spp_profinet

Generated by Doxygen 1.8.6

Mon Feb 1 2016 08:33:44

Contents

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 2

File Index

2.1 File List

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

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

buffy	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

this	the calling buffer
bit_offset	the offset for from the currenty buffer position
the	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

this	the calling buffer
bit_offset	the offset for from the currenty buffer position
the	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

this	the calling buffer
bit_offset	the offset for from the currenty buffer position
the	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

this	the calling buffer
bit_offset	the offset for from the currenty buffer position
the	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.
- unit64_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

this	the calling Dissector
buf	the buffer pointing to the package data currently being processed
tree	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

this	the dissector calling Dissector_getSub
data	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

this	the dissector to register the subDissector on
subDissector	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

this	the calling register
dissector	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

this	the DissectorRegister calling
data	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

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

this	the calling ProtocolTree
info	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

this	the calling ProtocolTree
the	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

proto	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

sender	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

this	the calling sender
truffle	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 imediately 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/Buffy-int.h File Reference

The internal structure of Buffy.

Data Structures

- struct Buffy_ops

  The operations that can be called by a Buffy buffer.
- struct Buffy

  Buffer for dissecting packages in the profinet plugin.

Functions

- Buffy_t ∗ **Buffy_new** (Packet ∗p)

4.1.1 Detailed Description

The internal structure of Buffy.

4.2 src/Profinet/Buffy.h File Reference

The interface for Buffy.

Functions

- Buffy_t ∗ Buffy_new (Packet ∗p)

  Creates a new buffer from the given snort package.
- 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.

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

buffy	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

this	the calling buffer
bit_offset	the offset for from the currenty buffer position
the	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

this	the calling buffer
bit_offset	the offset for from the currenty buffer position
the	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

this	the calling buffer
bit_offset	the offset for from the currenty buffer position
the	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

this	the calling buffer
bit_offset	the offset for from the currenty buffer position
the	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

p	the packet as defined by snort

Returns

the instantiated Buffer

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

This Header discribes 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 discribes 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

this	the calling Dissector
buf	the buffer pointing to the package data currently being processed
tree	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

this	the dissector calling Dissector_getSub
data	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

ops	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

this	the dissector to register the subDissector on
subDissector	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

this	the DissectorRegister calling
data	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

this	the calling register
dissector	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

ops	the pointer to the operations used for this DissectorRegister

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

this	the calling ProtocolTree
info	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

this	the calling ProtocolTree
the	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

proto	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 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

sender	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

ops	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

this	the calling sender
truffle	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 arguements 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