

# FIPCATURE Library

## *User's Guide*

Exoligent's Team

Version v1.0.0, 2023-02-07

# Table of Contents

1. Disclaimer .....	1
2. Introduction .....	2
3. Installation .....	3
3.1. Linux .....	3
3.1.1. Kernel Module .....	4
3.1.2. Library .....	5
3.2. Windows .....	6
3.2.1. Driver .....	7
3.2.2. Library .....	9
3.3. Examples .....	10
3.3.1. Test - All Frames Sniffer .....	10
4. Functions .....	14
4.1. General .....	14
4.1.1. init .....	14
4.1.2. exit .....	15
4.1.3. version_get .....	16
4.1.4. error_get .....	17
4.1.5. strerror .....	18
4.2. Device .....	19
4.2.1. device_list_get .....	19
4.2.2. device_open .....	20
4.2.3. device_reset .....	21
4.2.4. device_close .....	22
4.2.5. device_infos_get .....	23
4.3. Capture Session .....	24
4.3.1. sess_init .....	24
4.3.2. sess_exit .....	27
4.3.3. sess_report_get .....	28
4.3.4. sess_stats_clear .....	31
4.3.5. sess_start .....	32
4.3.6. sess_stop .....	33
5. Structures .....	34
5.1. General .....	34
5.1.1. date .....	34
5.1.2. version .....	35
5.2. Device .....	36
5.2.1. dev_infos .....	36

5.3. Configuration	38
5.3.1. sess_cfg	38
5.4. Frames	41
5.4.1. frm_msg_addr	41
5.4.2. frm_msg_hdr	43
5.4.3. frm_msg	44
5.4.4. frm_usrdata	45
5.4.5. frm_var_list	46
5.4.6. frm_var	47
5.4.7. frm	48
5.5. Bus Arbiter	50
5.5.1. ba_aper_wind	50
5.5.2. ba_per_wind	51
5.5.3. ba_mcycle	52
5.5.4. ba_req	53
5.5.5. ba_wind	54
5.6. Infos/Status/Report	55
5.6.1. sess_ba_info	55
5.6.2. sess_infos	56
5.6.3. sess_node_info	57
5.6.4. sess_report	60
5.6.5. sess_stats	61
5.6.6. sess_status	64
5.6.7. sess	65
6. Enumerations	66
6.1. ba_id_type	66
6.2. ba_status	67
6.3. ba_wind_type	68
6.4. bitrate	69
6.5. error_code	70
6.6. frm_ctrl_type	71
6.7. frm_err_code	74
6.8. frm_pdu_type	75
6.9. frm_type	76
6.10. frm_usrdata_type	77
6.11. sess_state	78
Appendix A: SM-MPS variables	79
A.1. Identification - 0x10XY	79
A.2. Report - 0x11XY	80

A.3. Presence - 0x14XY .....	82
A.4. Presence Check - 0x9002.....	82
A.5. BA synchronization - 0x9003.....	84
Appendix B: Glossary of acronyms .....	85
Appendix C: Revision History .....	86

# Chapter 1. Disclaimer

Information contained in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. EXOLIGENT MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION, INCLUDING BUT NOT LIMITED TO ITS CONDITION, QUALITY, PERFORMANCE, MERCHANTABILITY OR FITNESS FOR PURPOSE. Exoligent disclaims all liability arising from this information and its use. Use of Exoligent devices in life support and/or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless Exoligent from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Exoligent intellectual property rights unless otherwise stated.

Exoligent SARL

390 rue d'Estienne d'Orves, 92700 Colombes - France

Tel: +33(0) 1 42 42 42 00

Web Site: <https://www.exoligent.com/>

Copyright © 2023 Exoligent SARL. All rights reserved.

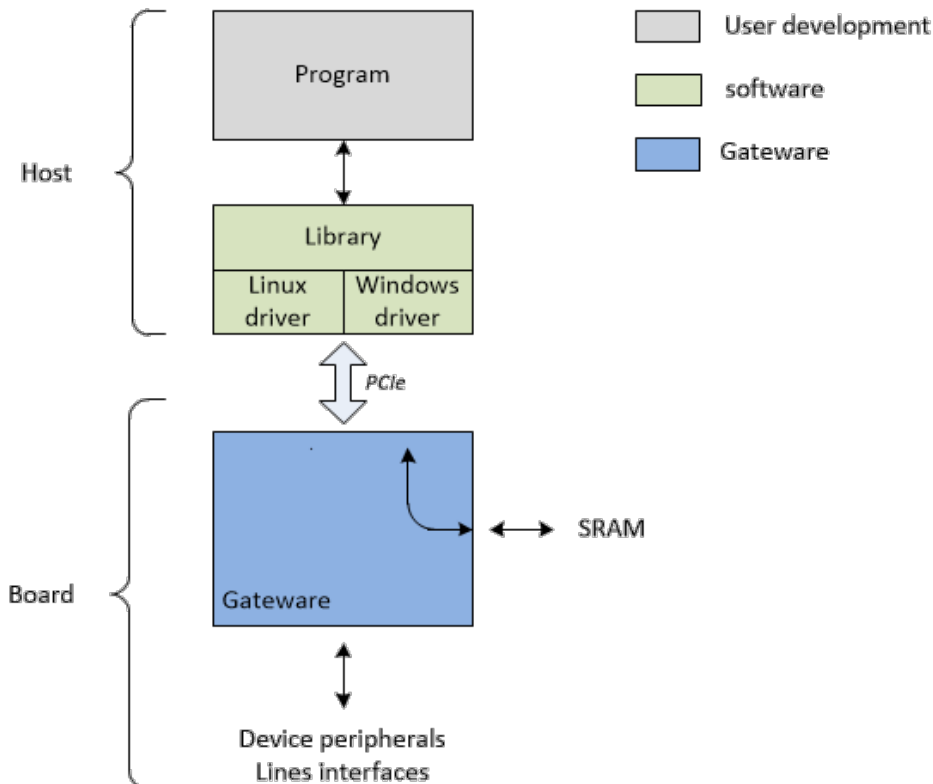
## Chapter 2. Introduction

FIPCapture library is a C API providing a programming interface to the Exoligent's FIPWatcher coprocessor.

It offers a set of functions to integrate a FIP bus spy in your user application :

- Frames Handler
- Frames Statistics
  - Errors Detection (PHY/DL/APP OSI Layers)
  - Frame Counters
- Bus Arbiter Information
  - BA address
  - Macrocycle Auto-Detection
- Nodes Information
  - Presence List
  - Nodes Identification
  - Nodes Local Report

*FIPCapture overview (Principle Diagram)*



# Chapter 3. Installation

Let's start by downloading the archive of the latest version of the library from the Exoligent website: [Download section](#)

## 3.1. Linux

The linux archive has the following tree structure

- **[docs]**
  - FIPCapture Library - User's Guide (\*.pdf) :  
The User's Guide in PDF format.
  - fipcapture.html :  
The User's Guide in HTML format.
- **[drivers]**
  - **[linux]**  
The source code of the Linux kernel module for Exoligent FIPWatcher PCI/PCIe devices:
    - **[udev.rules.d]**
      - 10-fipwatcher.rules
    - install.sh
    - Makefile
    - fw-drv.c
    - fw-drv.h
    - fw-pci.c
    - fw-pci.h
    - uninstall.sh
- **[include]**  
Header files to include in your projects to use the library:
  - fw-board.h
  - libfipcap.h
- **[lib]**
  - **[aarch64]**
    - **[static]**
      - libfipcap.a [arm64 static lib]
      - libfipcap.so.1.0.0 [arm64 shared lib]
  - **[arm]**
    - **[static]**

- libfipcap.a [arm32 static lib]
- libfipcap.so.1.0.0 [arm32 shared lib]
- **[i386]**
  - **[static]**
    - libfipcap.a [x86 static lib]
    - libfipcap.so.1.0.0 [x86 shared lib]
  - **[x86\_64]**
    - **[static]**
      - libfipcap.a [x86\_64 static lib]
      - libfipcap.so.1.0.0 [x86\_64 shared lib]
- **[tools] :**  
Turnkey examples to get started as soon as possible!
  - **[fipcap\_sniff\_all]**
  - **[fipudpd]**
- install.sh
- uninstall.sh
- release-notes.txt
- readme.txt

### 3.1.1. Kernel Module

Open a terminal, and go to the linux driver directory:

```
$ cd driver/linux
```

Execute the following commands to build and install the kernel module:

```
$ make  
$ sudo ./install.sh
```



#### Two files will be copied to your system

1. **fipwatcher.ko** file to the path:  
/lib/modules/\$(uname -r)/kernel/drivers/fip
2. **10-fipwatcher.rules** file to the path:  
/etc/udev/rules.d

To remove the kernel module, enter the following command :



```
$ sudo ./uninstall.sh
```

### 3.1.2. Library

Open a terminal, and go to the archive package root.

Then, enter the following command to install the FIPCapture library on your machine:

```
$ sudo ./install.sh
```



#### Files will be copied to your system

1. **libfipcap.a** and **libfipcap.so.1.0.0** files to the path:  
/usr/local/lib
2. **header (\*.h)** files to the path:  
/usr/local/include/fipcapture

To remove these files, enter the following command:

```
$ sudo ./uninstall.sh
```

## 3.2. Windows

Run the latest installer (.exe), and follow the wizard steps. All the files will be copied to the folder:

```
C:\Program Files (x86)\Exoligent\FIPCapture\
```

This directory has the following tree structure

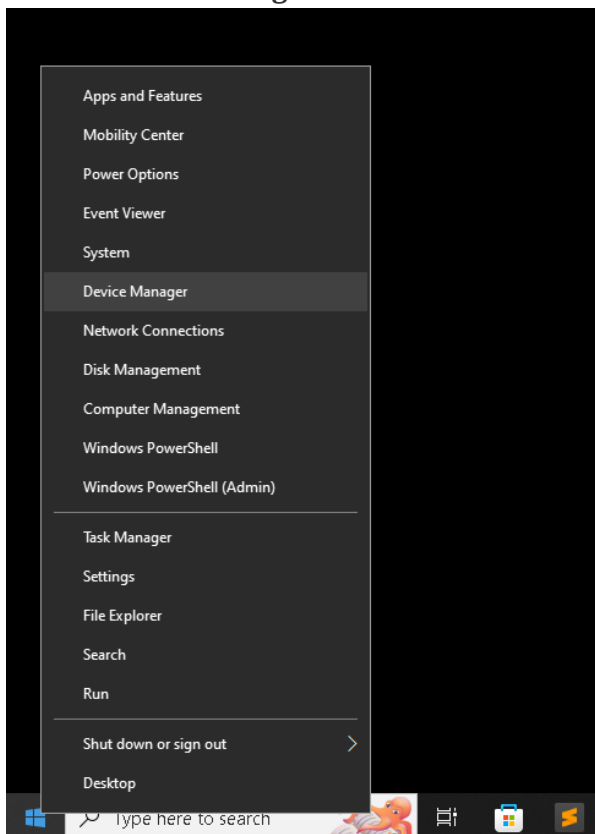
- **[docs]**
  - FIPCapture Library - User's Guide (\*.pdf) :  
The User's Guide in PDF format.
  - fipcapture.html :  
The User's Guide in HTML format.
- **[drivers]**
  - **[win]**  
The binaries of the Windows driver for Exoligent FIPWatcher PCI/PCIe devices:
    - **[Driver]**
      - **[x86]**
        - fipwatcher.sys
      - **[x86-64]**
        - fipwatcher64.sys
    - fipwatcher.cat
    - fipwatcher.inf
    - fipwatcher64.cat
- **[include]**  
Header files to include in your projects to use the library:
  - fw-board.h
  - libfipcap.h
- **[lib]**
  - **[i686]**
    - **[static]** - MinGW compiler: i686-8.1.0-posix-dwarf-rt\_v6-rev0
      - libfipcap.a [x86 static lib]
      - libfipcap.dll [x86 shared lib]
      - libfipcap.lib [x86 import lib]
  - **[x86\_64]**
    - **[static]** - MinGW compiler: x86\_64-8.1.0-posix-seh-rt\_v6-rev0

- libfipcap.a [x86\_64 static lib]
- libfipcap.dll [x86\_64 shared lib]
- libfipcap.lib [x86\_64 import lib]
- **[tools]** :  
Turnkey examples to get started as soon as possible!
  - **[fipcap\_sniff\_all]**
  - **[fipudpd]**
- release-notes.txt
- readme.txt

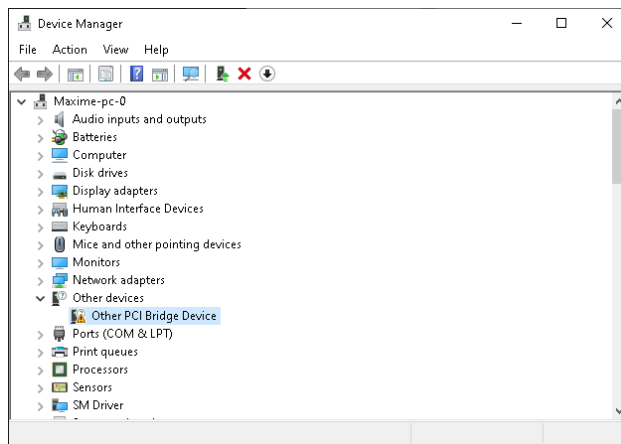
### 3.2.1. Driver

#### Installation

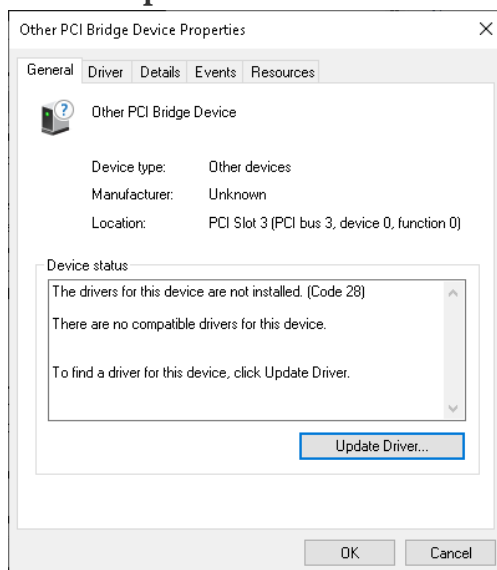
- Open the **Device Manager** window
  - Press **Windows+X** or right-click to **Start** button, then a menu will appear
  - Select **Device Manager** from the list



- Double-click on the new **Other PCI bridge device** detected



- Click on **Update the driver**

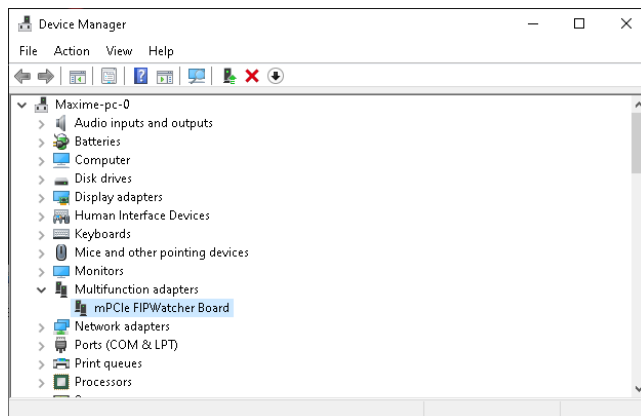


- Select the last package driver, and click on **Next** button

# Target directory for driver

C:\Program Files (x86)\Exoligent\FIPCapture\drivers\win

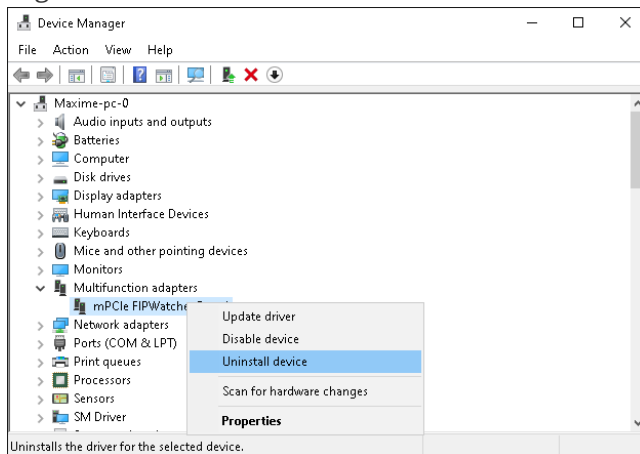
- FIPWatcher driver is now installed !



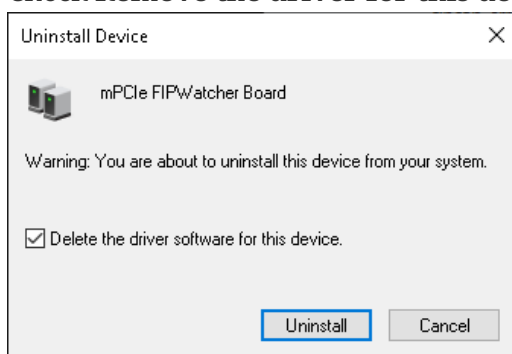
## Uninstallation

- Go to the **Device Manager** window

- Right-click on the FIPWatcher device to be removed, and click on **Uninstall** button



- Check **Remove the driver for this device**, and confirm the device uninstall



### 3.2.2. Library

The FIPCapture library will be automatically installed by the package wizard.



#### Files will be copied to your system

1. **libfipcap.dll** file to the path:  
C:\Program Files (x86)\Exoligent\FIPCapture\lib
2. **header (\*.h)** files to the path:  
C:\Program Files (x86)\Exoligent\FIPCapture\include

To remove the package, execute the uninstaller wizard:

```
C:\Program Files (x86)\Exoligent\FIPCapture\Uninstall.exe
```

## 3.3. Examples

To quickly get into the swing of things, the source code of some examples is provided with the package.

To access to the examples, open a terminal and enter the following commands:

*Linux*

```
# From extracted archive directory
$ cd tools
```

*Windows*

```
$ cd C:\Program Files (x86)\Exoligent\FIPCapture\tools
```



If you do not have administrator rights, it is advisable to copy/paste the following directories into your own workspace:

- C:\Program Files (x86)\Exoligent\FIPCapture\tools
- C:\Program Files (x86)\Exoligent\FIPCapture\lib
- C:\Program Files (x86)\Exoligent\FIPCapture\include

For example paste it to a new directory in 'My Documents':

- C:\Users\%USERNAME%\Documents\FIPCapture\tools
- C:\Users\%USERNAME%\Documents\FIPCapture\lib
- C:\Users\%USERNAME%\Documents\FIPCapture\include

We will now briefly describe the example(s):

### 3.3.1. Test - All Frames Sniffer

This example aims to perform a FIP capture session to sniff all the frames of a FIP network and display them.

**Build the example:**

*Linux*

```
$ cd tools/fipcap_sniff_all

# Build for i386 target (32-bit)
$ make MACHINE=i386

# Build for x86_64 target (64-bit)
```

```
$ make MACHINE=x86_64
```

### Windows

```
# Build the example with static method way.
$ set SYS_NAME=windows
$ cd tools/fipcap_sniff_all

# Build for i686 target (32-bit)
# Note: MinGW compiler: i686-8.1.0-posix-dwarf-rt_v6-rev0
$ make MACHINE=i686

# Build for x86_64 target (64-bit)
# Note: MinGW compiler: x86_64-8.1.0-posix-seh-rt_v6-rev0
$ make MACHINE=x86_64
```

### Get the help:

#### Linux

```
$ ./fipcap_sniff_all -h
```

#### Windows

```
$ fipcap_sniff_all.exe -h
```

Usage: fipcap\_sniff\_all [OPTION]...

It captures all the frames of the FIP network.

By default, if no option is added, the app opens the first PCI/PCIe device with index 1 [-i 1] and statistics verbosity.

#### Options:

- i device index [default=1]
- l list the FIP boards present on the host machine
- v be verbose (-vv or -vvv for higher verbosity)
- h show version and this help and exit

#### Examples:

```
/* statistics verbosity */
fipcap_sniff_all
```

```
/* raw frames verbosity */
fipcap_sniff_all -vv
```

```
/* detailed frames verbosity */
```

```
fipcap_sniff_all -vvv
```

**Launch a FIP capture:***Linux*

```
$ ./fipcap_sniff_all -i 1 -vv
```

*Windows*

```
$ fipcap_sniff_all.exe -i 1 -vv
```

```
[02-07 12:08:30.359609] app => [info] [fip] device info
[02-07 12:08:30.359687] app => [info]     index      : 1
[02-07 12:08:30.359705] app => [info]     sn         : 0x01270422
[02-07 12:08:30.359718] app => [info]     vid        : 0x11aa
[02-07 12:08:30.359730] app => [info]     did        : 0x1556
[02-07 12:08:30.359745] app => [info]     ssvid     : 0x11aa
[02-07 12:08:30.359759] app => [info]     ssdid    : 0x5802
[02-07 12:08:30.359773] app => [info]     bar_cnt   : 2
[02-07 12:08:30.359787] app => [info]     bar_bsz[0] : 4096
[02-07 12:08:30.359802] app => [info]     bar_base[0] : 0xa2000000
[02-07 12:08:30.359816] app => [info]     bar_bsz[1] : 33554432
[02-07 12:08:30.359831] app => [info]     bar_base[1] : 0xa0000000
[02-07 12:08:30.359843] app => [info]     irq_number : 149
[02-07 12:08:30.359857] app => [info]     drv_version : 3.1.0
[02-07 12:08:30.359873] app => [info] test: v1.0.0 - fipcap lib: v1.0.0
[02-07 12:08:30.359886] app => [info] [fip] capture configuration
[02-07 12:08:30.359900] cap => [info] [fip] capture init
[02-07 12:08:30.477491] cap => [event] reset component (fipwatcher)
[02-07 12:08:30.477563] cap => [info] [fip] capture infos
[02-07 12:08:30.477569] cap => [info]     sn         : 0x01270422
[02-07 12:08:30.477572] cap => [info]     driver      : v3.1.0 [build date:
01/30/23]
[02-07 12:08:30.477575] cap => [info]     library     : v1.0.0 [build date:
02/07/23]
[02-07 12:08:30.477577] cap => [info] [fip] capture start
[02-07 12:08:30.484764] app => [info] app started. press enter to close...

[...]
```

The example sequence is as follows:

- Open PCI/PCIe fipwatcher device (index 1)
- Load Session Configuration



- Start the capture
- Infinite Loop



During this loop, an interrupt is raised by the FIP coprocessor each time frame packets are received from the FIP network.

This interrupt triggers the user frame handler (see: `tst_fipcap_frm_handler` function in the file: `tools/fipcap_sniff_all/session.c`).

Depending on the verbosity set, the display of the frames is more or less detailed.

- Break the loop on an user keyboard press
- Stop the capture
- Unload Configuration
- Close PCI/PCIe FIPWatcher device

# Chapter 4. Functions

In this chapter, we will discover and describe the whole API (Application Programming Interface) of *FIPCAPTURE*.

## 4.1. General

### 4.1.1. init

#### Description

Library internal initialization

#### Prototype

```
int fipcap_init()
```

#### Parameters

- *IN* - None
- *OUT* - None

#### Return Value

If successful, `fipcap_init()` returns 0.

If unsuccessful, `fipcap_init()` returns -1 and sets `errno` value.

#### Remarks



This function must always be called before using any other function of the library.

### 4.1.2. exit

#### Description

Free all internal ressources used by the FIPCapture library

#### Prototype

```
void fipcap_exit()
```

#### Parameters

- *IN* - None
- *OUT* - None

#### Return Value

NONE

#### Remarks



This function must always be called at the end of the use of the library.

### 4.1.3. version\_get

#### Description

Gets the software library version.

#### Prototype

```
const struct fipcap_version *fipcap_version_get()
```

#### Parameters

- *IN* - None
- *OUT* - None

#### Return Value

Pointer to a `struct fipcap_version`.

#### Example

```
int main(int argc, char *argv[])
{
    const struct fipcap_version *lib_version;

    fipcap_init();

    /* fipcap - get lib version */
    lib_version = fipcap_version_get();

    printf("fipcap lib: v%d.%d.%d [build date: %02d/%02d/%02d]\n",
        lib_version->info.major,
        lib_version->info.minor,
        lib_version->info.patch,
        lib_version->date.info.month,
        lib_version->date.info.day,
        lib_version->date.info.year);

    fipcap_exit();

    return 0;
}
```

#### 4.1.4. error\_get

##### Description

Gets the last error code.

##### Prototype

```
int fipcap_error_get()
```

##### Parameters

- *IN* - None
- *OUT* - None

##### Return Value

The last error code.

##### Remarks

See `enum fipcap_error_code` to get the list of the specific library errors codes.

### 4.1.5. strerror

#### Description

Gets the error string specified by its error code.



The library provides this error code via the *lvalue*: ***errno*** or via `fipcap_error_get()` function.

#### Prototype

```
const char *fipcap_strerror(int err)
```

#### Parameters

- *IN*
  - **err**:  
Error code.
- *OUT* - None

#### Return Value

Error in string format.

#### Remarks

See `enum fipcap_error_code` to get the list of the specific library errors codes.

## 4.2. Device

### 4.2.1. device\_list\_get

#### Description

Gets the list of PCI/PCIe fipwatcher devices present on the system.



The FIPWatcher driver supports up to **16** devices.

#### Prototype

```
int fipcap_device_list_get(struct fw_dev_infos *dev_infos, int *dev_cnt)
```

#### Parameters

- *IN* - None
- *OUT*
  - **dev\_infos:**  
Information list of detected devices.  
See `struct fw_dev_infos`.
  - **dev\_cnt:**  
Count of detected devices.

#### Return Value

If successful, `fipcap_device_list_get()` returns 0.

If unsuccessful, `fipcap_device_list_get()` returns -1 and sets `errno` to one of the following values:

- **EINVAL:**  
Invalid input/output parameter.

## 4.2.2. device\_open

### Description

Opens a PCI/PCIe fipwatcher device.

### Prototype

```
struct fipcap_dev *fipcap_device_open(uint8_t dev_id)
```

### Parameters

- *IN*
  - **dev\_id:**  
Device index on the system.
- *OUT* - None

### Return Value

If successful, `fipcap_device_open()` returns a new `struct fipcap_dev` pointer (opaque structure).

If unsuccessful, `fipcap_device_open()` returns NULL and sets `errno` to one of the following values:

- **EINVAL:**  
Invalid input parameter.
- **ENOMEM:**  
Memory allocation error.
- **[..]:**  
Other posix errors related to a file opening error



### 4.2.3. device\_reset

#### Description

Resets PCI/PCIe FIPWatcher device.

- FIP Coprocessor reset
- Logic interface reset (of carrier board)

#### Prototype

```
int fipcap_device_reset(struct fipcap_dev *dev)
```

#### Parameters

- *IN*
  - **dev:**  
Pointer to the device to reset.
- *OUT* - None

#### Return Value

If successful, `fipcap_device_reset()` returns 0.

If unsuccessful, `fipcap_device_reset()` returns -1 and sets `errno` to one of the following values:

- **EINVAL:**  
Invalid input parameter.

#### 4.2.4. device\_close

##### Description

Closes a PCI/PCIe fipwatcher device.

##### Prototype

```
int fipcap_device_close(struct fipcap_dev *dev)
```

##### Parameters

- *IN*
  - **dev:**  
Pointer to the device to close.
- *OUT* - None

##### Return Value

If successful, `fipcap_device_close()` returns 0.

If unsuccessful, `fipcap_device_close()` returns -1 and sets `errno` to one of the following values:

- **EINVAL:**  
Invalid device pointer.

### 4.2.5. device\_infos\_get

#### Description

Gets information relative to the PCI/PCIe fipwatcher device.

#### Prototype

```
int fipcap_device_infos_get(struct fipcap_dev *dev, struct fw_dev_infos *info)
```

#### Parameters

- *IN*
  - **dev:**  
Pointer to the device to query.
- *OUT*
  - **info:**  
Info structure.  
See `struct fw_dev_infos`.

#### Return Value

If successful, `fipcap_device_infos_get()` returns 0.

If unsuccessful, `fipcap_device_infos_get()` returns -1 and sets `errno` to one of the following values:

- **EINVAL:**  
Invalid input/output parameter.

## 4.3. Capture Session

### 4.3.1. sess\_init

#### Description

Creates a new FIP capture context inside the library and binds it with a fipwatcher coprocessor.

#### Prototype

```
struct fipcap_sess *fipcap_sess_init(struct fipcap_sess_cfg *cfg)
```

#### Parameters

- *IN*
  - **cfg:**  
Pointer to the session configuration.  
See `struct fipcap_sess_cfg`.
- *OUT* - None

#### Return Value

If successful, `fipcap_sess_init()` returns a new `struct fipcap_sess` pointer.

If unsuccessful, `fipcap_sess_init()` returns NULL and sets `errno` to one of the following values:

- **EINVAL:**  
Invalid input configuration.
- **ENOMEM:**  
Memory allocation error.
- **FIPCAP\_ERR\_SESS\_HANDLER\_MISSING:**  
Some user handlers are mandatory to continue the capture session creation:
  - `fipcap_reset_handler`
  - `fipcap_error_handler`
- **FIPCAP\_ERR\_DEV\_ALREADY\_BIND:**  
The provided device is already bound to another FIP capture session.
- **[..]:**  
Other posix errors related to a file opening error

#### Example

```
void usr_rst_handler(struct fipcap_sess *sess)
{
    /* calling the reset function for a pci/pcie device */
    if (fipcap_device_reset(sess->infos->cfg.dev)) {
        printf("reset failed: %s\n", fipcap_strerror(errno));
        return;
    }
}
```

```
}
_print(src, evt, "reset coprocessor component (fipwatcher) done\n");
}

void usr_err_handler(struct fipcap_sess *sess,
                    enum fipcap_error_code code)
{
    printf("error_handler: %s (err_code=%d)\n",
          fipcap_strerror(code), code);
}

int main(int argc, char *argv[])
{
    int err = 0;
    struct fipcap_dev *dev;
    struct fipcap_sess *sess;
    struct fipcap_sess_cfg cfg;

    /* fipcap lib initialization [mandatory] */
    if (fipcap_init()) {
        printf("cannot init fipcap library: %s\n",
              fipcap_strerror(errno));
        return -1;
    }

    /* open a pci/pcie device (1st index) */
    dev = fipcap_device_open(1);
    if (!dev) {
        printf("cannot open device: %s\n",
              fipcap_strerror(errno));
        fipcap_exit();
        return -1;
    }

    /**
     * Session initialization:
     * Minimal set-up for a PCI/PCIE fipwatcher device
     */
    cfg.dev = dev; /* coprocessor attachment: pci/pcie device to bind */
    cfg.fipcap_error_handler = usr_err_handler; /* local handler to notify
                                                the internal errors of
                                                the library */
    cfg.fipcap_reset_handler = usr_rst_handler; /* local handler to reset
                                                the bound device */

    sess = fipcap_sess_init(&cfg);
    if (!sess) {
        printf("session initialization failed: %s\n",
              fipcap_strerror(errno));
    }
}
```

```
        err = -1;
        goto end;
    }

    /**
     * Other tasks
     */
    /* ... */

end:
    /**
     * Session exit
     */
    /*...*/

    /* close device */
    fipcap_device_close(dev);

    /* fipcap lib exit [mandatory] */
    fipcap_exit();
    return err;
}
```

### 4.3.2. sess\_exit

#### Description

Stops the coprocessor and deallocates all resources attached to the capture session inside the library.

#### Prototype

```
int fipcap_sess_exit(struct fipcap_sess *sess)
```

#### Parameters

- *IN*
  - **sess:**  
Pointer to the session to exit.  
See `struct fipcap_sess`.
- *OUT* - None

#### Return Value

If successful, `fipcap_sess_exit()` returns 0.

If unsuccessful, `fipcap_sess_exit()` returns -1 and sets `errno` to one of the following values:

- **EINVAL:**  
Invalid input parameter.
- **FIPCAP\_ERR\_DEV\_IRQ\_HANDLER\_STOPPED:**  
Internal IRQ handler for PCI/PCIe fipwatcher device is already stopped.
- **FIPCAP\_ERR\_SESS\_NOT\_STOP:**  
Cannot stop the capture. The exit procedure has therefore failed.

### 4.3.3. sess\_report\_get

#### Description

Gets the full diagnostic report of the FIP capture.

#### Prototype

```
int fipcap_sess_report_get(struct fipcap_sess *sess,
                          struct fipcap_sess_report *report)
```

#### Parameters

- *IN*
  - **sess:**  
Pointer to the target capture session.  
See `struct fipcap_sess`.
- *OUT*
  - **report:**  
Pointer to an output `struct fipcap_sess_report`.

#### Return Value

If successful, `fipcap_sess_report_get()` returns 0.

If unsuccessful, `fipcap_sess_report_get()` returns -1 and sets `errno` to one of the following values:

- **EINVAL:**  
Invalid input/output parameters.

#### Example

```
int main(int argc, char *argv[])
{
    int err = 0;
    struct fipcap_sess *sess;
    struct fipcap_sess_report rep; /* full report */

    /**
     * Session initialization
     */
    /*...*/

    /**
     * Session startup
     */
    /*...*/

    /* get capture report */
}
```



```

err = fipcap_sess_report_get(sess, &rep);
if (err) {
    printf("capture report getter failed: %s\n", fipcap_strerror(errno));
    goto end;
}

printf("  CAPTURED FRAMES      : TOTAL=%d\n", rep.stats.frm.total);
printf("    id_dat             : %d\n", rep.stats.frm.id_dat);
printf("    id_msg             : %d\n", rep.stats.frm.id_msg);
printf("    id_rq1            : %d\n", rep.stats.frm.id_rq1);
printf("    id_rq2            : %d\n", rep.stats.frm.id_rq2);
printf("    rp_dat            : %d\n", rep.stats.frm.rp_dat);
printf("    rp_dat_msg        : %d\n", rep.stats.frm.rp_dat_msg);
printf("    rp_dat_rq1        : %d\n", rep.stats.frm.rp_dat_rq1);
printf("    rp_dat_rq2        : %d\n", rep.stats.frm.rp_dat_rq2);
printf("    rp_dat_rq1_msg    : %d\n", rep.stats.frm.rp_dat_rq1_msg);
printf("    rp_dat_rq2_msg    : %d\n", rep.stats.frm.rp_dat_rq2_msg);
printf("    rp_rq1            : %d\n", rep.stats.frm.rp_rq1);
printf("    rp_rq2            : %d\n", rep.stats.frm.rp_rq2);
printf("    rp_msg_noack       : %d\n", rep.stats.frm.rp_msg_noack);
printf("    rp_msg_ack_even    : %d\n", rep.stats.frm.rp_msg_ack_even);
printf("    rp_msg_ack_odd     : %d\n", rep.stats.frm.rp_msg_ack_odd);
printf("    rp_ack_even_p      : %d\n", rep.stats.frm.rp_ack_even_p);
printf("    rp_ack_odd_p       : %d\n", rep.stats.frm.rp_ack_odd_p);
printf("    rp_ack_even_m      : %d\n", rep.stats.frm.rp_ack_even_m);
printf("    rp_ack_odd_m       : %d\n", rep.stats.frm.rp_ack_odd_m);
printf("    rp_fin             : %d\n", rep.stats.frm.rp_fin);
printf("    padding            : %d\n\n", rep.stats.frm.padding);
printf("  ERRORS DETECTED      : TOTAL=%d\n", rep.stats.frm_err.total);
printf("    ** physical layer errors\n");
printf("    pre_mis            : %d\n", rep.stats.frm_err.pre_mis);
printf("    fsd_mis            : %d\n", rep.stats.frm_err.fsd_mis);
printf("    fsd_unk            : %d\n", rep.stats.frm_err.fsd_unk);
printf("    fed_mis            : %d\n", rep.stats.frm_err.fed_mis);
printf("    ** data-link layer errors\n");
printf("    ctl_unk            : %d\n", rep.stats.frm_err.ctl_unk);
printf("    fcs_bad            : %d\n", rep.stats.frm_err.fcs_bad);
printf("    ** application layer errors\n");
printf("    pdu_bad            : %d\n", rep.stats.frm_err.pdu_bad);
printf("    len_bad            : %d\n", rep.stats.frm_err.len_bad);
printf("    ** library errors\n");
printf("    idx_loss           : %d\n", rep.stats.frm_err.idx_loss);

/**
 * Other tasks
 */
/* ... */

```

end:

```
/**  
 * Session exit  
 */  
/*...*/  
return err;  
}
```

### 4.3.4. sess\_stats\_clear

#### Description

Clears the statistics part of the diagnostic report.

#### Prototype

```
int fipcap_sess_stats_clear(struct fipcap_sess *sess)
```

#### Parameters

- *IN*
  - **sess:**  
Pointer to the target capture session.  
See `struct fipcap_sess`.
- *OUT* - None

#### Return Value

If successful, `fipcap_sess_stats_clear()` returns 0.

If unsuccessful, `fipcap_sess_stats_clear()` returns -1 and sets `errno` to one of the following values:

- **EINVAL:**  
Invalid input/output parameters.

### 4.3.5. sess\_start

#### Description

Starts the FIP capture.

#### Prototype

```
int fipcap_sess_start(struct fipcap_sess *sess)
```

#### Parameters

- *IN*
  - **sess:**  
Pointer to the session to be started by the coprocessor.
- *OUT* - None

#### Return Value

If successful, `fipcap_sess_start()` returns 0.

If unsuccessful, `fipcap_sess_start()` returns -1 and sets `errno` to one of the following values:

- **EINVAL:**  
Invalid input parameters.
- **FIPCAP\_ERR\_SESS\_NOT\_STOP:**  
The application context to be loaded is already active elsewhere.

### 4.3.6. sess\_stop

#### Description

Stops the FIP capture.



Use the `fipcap_sess_start()` function to start a new capture session.  
No need to reinitialize the session with `fipcap_sess_init()`.

#### Prototype

```
int fipcap_sess_stop(struct fipcap_sess *sess)
```

#### Parameters

- *IN*
  - **sess:**  
Pointer to the session to stop.  
See `struct fipcap_sess`.
- *OUT* - None

#### Return Value

If successful, `fipcap_sess_stop()` returns 0.

If unsuccessful, `fipcap_sess_stop()` returns -1 and sets `errno` to one of the following values:

- **EINVAL:**  
Invalid input/output parameters.

# Chapter 5. Structures

## 5.1. General

### 5.1.1. date

#### Description

Date structure.

#### Definition

```
struct fipcap_date {  
    union {  
        uint32_t raw;  
        struct {  
            uint8_t day;  
            uint8_t month;  
            uint8_t year;  
            uint8_t reserved;  
        } info;  
    };  
};
```

#### Members

Name	Type	Description
day	uint8_t	Day of the date.
month	uint8_t	Month of the date.
year	uint8_t	Year of the date.

## 5.1.2. version

### Description

Version structure.

### Definition

```
struct fipcap_version {
    union {
        uint32_t raw;
        struct {
            uint8_t patch;
            uint8_t minor;
            uint8_t major;
            uint8_t reserved;
        } info;
    };
    /* build date */
    struct fipcap_date date;
};
```

### Members

Name	Type	Description
patch	uint8_t	Patch version Marks bug fixes
minor	uint8_t	Minor version New features without break of compatibility
major	uint8_t	Major version Break of compatibility (ex: API changes)
date	struct fipcap_date	Build date See struct fipcap_date

## 5.2. Device

### 5.2.1. dev\_infos

#### Description

System information about the FIPWatcher board.

#### Definition

```
struct fw_dev_infos {
    uint32_t    index;
    uint32_t    sn;
    /* device general info */
    uint16_t    vid;
    uint16_t    did;
    uint16_t    ssvid;
    uint16_t    ssdid;
    /* physical BAR addresses */
    int         bar_cnt;
    uint64_t    bar_bsz[BAR_MAX];
    uint64_t    bar_base[BAR_MAX];
    /* irq infos*/
    int         irq_number;
    /* driver infos */
    uint32_t    drv_version;
    uint32_t    drv_date;
    /* fip infos */
    int         fip_bitrate;
};
```

#### Members

Name	Type	Description
index	uint32_t	Device index
sn	uint32_t	Serial Number
vid	uint16_t	Vendor ID
did	uint16_t	Device ID
ssvid	uint16_t	Subsystem Vendor ID
ssdid	uint16_t	Subsystem Device ID
bar_cnt	int	Number of BAR (Base Address Register) for this device
bar_bsz	uint64_t[BAR_MAX]	Size of the BAR areas



Name	Type	Description
bar_base	uint64_t[BAR_MAX]	Memory base start address for each area.
irq_number	int	IRQ number
drv_version	uint32_t	Driver version
drv_date	uint32_t	Driver built date
fip_bitrate	int	FIP bitrate (Hardware configuration)

## 5.3. Configuration

### 5.3.1. sess\_cfg


#### Description


Session configuration structure for the capture.

#### Definition

```
struct fipcap_sess_cfg {  
    /**  
     * General FIP settings  
     */  
    uint16_t padding;  
    /**  
     * Hardware access  
     */  
    struct fipcap_dev *dev;  
    unsigned long dpm_base_addr;  
    /**  
     * General Handlers  
     */  
    void *user_ctx;  
    void (* fipcap_frm_handler)(struct fipcap_sess *sess,  
                               struct fipcap_frm *frm);  
    void (* fipcap_diag_handler)(struct fipcap_sess *sess,  
                                 struct fipcap_sess_report *report);  
    void (* fipcap_error_handler)(struct fipcap_sess *sess,  
                                  enum fipcap_error_code error);  
    void (* fipcap_reset_handler)(struct fipcap_sess *sess);  
};
```

#### Members

Name	Type	Description
<b>padding</b>	<code>uint16_t</code>	Padding identifier <div> If this field is set to 0, the default padding identifier is 0x9080.</div>
<b>dev</b>	<code>struct fipcap_dev *</code>	PCI/PCIe device to bind with the capture. Opaque structure created by the <code>fipcap_device_open()</code> function.
<b>dpm_base_addr</b>	<code>unsigned long</code>	Dual-port memory area start address

Name	Type	Description
<b>user_ctx</b>	<code>void *</code>	User context pointer <i>[Optional]</i>
<b>fipcap_frm_handler</b>	<code>void (* handler) ( struct fipcap_sess *sess, struct fipcap_frm *frm )</code>	General frame handler. This handler is called for each frame received from the network. <i>[Optional]</i> See: <code>struct fipcap_sess</code> <code>struct fipcap_frm</code>
<b>fipcap_diag_handler</b>	<code>void (* handler) ( struct fipcap_sess *sess, struct fipcap_sess_report *report )</code>	General diagnosis session handler <i>[Optional]</i> See: <code>struct fipcap_sess</code> <code>struct fipcap_sess_report</code>  <div>  <p>The library automatically calls this handler at a fixed period:</p> <ul style="list-style-type: none"> <li>• @31.25Kbps: 1.6s</li> <li>• @1Mbps: 500ms</li> <li>• @2.5Mbps: 200ms</li> <li>• @5Mbps: 100ms</li> <li>• @12.5Mbps: 40ms</li> <li>• @25Mbps: 20ms</li> </ul> </div>
<b>fipcap_error_handler</b>	<code>void (* handler) ( struct fipcap_sess *sess, enum fipcap_error_code error )</code>	Internal library error handler <b>[Mandatory]</b> See: <code>struct fipcap_sess</code> <code>enum fipcap_error_code</code>
<b>fipcap_reset_handler</b>	<code>void (* handler) ( struct fipcap_sess *sess )</code>	Reset chip procedure handler <b>[Mandatory]</b> See <code>struct fipcap_sess</code>

## Remarks



### Hardware access

Two methods are provided to bind the FIP component with the library:

1. Use the `.dpm_base_addr` field to manually configure the FIPWatcher chip DPM (Dual-Port Memory) address access.

2. Use the `.dev` field to attach a FIPWatcher PCI/PCIe device.

*Note:* In this case no need to configure `.dpm_base_addr` field.

## 5.4. Frames

### 5.4.1. frm\_msg\_addr

#### Description

FIP message address.

#### Definition

```
struct fipcap_frm_msg_addr {
    union {
        uint32_t addr;
        struct {
            union {
                uint8_t seg;
                struct {
                    uint8_t seg_num:7;
                    uint8_t seg_group:1;
                };
            };
            union {
                uint16_t lsap;
                struct {
                    uint16_t lsap_num:15;
                    uint16_t lsap_group:1;
                };
            };
            uint8_t reserved;
        };
    };
};
```

#### Members

Name	Type	Description
<b>addr</b>	<code>uint32_t</code>	Data Link Layer (DLL) address
<b>seg</b>	<code>uint8_t</code>	FIP Segment field
<b>seg_num</b>	<code>uint8_t:7</code>	FIP Segment number
<b>seg_group</b>	<code>uint8_t:1</code>	Segment group: <ul style="list-style-type: none"><li>• 0: Individual segment</li><li>• 1: Group segment</li></ul>
<b>lsap</b>	<code>uint16_t</code>	Link Service Access Point (LSAP)

Name	Type	Description
lsap_num	uint16_t:15	LSAP number
lsap_group	uint16_t:1	LSAP group: <ul style="list-style-type: none"><li>• 0: Individual address</li><li>• 1: Group address</li></ul>
reserved	uint8_t	Reserved field

## 5.4.2. frm\_msg\_hdr

### Description

FIP Message header.

### Definition

```
struct fipcap_frm_msg_hdr {  
    struct fipcap_frm_msg_addr src;  
    struct fipcap_frm_msg_addr dst;  
};
```

### Members

Name	Type	Description
<b>src</b>	<code>struct fipcap_frm_msg_addr</code>	Message source address. See <code>struct fipcap_frm_msg_addr</code>
<b>dst</b>	<code>struct fipcap_frm_msg_addr</code>	Message destination address. See <code>struct fipcap_frm_msg_addr</code>

### 5.4.3. frm\_msg

#### Description

FIP Message container.

#### Definition

```
struct fipcap_frm_msg {  
    struct fipcap_frm_msg_hdr hdr;  
    uint16_t payload_bsz;  
    uint8_t *payload;  
};
```

#### Members

Name	Type	Description
<b>hdr</b>	<code>struct fipcap_frm_msg_hdr</code>	Message header. See <code>struct fipcap_frm_msg_hdr</code>
<b>payload_bsz</b>	<code>uint16_t</code>	Useful message byte size (without header). Range = [1;256]
<b>payload</b>	<code>uint8_t *</code>	Pointer to the useful message data.



#### 5.4.4. frm\_usrdata

##### Description

Generic frame container structure for the user payload.

##### Definition

```
struct fipcap_frm_usrdata {  
    enum fipcap_frm_usrdata_type type;  
    union {  
        struct fipcap_frm_var var;  
        struct fipcap_frm_var_list var_list;  
        struct fipcap_frm_msg msg;  
    };  
};
```

##### Members

Name	Type	Description
<b>type</b>	<code>enum fipcap_frm_usrdata_type</code>	Frame user-data type (Variable, Variable List or Message) See <code>enum fipcap_frm_usrdata_type</code> .
<ul style="list-style-type: none"><li>• <b>var</b></li><li>• <b>var_list</b></li><li>• <b>msg</b></li></ul>	<code>union {     struct fipcap_frm_var var;     struct fipcap_frm_var_list var_list;     struct fipcap_frm_msg msg; }</code>	Specialized container structure according to the <code>.type</code> field of the frame. See: <code>struct fipcap_frm_var</code> <code>struct fipcap_frm_var_list</code> <code>struct fipcap_frm_msg</code>

### 5.4.5. frm\_var\_list

#### Description

FIP aperiodic variable list container.

#### Definition

```
struct fipcap_frm_var_list {  
    uint32_t reserved[2];  
    uint16_t ids_cnt;  
    uint16_t *ids;  
};
```

#### Members

Name	Type	Description
<b>reserved</b>	uint8_t[2]	Reserved fields.
<b>ids_cnt</b>	uint16_t	Number of FIP identifiers contained in the list. Range = [1;64]
<b>ids</b>	uint16_t *	Pointer to the list of identifiers.

### 5.4.6. frm\_var

#### Description

FIP Variable container.

#### Definition

```
struct fipcap_frm_var {  
    uint8_t reserved[7];  
    uint8_t pdu;  
    uint16_t payload_bsz;  
    uint8_t *payload;  
};
```

#### Members

Name	Type	Description
<b>reserved</b>	uint8_t[7]	Reserved fields.
<b>pdu</b>	uint8_t	Protocol Data Unit (see enum <code>fipcap_frm_pdu_type</code> ).
<b>payload_bsz</b>	uint16_t	Useful variable byte size. Range = [1;126]
<b>payload</b>	uint8_t *	Pointer to the useful variable data.

## 5.4.7. frm

### Description

FIP frame container.

Contains all information about a FIP frame received from the network.

### Definition

```
struct fipcap_frm {
    uint64_t idx;
    struct {
        uint64_t epoch;
        uint32_t frame_nstime;
        uint32_t interframe_nstime;
    } time_info;
    uint8_t error;
    uint16_t raw_bsz;
    uint8_t *raw;
    /* dissected frame */
    uint8_t ctrl;
    uint16_t id;
    uint16_t fcs;
    struct fipcap_frm_usrdata usrdata;
};
```

### Members

Name	Type	Description
<b>idx</b>	uint64_t	Frame captured index
<b>epoch</b>	uint64_t	Epoch of the received frame.
<b>frame_nstime</b>	uint32_t	Frame duration in nanoseconds. (accuracy = +/-100ns)
<b>interframe_nstime</b>	uint32_t	Previous inter-frame duration in nanoseconds. (accuracy = +/-100ns)
<b>error</b>	uint8_t	Frame decoding error code. (see <code>enum fipcap_frm_err_code</code> )
<b>raw_bsz</b>	uint16_t	Raw data byte size
<b>raw</b>	uint8_t *	Pointer to the raw frame data.
<b>ctrl</b>	uint8_t	FIP Control byte. (see <code>enum fipcap_frm_ctrl_type</code> )
<b>id</b>	uint16_t	FIP identifier related to the frame.

Name	Type	Description
<b>fcs</b>	<code>uint16_t</code>	Frame Check Sequence of the FIP frame.
<b>usrdata</b>	<code>struct fipcap_frm_usrdata</code>	User payload data. (see <code>struct fipcap_frm_usrdata</code> )

## 5.5. Bus Arbiter

### 5.5.1. ba\_aper\_wind

#### Description

Structure describing an aperiodic bus arbiter macrocycle window.

#### Definition

```
struct fipcap_ba_aper_wind {  
    uint32_t end_ustime;  
};
```

#### Members

Name	Type	Description
<b>end_ustime</b>	<code>uint32_t</code>	End of aperiodic window in us. (relative to macrocycle start)

## 5.5.2. ba\_per\_wind

### Description

Structure describing a periodic bus arbiter macrocycle window.

### Definition

```
#define _FIPCAP_BA_WIND_REQ_CNT_MAX 1000

struct fipcap_ba_per_wind {
    int req_cnt;
    struct fipcap_ba_req req_list[_FIPCAP_BA_WIND_REQ_CNT_MAX];
};
```

### Members

Name	Type	Description
<b>req_cnt</b>	<code>int</code>	Number of periodic requests contained in the window.
<b>req_list</b>	<code>struct fipcap_ba_req req_list[]</code>	List of periodic requests. See <code>struct fipcap_ba_req</code>

### 5.5.3. ba\_mcycle

#### Description

Structure describing a bus arbiter macrocycle.

#### Definition

```
#define _FIPCAP_BA_WIND_CNT_MAX 50

struct fipcap_ba_mcycle {
    int wind_cnt;
    struct fipcap_ba_wind wind_list[_FIPCAP_BA_WIND_CNT_MAX];
    uint32_t duration_ustime;
};
```

#### Members

Name	Type	Description
<b>wind_cnt</b>	<code>int</code>	Number of windows inside the macrocycle.
<b>wind_list</b>	<code>struct fipcap_ba_wind[]</code>	Macrocycle windows list. See <code>struct fipcap_ba_wind</code>
<b>duration_ustime</b>	<code>uint32_t</code>	Macrocycle duration in microseconds.



### 5.5.4. ba\_req

#### Description

Structure describing a bus arbiter request.

#### Definition

```
struct fipcap_ba_req {  
    uint16_t type;  
    uint16_t id;  
};
```

#### Members

Name	Type	Description
<b>type</b>	uint16_t	Type of request from the bus arbiter See <code>enum fipcap_ba_id_type</code>
<b>id</b>	uint16_t	FIP identifier associated with the request

### 5.5.5. ba\_wind

#### Description

Generic structure describing a bus arbiter macrocycle window.

#### Definition

```
struct fipcap_ba_wind {  
    int type;  
    union {  
        struct fipcap_ba_per_wind per;  
        struct fipcap_ba_aper_wind aper;  
    };  
};
```

#### Members

Name	Type	Description
<b>type</b>	<code>int</code>	Macrocycle window type See <code>struct fipcap_ba_wind_type</code>
<ul style="list-style-type: none"><li><b>per</b></li><li><b>aper</b></li></ul>	<code>union {     struct fipcap_ba_per_wind per;     struct fipcap_ba_aper_wind aper; }</code>	Specialized container structure according to the <code>.type</code> field of the frame. See: <code>struct fipcap_ba_per_wind</code> <code>struct fipcap_ba_aper_wind</code>

## 5.6. Infos/Status/Report

### 5.6.1. sess\_ba\_info



#### Description

Bus arbiter informations detected during the capture.

#### Definition

```
struct fipcap_sess_ba_info {  
    uint8_t addr;  
    uint8_t mcycle_idx;  
    struct fipcap_ba_mcycle mcycle;  
};
```

#### Members

Name	Type	Description
addr	uint8_t	<div>Address of the FIP node where the bus arbiter is active.</div> <div> This information is only significant if the identifier 0x9003 is transmitted on the network.</div>
mcycle_idx	uint8_t	<div>Index of the running macrocycle.</div> <div> This information is only significant if the identifier 0x9003 is transmitted on the network.</div>
mcycle	struct fipcap_ba_mcycle	<div>Sniffed macrocycle.</div> <div>See <code>struct fipcap_ba_mcycle</code></div>

## 5.6.2. sess\_infos

### Description

Structure containing information about the capture session (session configuration, software versions/dates).

### Definition

```
struct fipcap_sess_infos {  
    struct fipcap_sess_cfg cfg;  
    uint32_t sn;  
    struct fipcap_version drv_version;  
    struct fipcap_version lib_version;  
};
```

### Members

Name	Type	Description
<b>cfg</b>	<code>struct fipcap_sess_cfg</code>	User configuration attached to the session See <code>struct fipcap_sess_cfg</code>
<b>sn</b>	<code>uint32_t</code>	Coprocessor serial number
<b>drv_version</b>	<code>struct fipcap_version</code>	Driver version See <code>struct fipcap_version</code>
<b>lib_version</b>	<code>struct fipcap_version</code>	Library version See <code>struct fipcap_version</code>

### 5.6.3. sess\_node\_info

#### Description

Informations about a FIP node detected during the capture.



These information is extracted from the FIP network using SM-MPS variables. The following identifiers must be transmitted on the FIP network for the data to be meaningful.

- Presence infos:
  - 0x9002
  - 0x14XY where XY is node address
- Identification:
  - 0x10XY where XY is node address
- Report:
  - 0x11XY where XY is node address

#### Definition

```
struct fipcap_sess_node_info {  
    /**  
     * Presence infos  
     */  
    int is_present;  
    uint64_t last_pres_epoch;  
    uint64_t last_abs_epoch;  
    int absence_cnt;  
    int ba_status;  
    int ba_prio;  
    /**  
     * Identification  
     */  
    char manufacturer_name[128];  
    char model_name[128];  
    uint8_t revision;  
    char tag_name[128];  
    uint8_t smmps_class;  
    char vendor_field[128];  
    /**  
     * Report  
     */  
    uint16_t rx_ok_ch1;  
    uint16_t rx_ok_ch2;  
    uint16_t rx_nok_ch1;  
    uint16_t rx_nok_ch2;  
};
```

```

union channel_status {
    uint16_t value;
    struct {
        uint16_t tx_quality_ch1:1;
        uint16_t tx_quality_ch2:1;
        uint16_t rx_quality_ch1:1;
        uint16_t rx_quality_ch2:1;
        uint16_t valid_ch1:1;
        uint16_t valid_ch2:1;
        uint16_t traffic_ch1:1;
        uint16_t traffic_ch2:1;
        uint16_t summary_ch1:1;
        uint16_t summary_ch2:1;
        uint16_t reserved:6;
    };
} channel_status;
};

```

## Members

Name	Type	Description
is_present	int	0: node is absent 1: node is present
last_pres_epoch	uint64_t	Epoch of last presence detection.
last_abs_epoch	uint64_t	Epoch of last absence detection.
absence_cnt	int	Absence detection counter.
ba_status	int	Local bus arbiter status. See <code>enum fipcap_ba_status</code>
ba_prio	int	Local bus arbiter priority. Within the range [0..15] (0: higher).
manufacturer_name	char[128]	Vendor (or manufacturer) name in ASCII.
model_name	char[128]	Model name in ASCII.
revision	uint8_t	Revision number (ex: 0x23 [v2.3]).
tag_name	char[128]	Tag name.
smmps_class	uint8_t	SM-MPS conformity class.
vendor_field	char[128]	Additional information of the vendor.
rx_ok_ch1	uint16_t	Frames correctly received on channel 1 by time unit.
rx_ok_ch2	uint16_t	Frames correctly received on channel 2 by time unit.

Name	Type	Description
rx_nok_ch1	uint16_t	Frames incorrectly received on channel 1 by time unit.
rx_nok_ch2	uint16_t	Frames incorrectly received on channel 2 by time unit.
channel_status	union channel_status	Medium status of the node. For more information, see <i>SM-MPS Report (0x11xy)</i>

## 5.6.4. sess\_report

### Description


Global report of the FIP capture.

### Definition

```
#define _FIPCAP_NODE_CNT_MAX 256

struct fipcap_sess_report {
    struct fipcap_sess_status status;
    struct fipcap_sess_stats stats;
    struct fipcap_sess_node_info node[_FIPCAP_NODE_CNT_MAX];
    struct fipcap_sess_ba_info ba;
};
```

### Members

Name	Type	Description
status	struct fipcap_sess_status	Session FSM and operation status See struct fipcap_sess_status
stats	struct fipcap_sess_stats	Capture statistics See struct fipcap_sess_stats
node	struct fipcap_sess_node_info[]	FIP nodes informations See struct fipcap_sess_node_info <div> The tab index corresponds to the address of the FIP node.</div>
ba	struct fipcap_sess_ba_info	Bus arbiter infos See struct fipcap_sess_ba_info



### 5.6.5. sess\_stats

#### Description

Gathers all statistical counters of the session.



The counters can be cleared at any time during capture using the `fipcap_sess_stats_clear()` function.

#### Definition

```
struct fipcap_sess_stats {  
  
    /* frame counters */  
    struct {  
        uint64_t total;  
        uint64_t id_dat;  
        uint64_t id_msg;  
        uint64_t id_rq1;  
        uint64_t id_rq2;  
        uint64_t rp_dat;  
        uint64_t rp_dat_msg;  
        uint64_t rp_dat_rq1;  
        uint64_t rp_dat_rq2;  
        uint64_t rp_dat_rq1_msg;  
        uint64_t rp_dat_rq2_msg;  
        uint64_t rp_rq1;  
        uint64_t rp_rq2;  
        uint64_t rp_msg_noack;  
        uint64_t rp_msg_ack_even;  
        uint64_t rp_msg_ack_odd;  
        uint64_t rp_ack_even_p;  
        uint64_t rp_ack_odd_p;  
        uint64_t rp_ack_even_m;  
        uint64_t rp_ack_odd_m;  
        uint64_t rp_fin;  
        uint64_t padding;  
    } frm;  
  
    /* frame error counters */  
    struct {  
        uint64_t total;  
        /* -> Physical Layer errors */  
        uint64_t pre_mis;  
        uint64_t fsd_mis;  
        uint64_t fsd_unk;  
        uint64_t fed_mis;  
        /* -> Data-Link Layer errors */  
        uint64_t ctl_unk;  
    } err;  
};
```


```

uint64_t fcs_bad;
/* -> Application Layer errors */
uint64_t pdu_bad;
uint64_t len_bad;
/* -> Library errors */
uint64_t idx_loss;
} frm_err;
};

```

## Members

Name	Type	Description
frm.total	uint64_t	Total frames captured without error
frm.id_dat	uint64_t	Variable transfer request
frm.id_msg	uint64_t	Message transfer request
frm.id_rq1	uint64_t	Urgent aperiodic list transfer request
frm.id_rq2	uint64_t	Normal aperiodic list transfer request
frm.rp_dat	uint64_t	Variable response
frm.rp_dat_msg	uint64_t	Variable response with message request
frm.rp_dat_rq1	uint64_t	Variable response with urgent aperiodic request
frm.rp_dat_rq2	uint64_t	Variable response with normal aperiodic request
frm.rp_dat_rq1_msg	uint64_t	Variable response with message and urgent aperiodic request
frm.rp_dat_rq2_msg	uint64_t	Variable response with message and normal aperiodic request
frm.rp_rq1	uint64_t	Response containing a list of urgent aperiodic requests
frm.rp_rq2	uint64_t	Response containing a list of normal aperiodic requests
frm.rp_msg_noack	uint64_t	Message response without acknowledgment request (SDN)
frm.rp_msg_ack_even	uint64_t	Message response with acknowledgment request (even parity)
frm.rp_msg_ack_odd	uint64_t	Message response with acknowledgment request (odd parity)
frm.rp_ack_even_p	uint64_t	Positive message acknowledgment (even parity)
frm.rp_ack_odd_p	uint64_t	Positive message acknowledgment (odd parity)
frm.rp_ack_even_m	uint64_t	Negative message acknowledgment (even parity)
frm.rp_ack_odd_m	uint64_t	Negative message acknowledgment (odd parity)
frm.rp_fin	uint64_t	End of message transaction

Name	Type	Description
frm.padding	uint64_t	Padding frames
frm_err.total	uint64_t	Total frames captured with an error
frm_err.pre_mis	uint64_t	Missing preamble (glitches on line)
frm_err.fsd_mis	uint64_t	Missing Frame Start Delimiter (FSD)
frm_err.fsd_unk	uint64_t	Unknown Frame Start Delimiter (FSD)
frm_err.fed_mis	uint64_t	Missing Frame End Delimiter (FED)
frm_err.ctl_unk	uint64_t	Unknown FIP control byte
frm_err.fcs_bad	uint64_t	Wrong Frame Check Sequence (FCS)
frm_err.pdu_bad	uint64_t	Unknown FIP Protocol Data Unit (PDU)
frm_err.len_bad	uint64_t	Bad FIP frame length (LEN)
frm_err.idx_loss	uint64_t	<p>Continuity break in the frame flow sent by the FIPWatcher coprocessor.</p> <div>  <p>Maybe the operating system is overloaded and does not unpack fast enough.</p> </div>

## 5.6.6. sess\_status

### Description

Capture session status information.

### Definition

```
struct fipcap_sess_status {  
    uint16_t state;  
    /* network information */  
    uint8_t frm_type;  
    uint8_t frm_bitrate;  
    /* capturing time */  
    uint64_t start_epoch;  
    uint64_t stop_epoch;  
    uint64_t duration_us;  
    /* fip time */  
    uint32_t turn_around_ustime;  
    uint32_t silence_ustime;  
};
```

### Members

Name	Type	Description
state	uint16_t	Session state (FSM: Finite State Machine) See <a href="#">enum fipcap_sess_state</a>
frm_type	uint8_t	Frame type detected See <a href="#">enum fipcap_frm_type</a>
frm_bitrate	uint8_t	FIP bitrate detected See <a href="#">enum fipcap_bitrate</a>
start_epoch	uint64_t	Epoch of start of the capture
stop_epoch	uint64_t	Epoch of stop of the capture
duration_us	uint64_t	Capture time in microseconds
turn_around_ustime	uint32_t	FIP turnaround time in microseconds
silence_ustime	uint32_t	FIP silence time in microseconds

### 5.6.7. sess

#### Description

Session capture object.

This is the handle structure to interact with a FIP sniffer.

#### Definition

```
struct fipcap_sess {  
    struct fipcap_sess_infos *infos;  
    void *priv;  
};
```

#### Members

Name	Type	Description
<b>infos</b>	<code>struct fipcap_sess_infos *</code>	Capture informations. See <code>struct fipcap_sess_infos</code>
<b>priv</b>	<code>void *</code>	Pointer to a reserved opaque structure

# Chapter 6. Enumerations

## 6.1. ba\_id\_type

### Description

Bus arbiter requests type.

### Definition

```
enum fipcap_ba_id_type {  
    _FIPCAP_BA_ID_TYPE_MIN,  
    FIPCAP_BA_ID_UNKNOWN,  
    FIPCAP_BA_ID_DAT,  
    FIPCAP_BA_ID_MSG,  
    FIPCAP_BA_ID_RQ,  
    _FIPCAP_BA_ID_TYPE_MAX,  
};
```

### Values

Constant	Description
FIPCAP_BA_ID_UNKNOWN	Unknown request.
FIPCAP_BA_ID_DAT	Variable request.
FIPCAP_BA_ID_MSG	Message request.
FIPCAP_BA_ID_RQ	Aperiodic variable list request.

## 6.2. ba\_status

### Description

Bus arbiter status for a FIP node.

### Definition

```
enum fipcap_ba_status {  
    _FIPCAP_BA_STATUS_MIN,  
    FIPCAP_BA_STATUS_NOT_SUPPORTED,  
    FIPCAP_BA_STATUS_NOT_ELIGIBLE,  
    FIPCAP_BA_STATUS_IDLE,  
    FIPCAP_BA_STATUS_ACTIVE,  
    _FIPCAP_BA_STATUS_MAX,  
};
```

### Values

Constant	Description
<b>FIPCAP_BA_STATUS_NOT_SUPPORTED</b>	The node does not support the bus arbiter service.
<b>FIPCAP_BA_STATUS_NOT_ELIGIBLE</b>	The node's bus arbiter is stopped; so it can't be eligible on the network.
<b>FIPCAP_BA_STATUS_IDLE</b>	The node's bus arbiter is in idle mode and is ready to take over if needed. Another bus arbiter is already active on the network.
<b>FIPCAP_BA_STATUS_ACTIVE</b>	The node's bus arbiter is active on the network.

## 6.3. ba\_wind\_type

### Description

Bus arbiter's macrocycle temporal window type.

### Definition

```
enum fipcap_ba_wind_type {  
    _FIPCAP_BA_WIND_TYPE_MIN,  
    FIPCAP_BA_WIND_UNKNOWN,  
    FIPCAP_BA_WIND_PER_VAR,  
    FIPCAP_BA_WIND_PER_MSG,  
    FIPCAP_BA_WIND_APER_VAR,  
    FIPCAP_BA_WIND_APER_MSG,  
    _FIPCAP_BA_WIND_TYPE_MAX,  
};
```

### Values

Constant	Description
FIPCAP_BA_WIND_UNKNOWN	Unknown window.
FIPCAP_BA_WIND_PER_VAR	Bus arbiter window for periodic variable requests.
FIPCAP_BA_WIND_PER_MSG	Bus arbiter window for periodic message requests.
FIPCAP_BA_WIND_APER_VAR	Bus arbiter window for aperiodic variable requests (sporadic requests).
FIPCAP_BA_WIND_APER_MSG	Bus arbiter window for aperiodic message requests (sporadic requests).



## 6.4. bitrate

### Description

FIP bitrate.

### Definition

```
enum fipcap_bitrate {  
    _FIPCAP_BITRATE_MIN = 1,  
    FIPCAP_BITRATE_31K25 = _FIPCAP_BITRATE_MIN,  
    FIPCAP_BITRATE_1M,  
    FIPCAP_BITRATE_2M5,  
    FIPCAP_BITRATE_5M,  
    FIPCAP_BITRATE_12M5,  
    FIPCAP_BITRATE_25M,  
    _FIPCAP_BITRATE_MAX,  
    _FIPCAP_BITRATE_UNKNOWN = 0,  
};
```

### Values

Constant	Value	Description
<code>_FIPCAP_BITRATE_MIN</code>	1	Minimum valid enum code for bitrate.
<code>FIPCAP_BITRATE_31K25</code>	1	FIP/WorldFIP bitrate at 31.25Kbps.
<code>FIPCAP_BITRATE_1M</code>	2	FIP/WorldFIP bitrate at 1Mbps.
<code>FIPCAP_BITRATE_2M5</code>	3	FIP/WorldFIP bitrate at 2.5Mbps.
<code>FIPCAP_BITRATE_5M</code>	4	FIP/WorldFIP bitrate at 5Mbps.
<code>FIPCAP_BITRATE_12M5</code>	5	FIP/WorldFIP bitrate at 12.5Mbps.
<code>FIPCAP_BITRATE_25M</code>	6	FIP/WorldFIP bitrate at 25Mbps.
<code>_FIPCAP_BITRATE_MAX</code>	7	Maximum enum code for bitrate.
<code>_FIPCAP_BITRATE_UNKNOWN</code>	0	Not supported or unknown FIP bitrate.

## 6.5. error\_code

### Description

Library and coprocessor error codes.

### Definition

```
enum fipcap_error_code {  
    _FIPCAP_ERR_CODE_MIN = 256,  
    FIPCAP_ERR_DEV_ALREADY_BIND = _FIPCAP_ERR_CODE_MIN,  
    FIPCAP_ERR_DEV_IRQ_HANDLER_STARTED,  
    FIPCAP_ERR_DEV_IRQ_HANDLER_STOPPED,  
    FIPCAP_ERR_DEV_DIAG_TASK_STARTED,  
    FIPCAP_ERR_DEV_DIAG_TASK_STOPPED,  
    FIPCAP_ERR_SESS_NOT_STOP,  
    FIPCAP_ERR_SESS_NOT_RUN,  
    FIPCAP_ERR_INVALID_CTX,  
    FIPCAP_ERR_SESS_HANDLER_MISSING,  
    _FIPCAP_ERR_CODE_MAX,  
};
```

### Values

Constant	Description
<b>FIPCAP_ERR_DEV_ALREADY_BIND</b>	The device is already bound to another FIP capture session.
<b>FIPCAP_ERR_DEV_IRQ_HANDLER_STARTED</b>	IRQ handler is already started.
<b>FIPCAP_ERR_DEV_IRQ_HANDLER_STOPPED</b>	IRQ handler is already stopped.
<b>FIPCAP_ERR_DEV_DIAG_TASK_STARTED</b>	Diagnostic task is already started.
<b>FIPCAP_ERR_DEV_DIAG_TASK_STOPPED</b>	Diagnostic task is already stopped.
<b>FIPCAP_ERR_SESS_NOT_STOP</b>	FIP capture session is currently running.
<b>FIPCAP_ERR_SESS_NOT_RUN</b>	FIP capture session is currently stopped.
<b>FIPCAP_ERR_INVALID_CTX</b>	Objects do not belong to the same session context.
<b>FIPCAP_ERR_SESS_HANDLER_MISSING</b>	Reset/Error handlers are mandatory.

## 6.6. frm\_ctrl\_type

### Description

Frame Control byte types.



### Definition

```
enum fipcap_frm_ctrl_type {  
    FIPCAP_FRM_CTRL_ID_DAT,  
    FIPCAP_FRM_CTRL_ID_MSG,  
    FIPCAP_FRM_CTRL_ID_RQ1,  
    FIPCAP_FRM_CTRL_ID_RQ2,  
    FIPCAP_FRM_CTRL_RP_DAT,  
    FIPCAP_FRM_CTRL_RP_DAT_MSG,  
    FIPCAP_FRM_CTRL_RP_DAT_RQ1,  
    FIPCAP_FRM_CTRL_RP_DAT_RQ2,  
    FIPCAP_FRM_CTRL_RP_DAT_RQ1_MSG,  
    FIPCAP_FRM_CTRL_RP_DAT_RQ2_MSG,  
    FIPCAP_FRM_CTRL_RP_RQ1,  
    FIPCAP_FRM_CTRL_RP_RQ2,  
    FIPCAP_FRM_CTRL_RP_MSG_NOACK,  
    FIPCAP_FRM_CTRL_RP_MSG_ACK_EVEN,  
    FIPCAP_FRM_CTRL_RP_MSG_ACK_ODD,  
    FIPCAP_FRM_CTRL_RP_ACK_EVEN_P,  
    FIPCAP_FRM_CTRL_RP_ACK_ODD_P,  
    FIPCAP_FRM_CTRL_RP_ACK_EVEN_M,  
    FIPCAP_FRM_CTRL_RP_ACK_ODD_M,  
    FIPCAP_FRM_CTRL_RP_FIN,  
};
```

### Values

Constant	Value	Description
<b>FIPCAP_FRM_CTRL_ID_DAT</b>	0x03	Bus arbiter request for a variable.
<b>FIPCAP_FRM_CTRL_ID_MSG</b>	0x05	Bus arbiter request for a message.
<b>FIPCAP_FRM_CTRL_ID_RQ1</b>	0x29	Urgent request from the bus arbiter for a list of aperiodic variables.
<b>FIPCAP_FRM_CTRL_ID_RQ2</b>	0x09	Normal request from the bus arbiter for a list of aperiodic variables.
<b>FIPCAP_FRM_CTRL_RP_DAT</b>	0x02	Response from a node containing a variable.
<b>FIPCAP_FRM_CTRL_RP_DAT_MSG</b>	0x06	Response from a node containing a variable and a message request.

Constant	Value	Description
FIPCAP_FRM_CTRL_RP_DAT_RQ1	0x2a	Response from a node containing a variable and an urgent aperiodic request.
FIPCAP_FRM_CTRL_RP_DAT_RQ2	0x0a	Response from a node containing a variable and an aperiodic request.
FIPCAP_FRM_CTRL_RP_DAT_RQ1_MSG	0x2e	Response from a node containing a variable, an urgent aperiodic request and a message request.
FIPCAP_FRM_CTRL_RP_DAT_RQ2_MSG	0x0e	Response from a node containing a variable, an aperiodic request and a message request.
FIPCAP_FRM_CTRL_RP_RQ1	0x28	Response to an urgent aperiodic request containing a list of identifiers to query.
FIPCAP_FRM_CTRL_RP_RQ2	0x08	Response to a normal aperiodic request containing a list of identifiers to query.
FIPCAP_FRM_CTRL_RP_MSG_NOACK	0x04	Response from a node containing a message. The response does not require an acknowledgement from the recipient.
FIPCAP_FRM_CTRL_RP_MSG_ACK_EVEN	0x14	Response from a node containing a message. The response requires an even parity acknowledgement from the recipient.
FIPCAP_FRM_CTRL_RP_MSG_ACK_ODD	0x94	Response from a node containing a message. The response requires an odd parity acknowledgement from the recipient.
FIPCAP_FRM_CTRL_RP_ACK_EVEN_P	0x30	Positive acknowledgment (even parity) transmitted by the recipient of the message.
FIPCAP_FRM_CTRL_RP_ACK_ODD_P	0xb0	Positive acknowledgment (odd parity) transmitted by the recipient of the message.
FIPCAP_FRM_CTRL_RP_ACK_EVEN_M	0x10	Negative acknowledgment (even parity) transmitted by the recipient of the message.  <div>  <p>The recipient node has received the message but cannot process it. Its message queue may be full.</p> </div>

Constant	Value	Description
FIPCAP_FRM_CTRL_RP_ACK_ODD_M	0x90	<div><div>Negative acknowledgment (odd parity) transmitted by the recipient of the message.</div><div><div>The recipient node has received the message but cannot process it. Its message queue may be full.</div></div></div>
FIPCAP_FRM_CTRL_RP_FIN	0x40	<div><div>End of a message transaction.</div><div><div>This frame is sent by the transmitter of the message to close the transaction.</div></div></div>

## 6.7. frm\_err\_code

### Description

Frame decoding error codes.

### Definition

```
enum fipcap_frm_err_code {  
    _FIPCAP_FRM_OK = 0,  
    _FIPCAP_FRM_ERR_MIN,  
    FIPCAP_FRM_ERR_PREAMBLE_MISSING = _FIPCAP_FRM_ERR_MIN,  
    FIPCAP_FRM_ERR_FSD_MISSING,  
    FIPCAP_FRM_ERR_FSD_UNKNOWN,  
    FIPCAP_FRM_ERR_FED_MISSING,  
    FIPCAP_FRM_ERR_CTL_UNKNOWN,  
    FIPCAP_FRM_ERR_FCS_BAD,  
    FIPCAP_FRM_ERR_PDU_BAD,  
    FIPCAP_FRM_ERR_LEN_BAD,  
    _FIPCAP_FRM_ERR_MAX,  
};
```

### Values

Constant	Description
<b>_FIPCAP_FRM_OK</b>	No error.
<b>FIPCAP_FRM_ERR_PREAMBLE_MISSING</b>	Preamble is missing.
<b>FIPCAP_FRM_ERR_FSD_MISSING</b>	Frame Start Delimiter is missing.
<b>FIPCAP_FRM_ERR_FSD_UNKNOWN</b>	Frame Start Delimiter is unknown.
<b>FIPCAP_FRM_ERR_FED_MISSING</b>	Frame End Delimiter is missing.
<b>FIPCAP_FRM_ERR_CTL_UNKNOWN</b>	Frame control unknown.
<b>FIPCAP_FRM_ERR_FCS_BAD</b>	Frame check sequence is bad.
<b>FIPCAP_FRM_ERR_PDU_BAD</b>	Unknown PDU.
<b>FIPCAP_FRM_ERR_LEN_BAD</b>	Frame length is bad.

## 6.8. frm\_pdu\_type


### Description

Frame PDU (Protocol Data Unit) types.

### Definition

```
enum fipcap_frm_pdu_type {  
    FIPCAP_FRM_PDU_MPS,  
    FIPCAP_FRM_PDU_SMMPS,  
};
```

### Values

Constant	Value	Description
FIPCAP_FRM_PDU_MPS	0x40	MPS type.
FIPCAP_FRM_PDU_SMMPS	0x50	<div><div>SM-MPS type.</div><div><div></div><div><p><b>System Management - Manufacturing Periodic/aperiodic Services</b></p><p>This type of frame informs about the general state of the active nodes on the network.</p><p>See the FIP identifiers: (XY is the hexadecimal node address)</p><ul style="list-style-type: none"><li>• ID_DAT(0x10XY)</li><li>• ID_DAT(0x11XY)</li><li>• ID_DAT(0x14XY)</li><li>• ID_DAT(0x9002)</li><li>• ID_DAT(0x9003)</li></ul></div></div></div>

## 6.9. frm\_type



### Description

Frame type.

### Definition

```
enum fipcap_frm_type {  
    _FIPCAP_FRM_TYPE_MIN = 1,  
    FIPCAP_FRM_FIP = _FIPCAP_FRM_TYPE_MIN,  
    FIPCAP_FRM_WORLDFIP,  
    _FIPCAP_FRM_TYPE_MAX,  
    _FIPCAP_FRM_TYPE_UNKNOWN = 0,  
};
```

### Values

Constant	Value	Description
<b>_FIPCAP_FRM_TYPE_MIN</b>	1	Minimum valid enum code for frame type.
<b>FIPCAP_FRM_FIP</b>	1	FIP frame type.  <div> <i>Type of frame delimiters and CRC</i> UTE (Union Technique de l'Electricité)</div>
<b>FIPCAP_FRM_WORLDFIP</b>	2	WorldFIP frame type.  <div> <i>Type of frame delimiters and CRC</i> IEC (International Electrotechnical Commission)</div>
<b>_FIPCAP_FRM_TYPE_MAX</b>	3	Maximum enum code for frame type.
<b>_FIPCAP_FRM_TYPE_UNKNOWN</b>	0	Unknown frame type.



## 6.10. frm\_usrdata\_type

### Description

Type of user payload data contained in the frame.

### Definition

```
enum fipcap_frm_usrdata_type {  
    _FIPCAP_FRM_USRDATA_TYPE_MIN,  
    FIPCAP_FRM_USRDATA_NONE,  
    FIPCAP_FRM_USRDATA_VAR,  
    FIPCAP_FRM_USRDATA_VAR_LIST,  
    FIPCAP_FRM_USRDATA_MSG,  
    _FIPCAP_FRM_USRDATA_TYPE_MAX,  
    _FIPCAP_FRM_USRDATA_TYPE_UNKNOWN,  
};
```

### Values

Constant	Description
FIPCAP_FRM_USRDATA_NONE	The frame does not contain user data (application layer).
FIPCAP_FRM_USRDATA_VAR	The frame contains a user data of type <i>variable</i> .
FIPCAP_FRM_USRDATA_VAR_LIST	The frame contains a user data of type <i>variable identifier list</i> .
FIPCAP_FRM_USRDATA_MSG	The frame contains a user data of type <i>message</i> .

## 6.11. sess\_state

### Description

FIP capture session state.

### Definition

```
enum fipcap_sess_state {  
    FIPCAP_SESS_STATE_INITIAL = 0,  
    FIPCAP_SESS_STATE_READY,  
    FIPCAP_SESS_STATE_RUNNING,  
    _FIPCAP_SESS_STATE_MAX,  
};
```

### Values

Constant	Value	Description
FIPCAP_SESS_STATE_INITIAL	0	Initial state. No config loaded.
FIPCAP_SESS_STATE_READY	1	The user context of the capture is loaded. Here, the sniffer is in stopped state, and is ready to start.
FIPCAP_SESS_STATE_RUNNING	2	Running state. Sniffer is active and catches network frames.
_FIPCAP_SESS_STATE_MAX	3	Max session state number.

# Appendix A: SM-MPS variables

The network management variables SM-MPS are automatically created and internally managed by each FIP node of the network.

These variables are useful to know the general state of the network as well as to get information about a particular FIP node.

## A.1. Identification - 0x10XY

### Description

The variable attached to the ID number 0x10XY (where XY is the node address) is called the *Identification* variable.

Each node produces this variable, and its payload allows to clearly identify the node on the network.

### Frame Format

Bytes	Description
0x50	PDU type (SM-MPS)
0xZZ	PDU length (must not exceed 126 bytes)
0x80	Manufacturer name field
0xZZ	Manufacturer name field length
0xZZ	First character for manufacturer name
...	...
0xZZ	Last character for manufacturer name
0x81	Model name field
0xZZ	Model name field length
0xZZ	First character for model name
...	...
0xZZ	Last character for model name
0x82	Revision field
0x01	Revision field length
0xZZ	Revision number (ex: 0x10 for v1.0)
0x83	Device tag name field <i>[Optional]</i>
0xZZ	Device tag name field length
0xZZ	First character for tag name
...	...

Bytes	Description
0xZZ	Last character for tag name
0x84	SM-MPS secondary function field
0x01	SM-MPS secondary function field length
0x10 (fixed)	SM-MPS secondary function field value: <ul style="list-style-type: none"> <li>• bit0: set to 1 if loading supported</li> <li>• bit1: set to 1 if remote reading supported</li> <li>• bit2: set to 1 if remote control supported</li> <li>• bit3: set to 1 if remote checking supported</li> <li>• bit4: set to 1 if report supported</li> <li>• the other bits are always set to 0</li> </ul>
0x8A	Vendor field <i>[Optional]</i>
0xZZ	Vendor field length
0xZZ	First byte for vendor field
...	...
0xZZ	Last byte for vendor field

## A.2. Report - 0x11XY



### Description

The variable attached to the ID number 0x11XY (where XY is the node address) is called the *Report* variable.

Each node produces this variable, and its payload contains various node-specific status counters (rx faults, number of transactions, ...).

### Frame Format

Bytes	Description
0x50	PDU type (SM-MPS)
0x0F	PDU length
0x50	Tag for counter of frames correctly received on channel 1 by Time Unit <sup>(1)</sup>
0xZZ	Counter Most Significant Byte
0xZZ	Counter Least Significant Byte
0x51	Tag for counter of frames correctly received on channel channel 2 by Time Unit <sup>(1)</sup>
0xZZ	Counter Most Significant Byte

Bytes	Description
0xZZ	Counter Least Significant Byte
<b>0x52</b>	Tag for counter of frames incorrectly received on channel 1 by Time Unit <sup>(1)</sup>
0xZZ	Counter Most Significant Byte
0xZZ	Counter Least Significant Byte
<b>0x53</b>	Tag for counter of frames incorrectly received on channel 2 by Time Unit <sup>(1)</sup>
0xZZ	Counter Most Significant Byte
0xZZ	Counter Least Significant Byte
<b>0x54</b>	Tag for channel status
0xZZ	<p>Channel status (MSB)</p> <ul style="list-style-type: none"> <li>bit1: Synthesis for channel 2 (1: OK, 0: NOK)</li> </ul> <div>  <p>Result of the binary operation for channel status LSB: b1 &amp; b3 &amp; b5 &amp; b7</p> </div> <ul style="list-style-type: none"> <li>bit0: Synthesis for channel 1 (1: OK, 0: NOK)</li> </ul> <div>  <p>Result of the binary operation for channel status LSB: b0 &amp; b2 &amp; b4 &amp; b6</p> </div>
0xZZ	<p>Channel status (LSB)</p> <ul style="list-style-type: none"> <li>bit7: Traffic<sup>(2)</sup> on channel 2 (1: OK, 0: NOK)</li> <li>bit6: Traffic<sup>(2)</sup> on channel 1 (1: OK, 0: NOK)</li> <li>bit5: Validity<sup>(3)</sup> on channel 2 (1: OK, 0: NOK)</li> <li>bit4: Validity<sup>(3)</sup> on channel 1 (1: OK, 0: NOK)</li> <li>bit3: RX quality<sup>(4)</sup> on channel 2 (1: OK, 0: NOK)</li> <li>bit2: RX quality<sup>(4)</sup> on channel 1 (1: OK, 0: NOK)</li> <li>bit1: TX quality<sup>(4)</sup> on channel 2 (1: OK, 0: NOK)</li> <li>bit0: TX quality<sup>(4)</sup> on channel 1 (1: OK, 0: NOK)</li> </ul>

### Remarks

- (1) Time Unit corresponds to the diagnostic period of the medium of the remote node.
- (2): Traffic signal is the measure of frames correctly received and/or transmitted on/by the channel by Time Unit<sup>(1)</sup>. To be OK, you need:
  - (TX frames OK + RX frames OK) > 0
- (3): Validity of a channel is given by the medium redundancy component of the remote node.
- (4): RX/TX quality is the measure of the error rate on the channel by Time Unit<sup>(1)</sup>.

To be OK, you need:

- (RX frames faults / RX frames OK) < Threshold
- (TX frames faults / TX frames OK) < Threshold

## A.3. Presence - 0x14XY

### Description

The variable attached to the ID number 0x14XY (where XY is the node address) is called the *Presence* variable.

Each node produces this variable and thus informs the other nodes of the FIP network of its presence.

### Frame Format

Bytes	Description
0x50	PDU type (SM-MPS)
0x05	PDU length
0x80	Presence parameter
0x03	Presence parameter length
0xZZ	Identification characteristics. See: <a href="#">PDU length for 0x10XY</a> <ul style="list-style-type: none"><li>• 0: Unknown length</li><li>• 1 to 0xFE: Known length</li><li>• 0xFF: No identification variable</li></ul>
0x00	Reserved
0xZZ	Bus Arbiter global status (about the producer). <ul style="list-style-type: none"><li>• bit4-bit7: BA status:<ul style="list-style-type: none"><li>◦ 0: BA not supported</li><li>◦ 1: BA not eligible (stopped mode)</li><li>◦ 2: BA idle (on standby)</li><li>◦ 3: BA active (master node)</li></ul></li><li>• bit0-bit3: BA priority (if BA is supported). Range is [0;15], with 0 the highest priority.</li></ul>

## A.4. Presence Check - 0x9002

## Description

The variable attached to the ID number 0x9002 is called the *Presence Check* or *Present List* variable.

This variable is produced by the master node (Bus arbiter active).

It summarizes in a single variable the whole list of nodes presence on the network.

## Frame Format

Bytes	Description
0x50	PDU type (SM-MPS)
0x44	PDU length
0x80	Channel 1 presence list
0x20	Channel 1 presence list length
0xZZ	Status of nodes address 0 to 7. (0: absent, 1: present) <ul style="list-style-type: none"> <li>• bit0: node 0</li> <li>• ...</li> <li>• bit7: node 7</li> </ul>
...	...
0xZZ	Status of nodes address 248 to 255. (0: absent, 1: present) <ul style="list-style-type: none"> <li>• bit0: node 248</li> <li>• ...</li> <li>• bit7: node 255</li> </ul>
0x81	Channel 2 presence list
0x20	Channel 2 presence list length
0xZZ	Status of nodes address 0 to 7. (0: absent, 1: present) <ul style="list-style-type: none"> <li>• bit0: node 0</li> <li>• ...</li> <li>• bit7: node 7</li> </ul>
...	...

Bytes	Description
0xZZ	Status of nodes address 248 to 255. (0: absent, 1: present) <ul style="list-style-type: none"><li>• bit0: node 248</li><li>• ...</li><li>• bit7: node 255</li></ul>

## A.5. BA synchronization - 0x9003

### Description

The variable attached to the ID number 0x9003 is called the *BA synchronisation* variable.

This variable is produced by the master node, and indicates the current status of the Bus Arbiter on the network.

### Frame Format

Bytes	Description
0x50	PDU type (SM-MPS)
0x04	PDU length
0x80	BA synchro parameter
0x02	BA synchro parameter length
0xZZ	Macrocycle number of current BA program
0xZZ	Physical node address of the master



## Appendix B: Glossary of acronyms

<b>BA</b>	Bus Arbiter
<b>DLL</b>	Data Link Layer
<b>DSAP</b>	Destination Service Access Point
<b>LLC</b>	Logical Link Control
<b>LSAP</b>	Link Service Access Point
<b>MPS</b>	Manufacturing Periodic/aperiodic Services
<b>PDU</b>	Protocol Data Unit
<b>SDA</b>	Send Data with Acknowledgement
<b>SDN</b>	Send Data with No acknowledgement
<b>SM-MPS</b>	System Management Manufacturing Periodic/aperiodic Services
<b>SSAP</b>	Source Service Access Point

## Appendix C: Revision History

Revision	Changes	Authors	Date
1.0.0	First version	MC	2023-02-07