spp_profinet

Generated by Doxygen 1.8.6

Sat Jan 30 2016 14:33:25

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 Dissector Struct Reference

Data Fields

- bool initialized
- const struct Dissector_ops * ops
- Dissector_t * calling

3.1.1 Field Documentation

3.1.1.1 Dissector_t* Dissector::calling

The dissector this dissector has been called from.

3.1.1.2 bool Dissector::initialized

Whether this dissector was initialized.

3.1.1.3 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

3.2 Dissector_ops Struct Reference

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.

3.2.1 Field Documentation

3.2.1.1 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

3.2.1.2 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

3.2.1.3 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

3.2.1.4 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

3.2.1.5 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.

3.2.1.6 size_t Dissector_ops::Dissector_size

Returns the number of subdissectors in this dissector.

Returns

the number of sub-dissectors in this dissector

3.2.1.7 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

3.3 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

3.3.1 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.

3.3.2 Field Documentation

3.3.2.1 bool DissectorRegister::initialized

Whether this dissector register is initialized.

3.3.2.2 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

3.4 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.

3.4.1 Detailed Description

The operations that can be called by a DissectorRegister.

3.4.2 Member Function Documentation

3.4.2.1 **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

3.4.3 Field Documentation

3.4.3.1 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

3.4.3.2 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

3.5 Entry Struct Reference

Data Fields

- char **stringValue** [256]
- uint64_t **intValue**
- int **endianess**

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

- src/Profinet/Entry.h

3.6 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/Profinet/Buffy.h File Reference

The interface for Buffy.

Functions

- Buffy_t ∗ **Buffy_new** (Packet ∗p)
- 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.

4.1.1 Detailed Description

The interface for Buffy. Creates a new buffer from the given snort package.

Parameters

p	the packet as defined by snort

Returns

the instantiated Buffer

4.1.2 Function Documentation

4.1.2.1 void Buffy_free (Buffy_t ∗ *buffy*)

Frees the given buffer from memory.

Parameters

buffy	the buffer to be freed

4.1.2.2 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

4.1.2.3 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

4.1.2.4 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

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

Data Structures

- struct Dissector_ops
- struct Dissector

Functions

- Dissector_t ∗ **Dissector_new** (const struct Dissector_ops ∗ops)

4.2.1 Detailed Description

This Header discribes the internal structure of the Dissector type. It defines the basic interface for operations.

4.3 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.

4.3.1 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.

4.3.2 Function Documentation

4.3.2.1 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

4.3.2.2 void Dissector_free (Dissector_t ∗ *dissector*)

Frees the given dissector.

4.3.2.3 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

4.3.2.4 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

4.3.2.5 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.

4.4 src/Profinet/DissectorRegister-int.h File Reference

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)

4.4.1 Detailed Description

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

4.5 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.

4.5.1 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.

4.5.2 Function Documentation

4.5.2.1 **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

4.5.2.2 **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

4.5.2.3 **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

4.6 src/Profinet/Entry.h File Reference

The interface for entries.

Data Structures

- struct Entry

Macros

- #define **LITTLE_ENDIAN** 0
- #define **BIG_ENDIAN** 1

Typedefs

- typedef struct Entry **Entry_t**

Functions

- Entry_t ∗ Entry_new ()
- Entry_t ∗ Truffle_get (Truffle_t ∗this, char ∗key)

  Returns the entry that is mapped to the given key in this truffle.
- Entry_t ∗ Truffle_putChars (Truffle_t ∗this, char ∗key, char ∗value)

  Inserts the given entry into the truffle mapped to the key.
- Entry_t ∗ **Truffle_putInt** (Truffle_t ∗this, char ∗key, uint64_t value)

4.6.1 Detailed Description

The interface for entries.

4.6.2 Function Documentation

4.6.2.1 Entry_t∗ Entry_new ( )

Creates a new entry.

Returns

the newly created entry

4.6.2.2 Entry_t∗ Truffle_get (Truffle_t ∗ *this*, char ∗ *key*)

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

Parameters

this	the calling truffle
key	the key to be looking for

4.6.2.3 Entry_t∗ Truffle_putChars (Truffle_t ∗ *this*, char ∗ *key*, char ∗ *value*)

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

Parameters

this	the calling truffle
key	the value to map the entry to
entry	the entry to be put into the truffle

Returns

NULL if there was no previous entry mapped to the given key, the existing entry otherwise - which will be overwritten

4.7 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)
- void Sender_free (Sender_t ∗dissector)
- int Sender_send (Sender_t ∗this, Truffle_t ∗truffle)

4.7.1 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.

4.7.2 Function Documentation

4.7.2.1 void Sender_free (Sender_t ∗ *dissector*)

Frees the given dissector.

4.7.2.2 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

4.7.2.3 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.

4.8 src/Profinet/Truffle.h File Reference

The interface for truffles.

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)

4.8.1 Detailed Description

The interface for truffles. Truffle header file

4.8.2 Function Documentation

4.8.2.1 Entry_t∗ Truffle_get (Truffle_t ∗ *this,* char ∗ *key*)

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

Parameters

this	the calling truffle
key	the key to be looking for

4.8.2.2 Truffle_t∗ Truffle_new ( )

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

4.8.2.3 Entry_t∗ Truffle_put (Truffle_t ∗ *this,* char ∗ *key,* Entry_t *entry*)

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

Parameters

this	the calling truffle

key	the value to map the entry to
entry	the entry to be put into the truffle

4.9 src/spp_profinet.c File Reference

Functions

- void SetupProfiNet ()
- void DissectorInit ()

Variables

- DissectorRegister_t ∗ tlRegister
- Sender_t ∗ sender

4.9.1 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 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?

4.9.2 Function Documentation

4.9.2.1 void DissectorInit ( )

Initializes the dissectors for the profinet protocols.

4.9.2.2 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.

4.9.3 Variable Documentation

4.9.3.1 Sender_t∗ sender

The ipc sender.

4.9.3.2 DissectorRegister_t∗ tlRegister

The top level dissector register.

4.10 src/spp_profinet.h File Reference

Functions

- void SetupProfiNet ()

4.10.1 Detailed Description

Snort Preprocessor Plugin Header

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

4.10.2 Function Documentation

4.10.2.1 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