VAPI CORE

Generated by Doxygen 1.7.1

Sat May 11 2013 16:15:29

Contents

Chapter 1

Main Page

VAPI CORE is a lightweight IPC (Inter Process Communication) library to executed APIs of other processes on the Linux OS, which is written by C language and distributed under the BSD (Berkeley Standard Distribution) license.

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

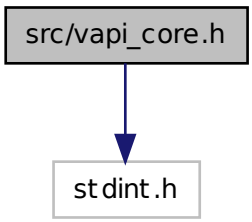
Chapter 3

File Documentation

3.1 src/vapi_core.h File Reference

```
#include <stdint.h>
```

Include dependency graph for vapi_core.h:



Functions

- int32_t vapi_core_open (uint16_t dstport)

  "vapi_core_open()" establish a local TCP connection with the sub process listening on the port number specified by "dstport". It can be several times called with the same "dstport".

- int32_t vapi_core_close (int32_t fd)

  "vapi_core_close()" close the local TCP connection with the sub process.

- int32_t vapi_core_invoke (int32_t fd, int32_t api_id, void *p_arg, uint32_t arg_len)

  "vapi_core_invoke()" requests executing a API function specified by the "api_id" to the sub module, and receives the acknowledgement from the sub module synchronously.

3.1.1 Function Documentation

3.1.1.1 int32_t vapi_core_close (int32_t *fd*)

"vapi_core_close()" close the local TCP connection with the sub process.

Parameters

[in] ***fd*** The descriptor.

Returns

0 for success, and -1 for error.

3.1.1.2 int32_t vapi_core_invoke (int32_t *fd*, int32_t *api_id*, void * *p_arg*, uint32_t *arg_len*)

"vapi_core_invoke()" requests executing a API function specified by the "api_id" to the sub module, and receives the acknowledgement from the sub module synchronously.

Parameters

[in] ***fd*** The descriptor.

[in] ***api_id*** The API function ID to be executed.

[in,out] ***p_arg*** The pointer to the arguments.

[in] ***arg_len*** The length of the arguments.

Returns

0 for success, and -1 for error.

3.1.1.3 int32_t vapi_core_open (uint16_t *dstport*)

"vapi_core_open()" establish a local TCP connection with the sub process listening on the port number specified by "dstport". It can be several times called with the same "dstport".

Parameters

[in] ***dstport*** The destination port number listened by the sub process.

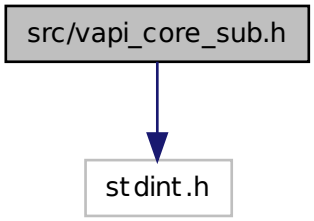
Returns

It returns a descriptor. If error happened, -1 will return.

3.2 src/vapi_core_sub.h File Reference

```
#include <stdint.h>
```

Include dependency graph for vapi_core_sub.h:



Typedefs

- typedef int(∗ vapi_core_sub_handler_t)(int32_t api_id, void ∗p_arg, uint32_t arg_len, void ∗p_-cookie)

  "vapi_core_sub_handler_t" is the type of handler function to be called when an invoked request is received from the host side.

Functions

- int32_t vapi_core_sub_open (uint16_t port, vapi_core_sub_handler_t handler, const void ∗p_-cookie)

  "vapi_core_sub_open()" binds to the specified "port" and listen on it. Then it creates a thread of accepting from the host side.

- int32_t vapi_core_sub_close (int32_t fd)

  "vapi_core_sub_close()" close the listened socket. However the accepted sockets would not be closed. This sockets should be closed by the host side.

- int32_t vapi_core_sub_get_port (int32_t fd, uint16_t ∗p_port)

  "vapi_core_sub_get_port()" gets the listened port number.

3.2.1 Typedef Documentation

3.2.1.1 typedef int(∗ vapi_core_sub_handler_t)(int32_t api_id, void ∗p_arg, uint32_t arg_len, void ∗p_cookie)

"vapi_core_sub_handler_t" is the type of handler function to be called when an invoked request is received from the host side.

Parameters

`[in]` ***api_id*** The API function ID to be executed.

[in,out] ***p_arg*** The pointer to the arguments.

[in] ***arg_len*** The length of the arguments.

[in,out] ***p_cookie*** The pointer to the user data.

Returns

0 for success, and the other values for handling error. If not 0, vapi_core_invoke() of the host side will return error.

3.2.2 Function Documentation

3.2.2.1 int32_t vapi_core_sub_close (int32_t *fd*)

"vapi_core_sub_close()" close the listened socket. However the accepted sockets would not be closed. This sockets should be closed by the host side.

Parameters

[in] ***fd*** The descriptor.

Returns

0 for success, and -1 for error.

3.2.2.2 int32_t vapi_core_sub_get_port (int32_t *fd*, uint16_t * *p_port*)

"vapi_core_sub_get_port()" gets the listened port number.

Parameters

[in] ***fd*** The descriptor.

[out] ***p_port*** The pointer of port.

Returns

0 for success, and -1 for error.

3.2.2.3 int32_t vapi_core_sub_open (uint16_t *port*, vapi_core_sub_handler_t *handler*, const void * *p_cookie*)

"vapi_core_sub_open()" binds to the specified "port" and listen on it. Then it creates a thread of accepting from the host side.

Parameters

[in] ***port*** The port number to be listened. If 0, kernel will select an available port automatically, which can be gotten by vapi_core_sub_get_port() .

[in] ***handler*** The handler function to be called when an invoked request is received from the host side.

[in] ***p_cookie*** The pointer to the user data.

Returns

It returns a descriptor. If error happened, -1 will return.

Index