IPC Wrapper

Axe Inc.

Version - Feb. 23, 2004

(English Translation by Osamu MIHARA, Oct. 22, 2006)

1. General

Usually, Vector printer driver is linked to the glue code as a shared library. However, in case which vector driver contains know-hows cannot be opened or confidential item, implementing vector printer driver as an independent process to avoid license problem. For this purpose, vector driver is allowed to implemented as a IPC extension type. IPC Wrapper is a module in order to separate the process of vector driver library from glue code.

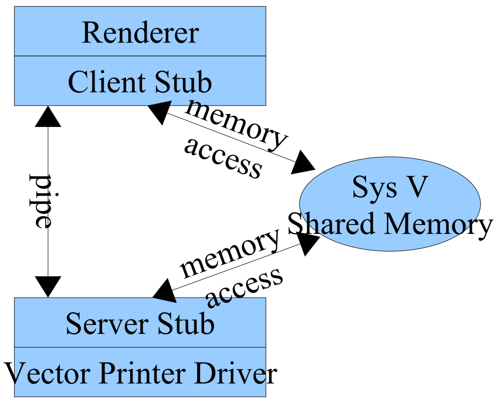
The IPC Wrapper provides vector driver API to glue code, and calls vector driver with vector driver API. As a result, glue code and vector driver are not needed to modify source code.

The IPC wrapper is provides under a MIT-like license.

2. Protocol

2.1 General

The right figure shows the module structure between renderer and vector printer driver.



RPC client stub is linked to render process. On the other hand, server stub is linked to vector driver. The interface between render and client stub, and between server stub and vector driver are vector printer driver API (VPDAPI).

When OpenPrinter() is called from the renderer, client stub executes printer driver process and do the rest of the process of OpenPrinter() after establishing communication route using pipe.

Each API call is constructed as packets by client stud, buffered, then sent out to the pipe. Server stub read the packets out from pipe, analyses then calls corresponding VPDAPI, if necessary, returns result via reverse route. Large data, such as images, are sent via shared memory of System V IPC.

Buffer of client stub is flushed when a API, which needs return value, is called. When an API, which does not take return value, is called, it does not executed immediately on the printer driver side. In this case, return value of the API on the renderer side is a tentative one. EndJob(), EndDoc() and EndPage() functions are always returned after the executions on the printer driver side are finished.

2.2 Packet Format

The byte order in packets are CPU native. There are two types of packets – one is request packet from client to server, the other is response packet from server to client. The format of each packet are as follow

Request Packet			
Packet Size	Sequence Number	API function number	Parameter 1, ..., Parameter N

Response Packet			
Packet Size	Sequence Number	API function number	Return Value 1, ..., Return Value N

Format of each item is as follow:

Item	Type	Description
Packet Size	4 bytes integer	Packet Size in bytes exclude itself
Sequence Number	4 bytes integer	A sequence number incremented when a packet is created. This is used for error checking.
API function number	4 bytes integer	A number designating API function. In the response packet, if it is a negative number, an error is returned and the absolute value shows the error number.
Parameter 1, ..., Parameter N	-	Parameters for each function, excluding one is used for output.
Return Value 1, ..., Return Value N	-	Return values for each function. Return value 1 is the return value of the function, and the rest values are returned values via parameters. In case of error response, return value 1 shows the API function number where the error is occurred.

2.3 Data Types

Type of the data in packets are defined by the original API data types.

(1) Fundamental C types – Same as is the format in memory. Int is a 4 bytes integer, for example.

(2) enum – same as the format in memory.

(3) struct – a struct which member does not contains pointer takes same format as is in memory. If a member is a pointer, the contents of the member is included in the packet in order.

(4) pointer – places any of follows.

Expansion Type	
1 byte “0”	The contents which are pointed by the pointer is expanded. Whereas, padding data is placed in order to the offset of the start address becomes multiply of 4.
Shared Memory Type	
1 byte “1”	Shared Memory ID
NULL Pointer	
1 byte “2”	-

For the shared memory type, contents pointed by the pointer is expanded in the shared memory designated by the ID. This shared memory is detached when the first return value is returned by the called API function.2.4

2.4 Examples

QueryColorSpace

Request Packet

Size	Sequence Number	API Function Number	printerContext	pnum
17	123	15	156	0, 128

Response Packet

Size	Sequence Number	API Function Number	pcspace	pnum
34	123	15	0, 1, 2, 3, 4, 5	0, 5

5. Source code

Source code files related IPC Wrapper are located under vector-printer directory.

- opvp_rpc_client.c – client side code
- opvp_rpc_client.h – header file for client code
- opvp_rpc_core.c – RPC core code
- opvp_rpc_core.h – header file for RPC core code

- opvp_rpc_server.c – server side code
- opvp_rpc_server.h – header file for server side code
- opvp_test.c – testing code
- opvptest.sh – testing script
- opvp_null.c – dummy vector driver code for testing
- opvp_null.h – header for dummy vector driver code for testing

4. License

The IPC wrapper code is provided under following license.