

spp_profinet

Generated by Doxygen 1.8.10

Sat Jan 30 2016 13:00:17

Contents

1	Data Structure Index	1
1.1	Data Structures	1
2	File Index	3
2.1	File List	3
3	Data Structure Documentation	5
3.1	BoxStruct_struct Struct Reference	5
3.1.1	Detailed Description	5
3.1.2	Field Documentation	5
3.1.2.1	a	5
3.1.2.2	b	5
3.1.2.3	c	5
3.2	Dissector Struct Reference	6
3.2.1	Field Documentation	6
3.2.1.1	calling	6
3.2.1.2	initialized	6
3.2.1.3	ops	6
3.3	Dissector_ops Struct Reference	6
3.4	PNRTDissector Struct Reference	6
4	File Documentation	9
4.1	src/doxxygen_c.h File Reference	9
4.1.1	Detailed Description	10
4.1.2	Typedef Documentation	10
4.1.2.1	BoxEnum	10
4.1.2.2	BoxStruct	10
4.1.3	Enumeration Type Documentation	10
4.1.3.1	BoxEnum_enum	10
4.1.4	Function Documentation	10
4.1.4.1	Box_The_Function_Name(BoxParamType1 param1, BoxParamType2 param2)	10
4.1.4.2	Box_The_Last_One(void)	11

4.1.4.3	Box_The_Second_Function(void)	11
4.1.4.4	DissectorRegister_free(DissectorRegister_t *this)	12
4.2	src/Profinet/Buffy.h File Reference	12
4.2.1	Detailed Description	12
4.2.2	Function Documentation	12
4.2.2.1	Buffy_get_bits16(Buffy_t *buffy, unsigned int bit_offset, const int no_of_bits, const unsigned int encoding)	12
4.2.2.2	Buffy_get_bits32(Buffy_t *buffy, unsigned int bit_offset, const int no_of_bits, const unsigned int encoding)	12
4.2.2.3	Buffy_get_bits64(Buffy_t *buffy, unsigned int bit_offset, const int no_of_bits, const unsigned int encoding)	12
4.2.2.4	Buffy_get_bits8(Buffy_t *buffy, unsigned int bit_offset, const int no_of_bits)	13
4.3	src/Profinet/Dissector-int.h File Reference	13
4.3.1	Detailed Description	13
4.3.2	Variable Documentation	13
4.3.2.1	calling	13
4.3.2.2	initialized	14
4.3.2.3	ops	14
4.4	src/Profinet/Dissector.h File Reference	14
4.4.1	Detailed Description	14
4.4.2	Function Documentation	14
4.4.2.1	Dissector_dissect(Dissector_t *this, Buffer_t *buf, ProtocolTree_t *tree)	14
4.4.2.2	Dissector_free(Dissector_t *dissector)	14
4.4.2.3	Dissector_getSub(Dissector_t *this, uint64_t data)	15
4.4.2.4	Dissector_new(const struct dissector_ops *ops)	16
4.4.2.5	Dissector_registerSub(Dissector_t *this, Dissector_t *subDissector)	16
4.5	src/Profinet/DissectorRegister.h File Reference	16
4.5.1	Detailed Description	16
4.5.2	Function Documentation	17
4.5.2.1	DissectorRegister_free(DissectorRegister_t *this)	17
4.5.2.2	DissectorRegister_get(DissectorRegister_t *this, uint64_t data)	17
4.5.2.3	DissectorRegister_insert(DissectorRegister_t *this, DissectorRegister_t *dissector)	17
4.5.2.4	DissectorRegister_new(const struct DissectorRegister_ops *ops)	17
4.6	src/Profinet/Sender.h File Reference	17
4.6.1	Detailed Description	18
4.6.2	Function Documentation	18
4.6.2.1	Sender_free(Sender_t *dissector)	18
4.6.2.2	Sender_new(const struct sender_ops *ops)	18
4.6.2.3	Sender_send(Sender_t *this, Truffle_t *truffle)	18
4.7	src/Profinet/Truffle.h File Reference	18
4.7.1	Detailed Description	19

4.7.2	Function Documentation	19
4.7.2.1	Truffle_get(Truffle_t *this, char *key)	19
4.7.2.2	Truffle_new()	19
4.7.2.3	Truffle_put(Truffle_t *this, char *key, Entry_t entry)	19
4.8	src/spp_profinet.c File Reference	19
4.8.1	Detailed Description	19
4.8.2	Function Documentation	20
4.8.2.1	DissectorInit()	20
4.8.2.2	SetupProfiNet()	20
4.8.3	Variable Documentation	20
4.8.3.1	sender	20
4.8.3.2	tlRegister	20
4.9	src/spp_profinet.h File Reference	20
4.9.1	Detailed Description	20
4.9.2	Function Documentation	20
4.9.2.1	SetupProfiNet()	20
	Index	23

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

BoxStruct_struct	
Use brief, otherwise the index won't have a brief explanation	5
Dissector	6
Dissector_ops	6
PNRTDissector	6

Chapter 2

File Index

2.1 File List

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

src/ doxygen_c.h	
File containing example of doxygen usage for quick reference	9
src/ spp_profinet.c	19
src/ spp_profinet.h	20
src/Profinet/ Buffy.h	12
src/Profinet/ Dissector-int.h	13
src/Profinet/ Dissector.h	14
src/Profinet/ DissectorRegister-int.h	??
src/Profinet/ DissectorRegister.h	16
src/Profinet/ Entry.h	??
src/Profinet/ Sender.h	17
src/Profinet/ Truffle.h	18

Chapter 3

Data Structure Documentation

3.1 BoxStruct_struct Struct Reference

Use brief, otherwise the index won't have a brief explanation.

```
#include <doxygen_c.h>
```

Data Fields

- int [a](#)
- int [b](#)
- double [c](#)

3.1.1 Detailed Description

Use brief, otherwise the index won't have a brief explanation.

Detailed explanation.

3.1.2 Field Documentation

3.1.2.1 int BoxStruct_struct::a

Some documentation for the member [BoxStruct::a](#).

3.1.2.2 int BoxStruct_struct::b

Some documentation for the member [BoxStruct::b](#).

3.1.2.3 double BoxStruct_struct::c

Etc.

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

- [src/doxygen_c.h](#)

3.2 Dissector Struct Reference

Data Fields

- bool [initialized](#)
- const struct [Dissector_ops](#) * [ops](#)
- [Dissector_t](#) * [calling](#)

3.2.1 Field Documentation

3.2.1.1 [Dissector_t](#)* [Dissector::calling](#)

the dissector this dissector has been called from

3.2.1.2 bool [Dissector::initialized](#)

whether this dissector was initialized

3.2.1.3 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](#)

3.3 Dissector_ops Struct Reference

Data Fields

- size_t [Dissector_size](#)
- unit64_t [lower](#)
- uint64_t [upper](#)
- void(* [Dissector_free](#))([Dissector_t](#) *dissector)
- [Dissector_t](#) *(* [Dissector_registerSub](#))([Dissector_t](#) *this, [Dissector_t](#) *subDissector, Interval interval)
- [Dissector_t](#) *(* [Dissector_getSub_64](#))([Dissector_t](#) *this, uint64_t data)
- [Dissector_t](#) *(* [Dissector_getSub_32](#))([Dissector_t](#) *this, uint32_t data)
- [Dissector_t](#) *(* [Dissector_getSub_16](#))([Dissector_t](#) *this, uint16_t data)
- [Dissector_t](#) *(* [Dissector_getSub_8](#))([Dissector_t](#) *this, uint8_t data)
- [Dissector_t](#) *(* [Dissector_getSub](#))([Dissector_t](#) *this, void *data, int len)
- int(* [Dissector_dissect](#))([Dissector_t](#) *this, Buffer_t *buf, ProtocolTree_t *tree)

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

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

3.4 PNRTDissector Struct Reference

Data Fields

- struct [Dissector](#) [dissector](#)

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

- `src/Profinet/PNRTDissector.c`

Chapter 4

File Documentation

4.1 src/doxygen_c.h File Reference

File containing example of doxygen usage for quick reference.

```
#include <systemheader1.h>
#include <systemheader2.h>
#include <box/header1.h>
#include <box/header2.h>
#include "local_header1.h"
#include "local_header2.h"
```

Data Structures

- struct [BoxStruct_struct](#)

Use brief, otherwise the index won't have a brief explanation.

Typedefs

- typedef enum [BoxEnum_enum](#) BoxEnum
- typedef struct [BoxStruct_struct](#) BoxStruct

Use brief, otherwise the index won't have a brief explanation.

Enumerations

- enum [BoxEnum_enum](#) { [BOXENUM_FIRST](#), [BOXENUM_SECOND](#), [BOXENUM_ETC](#) }

Functions

- BOXEXPORT [BoxStruct](#) * [Box_The_Function_Name](#) (BoxParamType1 param1, BoxParamType2 param2)

Example showing how to document a function with Doxygen.

- BOXEXPORT void * [Box_The_Second_Function](#) (void)

A simple stub function to show how links do work.

- BOXEXPORT void [Box_The_Last_One](#) (void)
- void [DissectorRegister_free](#) (DissectorRegister_t *this)

4.1.1 Detailed Description

File containing example of doxygen usage for quick reference.

Author

My Self

Date

9 Sep 2012 Here typically goes a more extensive explanation of what the header defines. Doxygens tags are words preceeded by either a backslash \ or by an at symbol @.

See also

<http://www.stack.nl/~dimitri/doxygen/docblocks.html>
<http://www.stack.nl/~dimitri/doxygen/commands.html>

4.1.2 Typedef Documentation

4.1.2.1 typedef enum **BoxEnum_enum** **BoxEnum**

brief Use brief, otherwise the index won't have a brief explanation.

Detailed explanation.

4.1.2.2 typedef struct **BoxStruct_struct** **BoxStruct**

Use brief, otherwise the index won't have a brief explanation.

Detailed explanation.

4.1.3 Enumeration Type Documentation

4.1.3.1 enum **BoxEnum_enum**

brief Use brief, otherwise the index won't have a brief explanation.

Detailed explanation.

Enumerator

BOXENUM_FIRST Some documentation for first.
BOXENUM_SECOND Some documentation for second.
BOXENUM_ETC Etc.

4.1.4 Function Documentation

4.1.4.1 **BOXEXPORT** **BoxStruct*** **Box_The_Function_Name** (**BoxParamType1** *param1*, **BoxParamType2** *param2*)

Example showing how to document a function with Doxygen.

Description of what the function does. This part may refer to the parameters of the function, like `param1` or `param2`. A word of code can also be inserted like `this` which is equivalent to `this` and can be useful to say that the function returns a `void` or an `int`. If you want to have more than one word in typewriter font, then just use `<tt>`. We can also include text verbatim,

like `this`

Sometimes it is also convenient to include an example of usage:

```
1 BoxStruct *out = Box_The_Function_Name(param1, param2);
2 printf("something...\n");
```

Or,

```
1 pyval = python_func(arg1, arg2)
2 print pyval
```

when the language is not the one used in the current source file (but **be careful** as this may be supported only by recent versions of Doxygen). By the way, **this is how you write bold text** or, if it is just one word, then you can just do **this**.

Parameters

<i>param1</i>	Description of the first parameter of the function.
<i>param2</i>	The second one, which follows <i>param1</i> .

Returns

Describe what the function returns.

See also

[Box_The_Second_Function](#)
[Box_The_Last_One](#)
<http://website/>

Note

Something to note.

Warning

Warning.

4.1.4.2 BOXEXPORT void Box_The_Last_One (void)

Brief can be omitted. If you configure Doxygen with JAVADOC_AUTOBRIEF=YES, then the first Line of the comment is used instead. In this function this would be as if

```
@brief Brief can be omitted.
```

was used instead.

4.1.4.3 BOXEXPORT void* Box_The_Second_Function (void)

A simple stub function to show how links do work.

Links are generated automatically for webpages (like <http://www.google.co.uk>) and for structures, like [BoxStruct_struct](#). For typedef-ed types use [BoxStruct](#). For functions, automatic links are generated when the parenthesis () follow the name of the function, like [Box_The_Function_Name\(\)](#). Alternatively, you can use [Box_↔The_Function_Name](#).

Returns

NULL is always returned.

4.1.4.4 void DissectorRegister_free (DissectorRegister_t * *this*)

Frees the given DissectorRegister.

4.2 src/Profinet/Buffy.h File Reference

Functions

- uint8_t [Buffy_get_bits8](#) (Buffy_t *buffy, unsigned int bit_offset, const int no_of_bits)
- uint16_t [Buffy_get_bits16](#) (Buffy_t *buffy, unsigned int bit_offset, const int no_of_bits, const unsigned int encoding)
- uint32_t [Buffy_get_bits32](#) (Buffy_t *buffy, unsigned int bit_offset, const int no_of_bits, const unsigned int encoding)
- uint64_t [Buffy_get_bits64](#) (Buffy_t *buffy, unsigned int bit_offset, const int no_of_bits, const unsigned int encoding)

4.2.1 Detailed Description

Snort Preprocessor Plugin Header

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

4.2.2 Function Documentation

4.2.2.1 uint16_t Buffy_get_bits16 (Buffy_t * *buffy*, unsigned int *bit_offset*, const int *no_of_bits*, const unsigned int *encoding*)

Get 1 - 16 bits returned in a uint16.

Parameters

<i>buffy</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

4.2.2.2 uint32_t Buffy_get_bits32 (Buffy_t * *buffy*, unsigned int *bit_offset*, const int *no_of_bits*, const unsigned int *encoding*)

Get 1 - 32 bits returned in a uint32.

Parameters

<i>buffy</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

4.2.2.3 uint64_t Buffy_get_bits64 (Buffy_t * *buffy*, unsigned int *bit_offset*, const int *no_of_bits*, const unsigned int *encoding*)

Get 1 - 64 bits returned in a uint64.

Parameters

<i>buffy</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

4.2.2.4 `uint8_t Buffy_get_bits8 (Buffy_t * buffy, unsigned int bit_offset, const int no_of_bits)`

Get 1 - 8 bits returned in a uint8.

Parameters

<i>buffy</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

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

Data Structures

- struct [Dissector_ops](#)
- struct [Dissector](#)

Functions

- struct [Dissector](#) **Dissector_new** (const struct [Dissector_ops](#) *ops)

Variables

- bool [initialized](#)
- const struct [Dissector_ops](#) * [ops](#)
- [Dissector_t](#) * [calling](#)

4.3.1 Detailed Description

This Header discribes the internal structure of the [Dissector](#) type. It defines the basic interface for operations.

4.3.2 Variable Documentation

4.3.2.1 [Dissector_t](#)* calling

the dissector this dissector has been called from

4.3.2.2 bool initialized

whether this dissector was initialized

4.3.2.3 const struct Dissector_ops* ops

the dissectors operations

4.4 src/Profinet/Dissector.h File Reference

Typedefs

- typedef struct [Dissector](#) **Dissector_t**

Functions

- [Dissector_t](#) * [Dissector_new](#) (const struct dissector_ops *ops)
- void [Dissector_free](#) ([Dissector_t](#) *dissector)
- [Dissector_t](#) * [Dissector_registerSub](#) ([Dissector_t](#) *this, [Dissector_t](#) *subDissector)
- [Dissector_t](#) * [Dissector_getSub](#) ([Dissector_t](#) *this, uint64_t data)
- int [Dissector_dissect](#) ([Dissector_t](#) *this, Buffer_t *buf, ProtocolTree_t *tree)

4.4.1 Detailed Description

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 Dissector
<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.

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 [Dissector](#) will be overwritten and returned.

4.5 src/Profinet/DissectorRegister.h File Reference**Typedefs**

- typedef struct DissectorRegister **DissectorRegister_t**

Functions

- DissectorRegister_t * [DissectorRegister_new](#) (const struct DissectorRegister_ops *ops)
- void [DissectorRegister_free](#) (DissectorRegister_t *this)
- Dissector_t * [DissectorRegister_insert](#) (DissectorRegister_t *this, DissectorRegister_t *dissector)
- Dissector_t * [DissectorRegister_get](#) (DissectorRegister_t *this, uint64_t data)

4.5.1 Detailed Description

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.

4.5.2 Function Documentation

4.5.2.1 void DissectorRegister_free (DissectorRegister_t * *this*)

Frees the given DissectorRegister.

4.5.2.2 Dissector_t* DissectorRegister_get (DissectorRegister_t * *this*, uint64_t *data*)

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

Parameters

<i>this</i>	the DissectorRegister calling
<i>data</i>	the value for looking up in the DissectorRegister

Returns

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

4.5.2.3 Dissector_t* DissectorRegister_insert (DissectorRegister_t * *this*, DissectorRegister_t * *dissector*)

Inserts a new [Dissector](#) for the given interval. If a dissector is already mapped

Parameters

<i>this</i>	the calling register
<i>the</i>	interval the given dissector will be mapped to
<i>dissector</i>	the dissector to be inserted

Returns

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

4.5.2.4 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.

Parameters

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

Returns

a pointer to the created DissectorRegister

4.6 src/Profinet/Sender.h File Reference

Typedefs

- typedef struct Sender **Sender_t**

Functions

- `Sender_t * Sender_new` (const struct sender_ops *ops)
- void `Sender_free` (Sender_t *dissector)
- int `Sender_send` (Sender_t *this, Truffle_t *truffle)

4.6.1 Detailed Description

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 implemented send operation.

-> 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.

4.6.2 Function Documentation

4.6.2.1 void Sender_free (Sender_t * *dissector*)

Frees the given dissector.

4.6.2.2 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

4.6.2.3 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.

4.7 src/Profinet/Truffle.h File Reference

Typedefs

- typedef struct Truffle **Truffle_t**

Functions

- `Truffle_t * Truffle_new ()`
- `Entry_t * Truffle_get (Truffle_t *this, char *key)`
- `Entry_t * Truffle_put (Truffle_t *this, char *key, Entry_t entry)`

4.7.1 Detailed Description

Truffle header file

4.7.2 Function Documentation

4.7.2.1 `Entry_t* Truffle_get (Truffle_t * this, char * key)`

Returns the entry that is mapped to the given key in this truffle.

Parameters

<i>this</i>	the calling truffle
<i>key</i>	the key to be looking for

4.7.2.2 `Truffle_t* Truffle_new ()`

Creates a new truffle that can be filled with data from the package.

4.7.2.3 `Entry_t* Truffle_put (Truffle_t * this, char * key, Entry_t entry)`

Inserts the given entry into the truffle mapped to the key.

Parameters

<i>this</i>	the calling truffle
<i>key</i>	the value to map the entry to
<i>entry</i>	the entry to be put into the truffle

4.8 src/spp_profinet.c File Reference

Functions

- void `SetupProfiNet ()`
- void `DissectorInit ()`

Variables

- `DissectorRegister_t * tlRegister`
- `Sender_t * sender`

4.8.1 Detailed Description

\$Id\$ Snort Preprocessor Plugin Source File ProfiNet Purpose:

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?

4.8.2 Function Documentation

4.8.2.1 void DissectorInit ()

Initializes the dissectors for the profinet protocols.

4.8.2.2 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.

4.8.3 Variable Documentation

4.8.3.1 Sender_t* sender

The ipc sender.

4.8.3.2 DissectorRegister_t* tlRegister

The top level dissector register.

4.9 src/spp_profinet.h File Reference

Functions

- void [SetupProfiNet](#) ()

4.9.1 Detailed Description

Snort Preprocessor Plugin Header

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

4.9.2 Function Documentation

4.9.2.1 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

a

BoxStruct_struct, 5

b

BoxStruct_struct, 5

BOXENUM_ETC

doxygen_c.h, 10

BOXENUM_FIRST

doxygen_c.h, 10

BOXENUM_SECOND

doxygen_c.h, 10

Box_The_Function_Name

doxygen_c.h, 10

Box_The_Last_One

doxygen_c.h, 11

Box_The_Second_Function

doxygen_c.h, 11

BoxEnum

doxygen_c.h, 10

BoxEnum_enum

doxygen_c.h, 10

BoxStruct

doxygen_c.h, 10

BoxStruct_struct, 5

a, 5

b, 5

c, 5

Buffy.h

Buffy_get_bits16, 12

Buffy_get_bits32, 12

Buffy_get_bits64, 12

Buffy_get_bits8, 13

Buffy_get_bits16

Buffy.h, 12

Buffy_get_bits32

Buffy.h, 12

Buffy_get_bits64

Buffy.h, 12

Buffy_get_bits8

Buffy.h, 13

c

BoxStruct_struct, 5

calling

Dissector, 6

Dissector-int.h, 13

Dissector, 6

calling, 6

initialized, 6

ops, 6

Dissector-int.h

calling, 13

initialized, 13

ops, 14

Dissector.h

Dissector_dissect, 14

Dissector_free, 14

Dissector_getSub, 14

Dissector_new, 16

Dissector_registerSub, 16

Dissector_dissect

Dissector.h, 14

Dissector_free

Dissector.h, 14

Dissector_getSub

Dissector.h, 14

Dissector_new

Dissector.h, 16

Dissector_ops, 6

Dissector_registerSub

Dissector.h, 16

DissectorInit

spp_profinet.c, 20

DissectorRegister.h

DissectorRegister_free, 17

DissectorRegister_get, 17

DissectorRegister_insert, 17

DissectorRegister_new, 17

DissectorRegister_free

DissectorRegister.h, 17

doxygen_c.h, 11

DissectorRegister_get

DissectorRegister.h, 17

DissectorRegister_insert

DissectorRegister.h, 17

DissectorRegister_new

DissectorRegister.h, 17

doxygen_c.h

BOXENUM_ETC, 10

BOXENUM_FIRST, 10

BOXENUM_SECOND, 10

Box_The_Function_Name, 10

Box_The_Last_One, 11

Box_The_Second_Function, 11

BoxEnum, 10

BoxEnum_enum, 10

BoxStruct, 10

DissectorRegister_free, 11

- initialized
 - Dissector, [6](#)
 - Dissector-int.h, [13](#)
- ops
 - Dissector, [6](#)
 - Dissector-int.h, [14](#)
- PNRTDissector, [6](#)
- sender
 - spp_profinet.c, [20](#)
- Sender.h
 - Sender_free, [18](#)
 - Sender_new, [18](#)
 - Sender_send, [18](#)
- Sender_free
 - Sender.h, [18](#)
- Sender_new
 - Sender.h, [18](#)
- Sender_send
 - Sender.h, [18](#)
- SetupProfiNet
 - spp_profinet.c, [20](#)
 - spp_profinet.h, [20](#)
- spp_profinet.c
 - DissectorInit, [20](#)
 - sender, [20](#)
 - SetupProfiNet, [20](#)
 - tlRegister, [20](#)
- spp_profinet.h
 - SetupProfiNet, [20](#)
- src/Profinet/Buffy.h, [12](#)
- src/Profinet/Dissector-int.h, [13](#)
- src/Profinet/Dissector.h, [14](#)
- src/Profinet/DissectorRegister.h, [16](#)
- src/Profinet/Sender.h, [17](#)
- src/Profinet/Truffle.h, [18](#)
- src/doxygen_c.h, [9](#)
- src/spp_profinet.c, [19](#)
- src/spp_profinet.h, [20](#)
- tlRegister
 - spp_profinet.c, [20](#)
- Truffle.h
 - Truffle_get, [19](#)
 - Truffle_new, [19](#)
 - Truffle_put, [19](#)
- Truffle_get
 - Truffle.h, [19](#)
- Truffle_new
 - Truffle.h, [19](#)
- Truffle_put
 - Truffle.h, [19](#)