of this description.

Comments:

Any comments?

#### 4.14.2 Function Documentation

4.14.2.1 `void DissectorInit ( )`

Initializes the dissectors for the profinet protocols.

4.14.2.2 `void SetupProfiNet ( )`

Registers the preprocessor keyword and initialization function into the preprocessor list. This is the function that gets called from `InitPreprocessors()` in `plugbase.c`.

#### 4.14.3 Variable Documentation

4.14.3.1 `Sender_t* sender`

The ipc sender.

#### 4.14.3.2 DissectorRegister\_t\* tlRegister

The top level dissector register.

### 4.15 src/spp\_profinet.h File Reference

Snort Preprocessor Plugin Header.

#### Functions

- void [SetupProfiNet](#) ()

#### 4.15.1 Detailed Description

Snort Preprocessor Plugin Header. This file gets included in plugbase.h when it is integrated into the rest of the program.

#### 4.15.2 Function Documentation

##### 4.15.2.1 void SetupProfiNet ( )

list of function prototypes to export for this preprocessor

Registers the preprocessor keyword and initialization function into the preprocessor list. This is the function that gets called from InitPreprocessors() in plugbase.c.

# Index

- branches
  - ProtocolTree, [15](#)
- Buffy, [5](#)
  - initialized, [5](#)
  - ops, [5](#)
  - p, [5](#)
- Buffy.h
  - Buffy\_free, [22](#)
  - Buffy\_get\_bits16, [22](#)
  - Buffy\_get\_bits32, [22](#)
  - Buffy\_get\_bits64, [22](#)
  - Buffy\_get\_bits8, [23](#)
  - Buffy\_new, [23](#)
- Buffy\_free
  - Buffy.h, [22](#)
  - Buffy\_ops, [6](#)
- Buffy\_get\_bits16
  - Buffy.h, [22](#)
  - Buffy\_ops, [6](#)
- Buffy\_get\_bits32
  - Buffy.h, [22](#)
  - Buffy\_ops, [6](#)
- Buffy\_get\_bits64
  - Buffy.h, [22](#)
  - Buffy\_ops, [7](#)
- Buffy\_get\_bits8
  - Buffy.h, [23](#)
  - Buffy\_ops, [7](#)
- Buffy\_new
  - Buffy.h, [23](#)
- Buffy\_ops, [5](#)
  - Buffy\_free, [6](#)
  - Buffy\_get\_bits16, [6](#)
  - Buffy\_get\_bits32, [6](#)
  - Buffy\_get\_bits64, [7](#)
  - Buffy\_get\_bits8, [7](#)
- calling
  - Dissector, [8](#)
- Dissector, [7](#)
  - calling, [8](#)
  - initialized, [8](#)
  - ops, [8](#)
- Dissector.h
  - Dissector\_dissect, [24](#)
  - Dissector\_free, [24](#)
  - Dissector\_getSub, [24](#)
  - Dissector\_new, [25](#)
  - Dissector\_registerSub, [25](#)
- Dissector\_dissect
  - Dissector.h, [24](#)
  - Dissector\_ops, [9](#)
- Dissector\_free
  - Dissector.h, [24](#)
  - Dissector\_ops, [9](#)
- Dissector\_getSub
  - Dissector.h, [24](#)
  - Dissector\_ops, [9](#)
- Dissector\_lower
  - Dissector\_ops, [9](#)
- Dissector\_new
  - Dissector.h, [25](#)
- Dissector\_ops, [8](#)
  - Dissector\_dissect, [9](#)
  - Dissector\_free, [9](#)
  - Dissector\_getSub, [9](#)
  - Dissector\_lower, [9](#)
  - Dissector\_registerSub, [9](#)
  - Dissector\_size, [10](#)
  - Dissector\_upper, [10](#)
- Dissector\_registerSub
  - Dissector.h, [25](#)
  - Dissector\_ops, [9](#)
- Dissector\_size
  - Dissector\_ops, [10](#)
- Dissector\_upper
  - Dissector\_ops, [10](#)
- DissectorInit
  - spp\_profinet.c, [33](#)
- DissectorRegister, [10](#)
  - initialized, [10](#)
  - ops, [10](#)
- DissectorRegister.h
  - DissectorRegister\_get, [26](#)
  - DissectorRegister\_insert, [26](#)
  - DissectorRegister\_new, [27](#)
- DissectorRegister\_get
  - DissectorRegister.h, [26](#)
  - DissectorRegister\_ops, [12](#)
- DissectorRegister\_insert
  - DissectorRegister.h, [26](#)
  - DissectorRegister\_ops, [11](#)
- DissectorRegister\_new
  - DissectorRegister.h, [27](#)
- DissectorRegister\_ops, [11](#)
  - DissectorRegister\_get, [12](#)
  - DissectorRegister\_insert, [11](#)
  - DissectorRegister\_size, [13](#)

- DissectorRegister\_size
  - DissectorRegister\_ops, 13
- EtherHeader, 13
- Frame, 13
- hInfo
  - ProtocolTree, 15
- HeaderInfo, 14
- initialized
  - Buffy, 5
  - Dissector, 8
  - DissectorRegister, 10
  - ProtocolTree, 15
  - Sender, 17
- ops
  - Buffy, 5
  - Dissector, 8
  - DissectorRegister, 10
  - ProtocolTree, 15
  - Sender, 17
- p
  - Buffy, 5
- PNRTDissector, 14
- PNRTDissector.c
  - PNRTDissector\_dissect, 27
  - PNRTDissector\_free, 27
  - PNRTDissector\_new, 28
- PNRTDissector\_dissect
  - PNRTDissector.c, 27
- PNRTDissector\_free
  - PNRTDissector.c, 27
- PNRTDissector\_new
  - PNRTDissector.c, 28
- parent
  - ProtocolTree, 15
- ProtocolTree, 15
  - branches, 15
  - hInfo, 15
  - initialized, 15
  - ops, 15
  - parent, 15
- ProtocolTree.h
  - ProtocolTree\_branch, 29
  - ProtocolTree\_findBranch, 29
  - ProtocolTree\_free, 29
  - ProtocolTree\_new, 30
- ProtocolTree\_branch
  - ProtocolTree.h, 29
  - ProtocolTree\_ops, 16
- ProtocolTree\_findBranch
  - ProtocolTree.h, 29
  - ProtocolTree\_ops, 16
- ProtocolTree\_free
  - ProtocolTree.h, 29
  - ProtocolTree\_ops, 16
- ProtocolTree\_new
  - ProtocolTree.h, 30
  - ProtocolTree\_ops, 16
- ProtocolTree\_ops, 15
  - ProtocolTree\_branch, 16
  - ProtocolTree\_findBranch, 16
  - ProtocolTree\_free, 16
  - ProtocolTree\_new, 16
- Sender, 17
  - initialized, 17
  - ops, 17
- sender
  - spp\_profinet.c, 33
  - UnixSocketSender, 19
- Sender.h
  - Sender\_free, 31
  - Sender\_new, 31
  - Sender\_send, 31
- Sender\_free
  - Sender.h, 31
  - Sender\_ops, 18
- Sender\_new
  - Sender.h, 31
- Sender\_ops, 17
  - Sender\_free, 18
  - Sender\_send, 18
- Sender\_send
  - Sender.h, 31
  - Sender\_ops, 18
- SetupProfiNet
  - spp\_profinet.c, 33
  - spp\_profinet.h, 34
- spp\_profinet.c
  - DissectorInit, 33
  - sender, 33
  - SetupProfiNet, 33
  - tlRegister, 33
- spp\_profinet.h
  - SetupProfiNet, 34
- src/Profinet/Buffy-int.h, 21
- src/Profinet/Buffy.h, 21
- src/Profinet/Dissector-int.h, 23
- src/Profinet/Dissector.h, 24
- src/Profinet/DissectorRegister-int.h, 25
- src/Profinet/DissectorRegister.h, 26
- src/Profinet/PNRTDissector.c, 27
- src/Profinet/ProtocolTree-int.h, 28
- src/Profinet/ProtocolTree.h, 28
- src/Profinet/Sender-int.h, 30
- src/Profinet/Sender.h, 30
- src/Profinet/Truffle.h, 31
- src/Profinet/UnixSocketSender.c, 32
- src/spp\_profinet.c, 33
- src/spp\_profinet.h, 34
- tlRegister
  - spp\_profinet.c, 33
- Truffle, 18

UnixSocketSender, [19](#)  
    sender, [19](#)  
UnixSocketSender.c  
    UnixSocketSender\_free, [32](#)  
    UnixSocketSender\_new, [32](#)  
    UnixSocketSender\_send, [32](#)  
UnixSocketSender\_free  
    UnixSocketSender.c, [32](#)  
UnixSocketSender\_new  
    UnixSocketSender.c, [32](#)  
UnixSocketSender\_send  
    UnixSocketSender.c, [32](#)