



spp_profinet Class Documentation

Brendl, Julian `julian.brendl@student.kit.edu`

Diez, Maximilian `maximilian.diez@student.kit.edu`

Giraud, Mark `mark.giraud@student.kit.edu`

Hermes, Jan `jan.hermes@student.kit.edu`

Höhler, Dimitri `dimitri.hoehler@student.kit.edu`

Kiechle, Valentin `valentin.kiechle@student.kit.edu`

February 1, 2016

Contents

1	Data Structure Index	1
1.1	Data Structures	1
2	File Index	2
2.1	File List	2
3	Data Structure Documentation	3
3.1	Buffy Struct Reference	3
3.1.1	Detailed Description	3
3.1.2	Field Documentation	3
3.1.2.1	initialized	3
3.1.2.2	ops	3
3.1.2.3	p	3
3.2	Buffy_ops Struct Reference	3
3.2.1	Detailed Description	4
3.2.2	Field Documentation	4
3.2.2.1	Buffy_free	4
3.2.2.2	Buffy_get_bits16	4
3.2.2.3	Buffy_get_bits32	4
3.2.2.4	Buffy_get_bits64	5
3.2.2.5	Buffy_get_bits8	5
3.3	Dissector Struct Reference	5
3.3.1	Detailed Description	6
3.3.2	Field Documentation	6
3.3.2.1	calling	6
3.3.2.2	initialized	6
3.3.2.3	ops	6
3.4	Dissector_ops Struct Reference	6
3.4.1	Detailed Description	7
3.4.2	Field Documentation	7
3.4.2.1	Dissector_dissect	7
3.4.2.2	Dissector_free	7
3.4.2.3	Dissector_getSub	7
3.4.2.4	Dissector_lower	7
3.4.2.5	Dissector_registerSub	8
3.4.2.6	Dissector_size	8
3.4.2.7	Dissector_upper	8
3.5	DissectorRegister Struct Reference	8
3.5.1	Detailed Description	8
3.5.2	Field Documentation	9
3.5.2.1	initialized	9
3.5.2.2	ops	9
3.6	DissectorRegister_ops Struct Reference	9
3.6.1	Detailed Description	9
3.6.2	Member Function Documentation	9
3.6.2.1	DissectorRegister_insert(DissectorRegister_t *this, Dissector_↵ t *dissector)	9
3.6.3	Field Documentation	10

3.6.3.1	DissectorRegister_get	10
3.6.3.2	DissectorRegister_size	10
3.7	EtherHeader Struct Reference	10
3.7.1	Detailed Description	11
3.8	Frame Struct Reference	11
3.8.1	Detailed Description	11
3.9	HeaderInfo Struct Reference	11
3.9.1	Detailed Description	12
3.10	PNRTDissector Struct Reference	12
3.11	ProtocolTree Struct Reference	12
3.11.1	Detailed Description	12
3.11.2	Field Documentation	12
3.11.2.1	branches	12
3.11.2.2	hInfo	12
3.11.2.3	initialized	13
3.11.2.4	ops	13
3.11.2.5	parent	13
3.12	ProtocolTree_ops Struct Reference	13
3.12.1	Detailed Description	13
3.12.2	Field Documentation	13
3.12.2.1	ProtocolTree_branch	13
3.12.2.2	ProtocolTree_findBranch	14
3.12.2.3	ProtocolTree_free	14
3.12.2.4	ProtocolTree_new	14
3.13	Sender Struct Reference	14
3.13.1	Detailed Description	15
3.13.2	Field Documentation	15
3.13.2.1	initialized	15
3.13.2.2	ops	15
3.14	Sender_ops Struct Reference	15
3.14.1	Detailed Description	15
3.14.2	Field Documentation	15
3.14.2.1	Sender_free	15
3.14.2.2	Sender_send	16
3.15	Truffle Struct Reference	17
3.15.1	Detailed Description	17
3.16	UnixSocketSender Struct Reference	17
3.16.1	Detailed Description	18
3.16.2	Field Documentation	18
3.16.2.1	sender	18
4	File Documentation	19
4.1	src/Profinet/Buffy-int.h File Reference	19
4.1.1	Detailed Description	19
4.2	src/Profinet/Buffy.h File Reference	19
4.2.1	Detailed Description	20
4.2.2	Function Documentation	20
4.2.2.1	Buffy_free(Buffy_t *buffy)	20
4.2.2.2	Buffy_get_bits16(Buffy_t *this, unsigned int bit_offset, const int no_↔ of_bits, const unsigned int encoding)	20
4.2.2.3	Buffy_get_bits32(Buffy_t *this, unsigned int bit_offset, const int no_↔ of_bits, const unsigned int encoding)	20

4.2.2.4	Buffy_get_bits64(Buffy_t *this, unsigned int bit_offset, const int no_of_bits, const unsigned int encoding)	20
4.2.2.5	Buffy_get_bits8(Buffy_t *this, unsigned int bit_offset, const int no_of_bits)	21
4.2.2.6	Buffy_new(Packet *p)	21
4.3	src/Profinet/Dissector-int.h File Reference	21
4.3.1	Detailed Description	22
4.4	src/Profinet/Dissector.h File Reference	22
4.4.1	Detailed Description	22
4.4.2	Function Documentation	22
4.4.2.1	Dissector_dissect(Dissector_t *this, Buffer_t *buf, ProtocolTree_t *tree)	22
4.4.2.2	Dissector_free(Dissector_t *dissector)	23
4.4.2.3	Dissector_getSub(Dissector_t *this, uint64_t data)	23
4.4.2.4	Dissector_new(const struct dissector_ops *ops)	23
4.4.2.5	Dissector_registerSub(Dissector_t *this, Dissector_t *subDissector)	23
4.5	src/Profinet/DissectorRegister-int.h File Reference	24
4.5.1	Detailed Description	24
4.6	src/Profinet/DissectorRegister.h File Reference	24
4.6.1	Detailed Description	25
4.6.2	Function Documentation	25
4.6.2.1	DissectorRegister_get(DissectorRegister_t *this, uint64_t data)	25
4.6.2.2	DissectorRegister_insert(DissectorRegister_t *this, Dissector_t *dissector)	25
4.6.2.3	DissectorRegister_new(const struct DissectorRegister_ops *ops)	25
4.7	src/Profinet/PNRTDissector.c File Reference	26
4.7.1	Detailed Description	26
4.7.2	Function Documentation	26
4.7.2.1	PNRTDissector_dissect(Dissector_t *this, Buffer_t *buf, ProtocolTree_t *tree)	26
4.7.2.2	PNRTDissector_free(Dissector_t *dissector)	26
4.7.2.3	PNRTDissector_new()	26
4.8	src/Profinet/ProtocolTree-int.h File Reference	27
4.8.1	Detailed Description	27
4.9	src/Profinet/ProtocolTree.h File Reference	27
4.9.1	Detailed Description	28
4.9.2	Function Documentation	28
4.9.2.1	ProtocolTree_branch(ProtocolTree_t *this, struct HeaderInfo *info)	28
4.9.2.2	ProtocolTree_findBranch(ProtocolTree_t *this, char *caption)	28
4.9.2.3	ProtocolTree_free(ProtocolTree_t *proto)	28
4.9.2.4	ProtocolTree_new()	29
4.10	src/Profinet/Sender-int.h File Reference	29
4.10.1	Detailed Description	29
4.11	src/Profinet/Sender.h File Reference	29
4.11.1	Detailed Description	30
4.11.2	Function Documentation	30
4.11.2.1	Sender_free(Sender_t *sender)	30
4.11.2.2	Sender_new(const struct sender_ops *ops)	30
4.11.2.3	Sender_send(Sender_t *this, Truffle_t *truffle)	30
4.12	src/Profinet/UnixSocketSender.c File Reference	31
4.12.1	Detailed Description	31
4.12.2	Function Documentation	31
4.12.2.1	UnixSocketSender_free(Sender_t *sender)	31

4.12.2.2	UnixSocketSender_new()	31
4.12.2.3	UnixSocketSender_send(Sender_t *this, Truffle_t *truffle)	31
4.13	src/spp_profinet.c File Reference	32
4.13.1	Detailed Description	32
4.13.2	Function Documentation	32
4.13.2.1	DissectorInit()	32
4.13.2.2	SetupProfiNet()	32
4.13.3	Variable Documentation	32
4.13.3.1	sender	32
4.13.3.2	tlRegister	33
4.14	src/spp_profinet.h File Reference	33
4.14.1	Detailed Description	33
4.14.2	Function Documentation	33
4.14.2.1	SetupProfiNet()	33
Index		34

1Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Buffy	Buffer for dissecting packages in the profinet plugin	3
Buffy_ops	The operations that can be called by a Buffy buffer	3
Dissector	Used to dissect certain data ranges within a package	5
Dissector_ops	The operations that can be called by a Dissector	6
DissectorRegister	The datastructure for registering Dissectors on their specific intervals	8
DissectorRegister_ops	The operations that can be called by a DissectorRegister	9
EtherHeader	Houses specific information about the ether header	10
Frame	Houses specific information about the frame	11
HeaderInfo	Info that can be inserte into a protocol tree as new branch	11
PNRTDissector	12
ProtocolTree	Buffer for dissecting packages in the profinet plugin	12
ProtocolTree_ops	The operations that can be called by a ProtocolTree	13
Sender	Sender for sending Truffles to a specified port/socket/mq/sma	14
Sender_ops	The operations that can be called by a Sender	15
Truffle	The datastructure for sending relevant information to another process	17
UnixSocketSender	Sends Truffles to a unix socket a client is reading from	17

2File Index

2.1 File List

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

src/spp_profinet.c	32
src/spp_profinet.h	33
src/Profinet/Buffy-int.h	
The interface for Buffy	19
src/Profinet/Buffy.h	
The interface for Buffy	19
src/Profinet/Dissector-int.h	
This Header describes the internal structure of the Dissector type, it defines the basic interface for operations	21
src/Profinet/Dissector.h	
The interface for dissectors	22
src/Profinet/DissectorRegister-int.h	
The internal structure of a dissector register. Including the operation structure and fields	24
src/Profinet/DissectorRegister.h	
The interface for dissector registers	24
src/Profinet/PNRTDissector.c	
This file houses the operations that are specific for a UnixSocketSender	26
src/Profinet/ProtocolTree-int.h	
The internal sturcture of ProtocolTree	27
src/Profinet/ProtocolTree.h	
The interface for ProtocolTree	27
src/Profinet/Sender-int.h	
The internal functionality of Sender	29
src/Profinet/Sender.h	
The sender interface	29
src/Profinet/Truffle.h	??
src/Profinet/UnixSocketSender.c	
This file houses the operations that are specific for a UnixSocketSender	31

3Data 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 <code>Dissector</code>
<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 `unit64_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 DissectorRegister calling
<i>data</i>	the value for looking up in the DissectorRegister

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

Data Fields

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

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 ProtocolTree
<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 ProtocolTree
<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 ProtocolTree 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](#)

4File Documentation

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

The interface for [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 interface for [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](#) [Bufgy_get_bits32](#) ([Bufgy_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](#) [Bufgy_get_bits64](#) ([Bufgy_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 interface for dissectors.

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 interface for dissectors.

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

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 DissectorRegister
------------	---

Returns

a pointer to the created [DissectorRegister](#)

4.7 src/Profinet/PNRTDissector.c File Reference

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

Data Structures

- struct [PNRTDissector](#)

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

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

[UnixSocketSender](#) uses Unix sockets for sending a [Truffle](#) to a listening client.

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 <code>ProtocolTree</code>
<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 <code>ProtocolTree</code>
<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 ProtocolTree 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 functionality 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 functionality 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/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 \(Sender_t *this, Truffle_t *truffle\)](#)

4.12.1 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.

4.12.2 Function Documentation**4.12.2.1 int UnixSocketSender_free (Sender_t * sender)**

See also

[Sender_free](#)

4.12.2.2 Sender_t* UnixSocketSender_new ()

See also

[Sender_new](#)

4.12.2.3 int UnixSocketSender_send (Sender_t * this, Truffle_t * truffle)

See also

[Sender_send](#)

4.13 src/spp_profinet.c File Reference

Functions

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

Variables

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

4.13.1 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?

4.13.2 Function Documentation

4.13.2.1 void `DissectorInit` ()

Initializes the dissectors for the profinet protocols.

4.13.2.2 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`.

4.13.3 Variable Documentation

4.13.3.1 `Sender_t`* `sender`

The ipc sender.

4.13.3.2 DissectorRegister_t* tlRegister

The top level dissector register.

4.14 src/spp_profinet.h File Reference

Functions

- void [SetupProfiNet](#) ()

4.14.1 Detailed Description

Snort Preprocessor Plugin Header

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

4.14.2 Function Documentation

4.14.2.1 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, 12
- Buffy, 3
 - initialized, 3
 - ops, 3
 - p, 3
- Buffy.h
 - Buffy_free, 20
 - Buffy_get_bits16, 20
 - Buffy_get_bits32, 20
 - Buffy_get_bits64, 20
 - Buffy_get_bits8, 21
 - Buffy_new, 21
- Buffy_free
 - Buffy.h, 20
 - Buffy_ops, 4
- Buffy_get_bits16
 - Buffy.h, 20
 - Buffy_ops, 4
- Buffy_get_bits32
 - Buffy.h, 20
 - Buffy_ops, 4
- Buffy_get_bits64
 - Buffy.h, 20
 - Buffy_ops, 5
- Buffy_get_bits8
 - Buffy.h, 21
 - Buffy_ops, 5
- Buffy_new
 - Buffy.h, 21
- Buffy_ops, 3
 - Buffy_free, 4
 - Buffy_get_bits16, 4
 - Buffy_get_bits32, 4
 - Buffy_get_bits64, 5
 - Buffy_get_bits8, 5
- calling
 - Dissector, 6
- Dissector, 5
 - calling, 6
 - initialized, 6
 - ops, 6
- Dissector.h
 - Dissector_dissect, 22
 - Dissector_free, 23
 - Dissector_getSub, 23
 - Dissector_new, 23
 - Dissector_registerSub, 23
- Dissector_dissect
 - Dissector.h, 22
 - Dissector_ops, 7
- Dissector_free
 - Dissector.h, 23
 - Dissector_ops, 7
- Dissector_getSub
 - Dissector.h, 23
 - Dissector_ops, 7
- Dissector_lower
 - Dissector_ops, 7
- Dissector_new
 - Dissector.h, 23
- Dissector_ops, 6
 - Dissector_dissect, 7
 - Dissector_free, 7
 - Dissector_getSub, 7
 - Dissector_lower, 7
 - Dissector_registerSub, 8
 - Dissector_size, 8
 - Dissector_upper, 8
- Dissector_registerSub
 - Dissector.h, 23
 - Dissector_ops, 8
- Dissector_size
 - Dissector_ops, 8
- Dissector_upper
 - Dissector_ops, 8
- DissectorInit
 - spp_profinet.c, 32
- DissectorRegister, 8
 - initialized, 9
 - ops, 9
- DissectorRegister.h
 - DissectorRegister_get, 25
 - DissectorRegister_insert, 25
 - DissectorRegister_new, 25
- DissectorRegister_get
 - DissectorRegister.h, 25
 - DissectorRegister_ops, 10
- DissectorRegister_insert
 - DissectorRegister.h, 25
 - DissectorRegister_ops, 9
- DissectorRegister_new
 - DissectorRegister.h, 25

- DissectorRegister_ops, 9
 - DissectorRegister_get, 10
 - DissectorRegister_insert, 9
 - DissectorRegister_size, 10
- DissectorRegister_size
 - DissectorRegister_ops, 10
- EtherHeader, 10
- Frame, 11
- hInfo
 - ProtocolTree, 12
- HeaderInfo, 11
- initialized
 - Buffy, 3
 - Dissector, 6
 - DissectorRegister, 9
 - ProtocolTree, 12
 - Sender, 15
- ops
 - Buffy, 3
 - Dissector, 6
 - DissectorRegister, 9
 - ProtocolTree, 13
 - Sender, 15
- p
 - Buffy, 3
- PNRTDissector, 12
- PNRTDissector.c
 - PNRTDissector_dissect, 26
 - PNRTDissector_free, 26
 - PNRTDissector_new, 26
- PNRTDissector_dissect
 - PNRTDissector.c, 26
- PNRTDissector_free
 - PNRTDissector.c, 26
- PNRTDissector_new
 - PNRTDissector.c, 26
- parent
 - ProtocolTree, 13
- ProtocolTree, 12
 - branches, 12
 - hInfo, 12
 - initialized, 12
 - ops, 13
 - parent, 13
- ProtocolTree.h
 - ProtocolTree_branch, 28
 - ProtocolTree_findBranch, 28
- ProtocolTree_free, 28
- ProtocolTree_new, 29
- ProtocolTree_branch
 - ProtocolTree.h, 28
- ProtocolTree_ops, 13
- ProtocolTree_findBranch
 - ProtocolTree.h, 28
 - ProtocolTree_ops, 14
- ProtocolTree_free
 - ProtocolTree.h, 28
 - ProtocolTree_ops, 14
- ProtocolTree_new
 - ProtocolTree.h, 29
 - ProtocolTree_ops, 14
- ProtocolTree_ops, 13
 - ProtocolTree_branch, 13
 - ProtocolTree_findBranch, 14
 - ProtocolTree_free, 14
 - ProtocolTree_new, 14
- Sender, 14
 - initialized, 15
 - ops, 15
- sender
 - spp_profinet.c, 32
 - UnixSocketSender, 18
- Sender.h
 - Sender_free, 30
 - Sender_new, 30
 - Sender_send, 30
- Sender_free
 - Sender.h, 30
 - Sender_ops, 15
- Sender_new
 - Sender.h, 30
- Sender_ops, 15
 - Sender_free, 15
 - Sender_send, 15
- Sender_send
 - Sender.h, 30
 - Sender_ops, 15
- SetupProfiNet
 - spp_profinet.c, 32
 - spp_profinet.h, 33
- spp_profinet.c
 - DissectorInit, 32
 - sender, 32
 - SetupProfiNet, 32
 - tlRegister, 32
- spp_profinet.h
 - SetupProfiNet, 33
- src/Profinet/Buffy-int.h, 19

- src/Profinet/Bufy.h, [19](#)
- src/Profinet/Dissector-int.h, [21](#)
- src/Profinet/Dissector.h, [22](#)
- src/Profinet/DissectorRegister-int.h, [24](#)
- src/Profinet/DissectorRegister.h, [24](#)
- src/Profinet/PNRTDissector.c, [26](#)
- src/Profinet/ProtocolTree-int.h, [27](#)
- src/Profinet/ProtocolTree.h, [27](#)
- src/Profinet/Sender-int.h, [29](#)
- src/Profinet/Sender.h, [29](#)
- src/Profinet/UnixSocketSender.c, [31](#)
- src/spp_profinet.c, [32](#)
- src/spp_profinet.h, [33](#)

- tlRegister
 - spp_profinet.c, [32](#)
- Truffle, [17](#)

- UnixSocketSender, [17](#)
 - sender, [18](#)
- UnixSocketSender.c
 - UnixSocketSender_free, [31](#)
 - UnixSocketSender_new, [31](#)
 - UnixSocketSender_send, [31](#)
- UnixSocketSender_free
 - UnixSocketSender.c, [31](#)
- UnixSocketSender_new
 - UnixSocketSender.c, [31](#)
- UnixSocketSender_send
 - UnixSocketSender.c, [31](#)