Study of the PCM Plug-in and Status Monitoring Interface

TORATANI Yasumasa
<toratani.yasumasa@canon.co.jp>
OpenPrinting WG Japan/Asia
Canon Inc.

15-17 November 2004

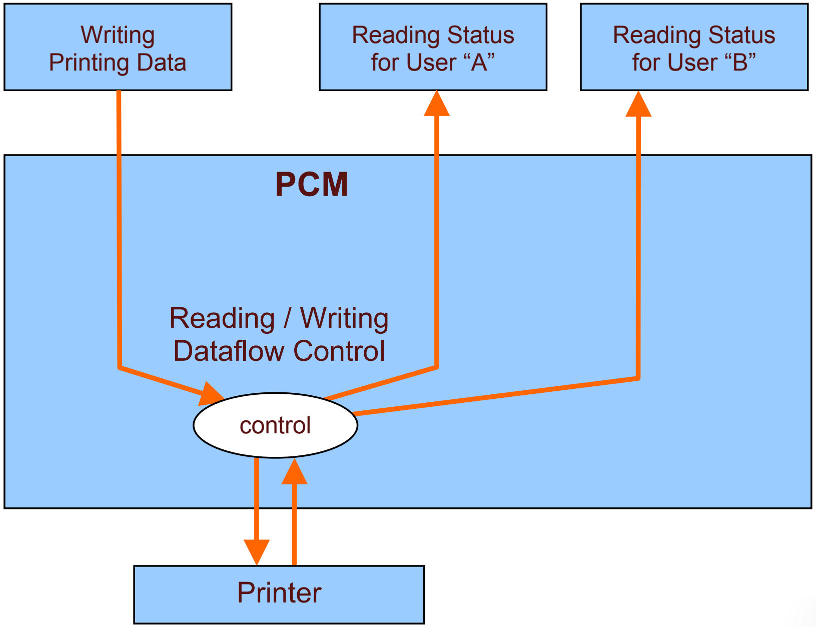
Agenda

- PCM Plug-in Interface
  - Backgrounds
  - Requirements
  - API Design
- Status Monitoring Interface
  - Just an idea
- Issues/Concerns, Next and Info.

PCM Plug-in Interface

Backgrounds

- PCM provides the multi-channel interface for;
  - Writing a printing data to the printer.
  - Reading a printer status data from the printer.



Backgrounds (Cont.)

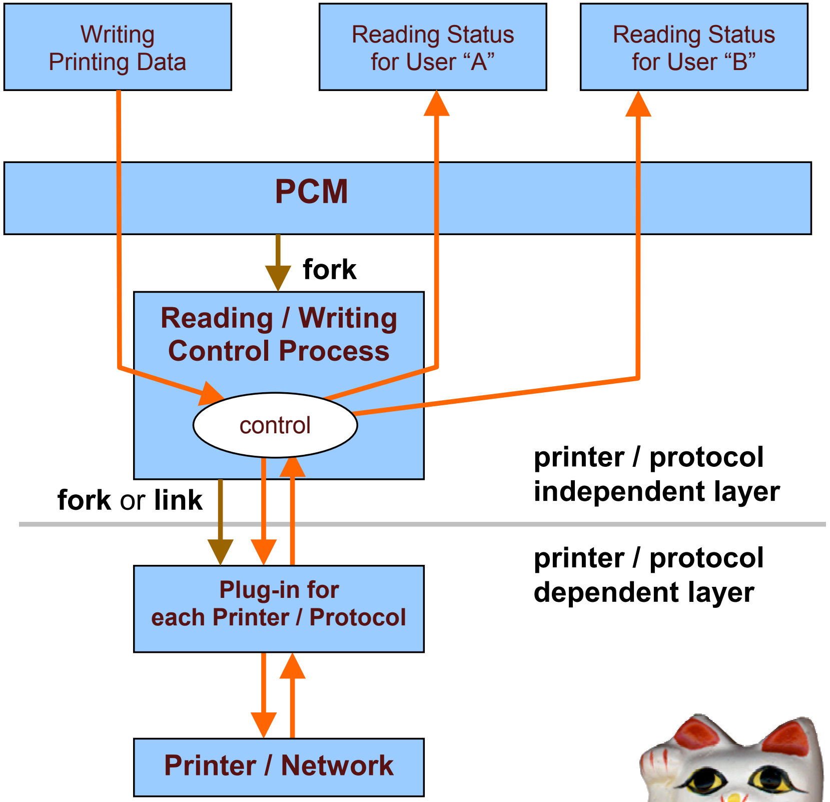
- Needs to control the multi-channel data flow.
  - Who controls?
- Each printer/protocol has its own commands and procedures.
  - Who deals with the printer/protocol dependencies?

↓

- PCM Should be separated into two layers;
  - Core module providing the multi-channel functionalities.
  - **Plug-in** providing each printer/protocol dependent functionalities.

Two Layers Arch.

- Reading / Writing Control Process
  - Forked by PCM for each printer.
  - Manages the dataflow due to;
  - Data writing request by PCM.
  - Data reading request by PCM.
  - Data writing ready from Plug-in.
  - Data reading ready from Plug-in.
- Plug-in for each Printer / Protocol
  - Manages the printer / protocol dependent;
  - Packet control.
  - Timing Control for reading / writing.
  - Data translation.


Writing Printing Data
Reading Status for User "A"
Reading Status for User "B"
PCM
fork
Reading / Writing Control Process
control
fork or link
printer / protocol independent layer
printer / protocol dependent layer
Plug-in for each Printer / Protocol
Printer / Network

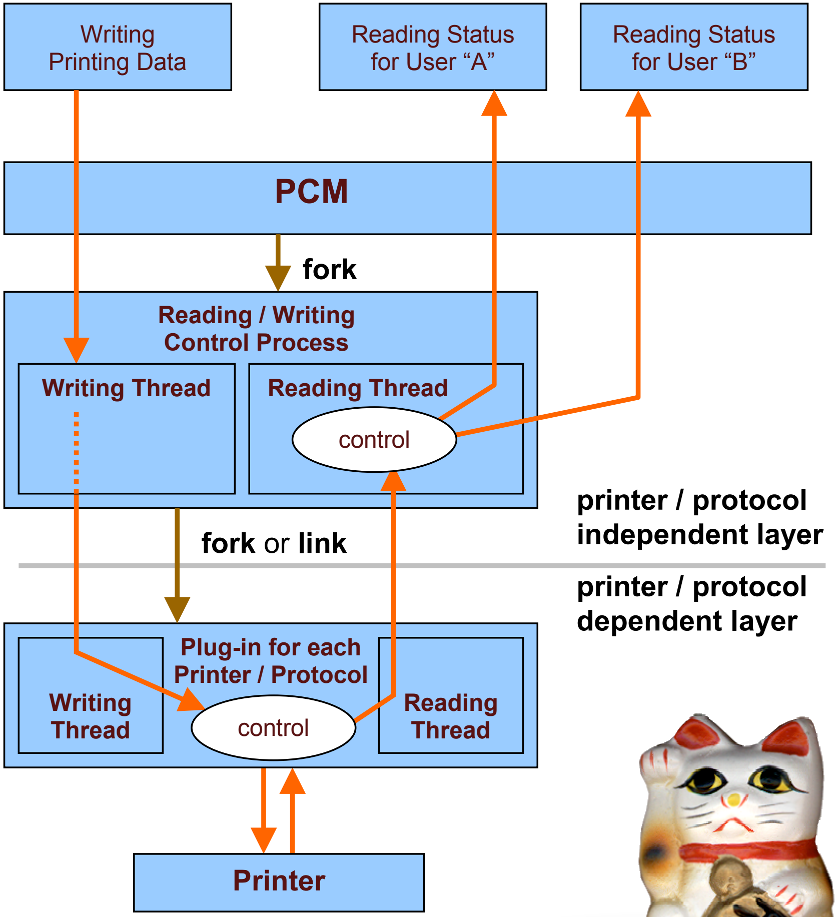
Two Layers Arch. by threads

- Reading / Writing Control Process

  Writing Thread
  - Manages the dataflow due to;
  - Data writing request by PCM.
  - Data writing ready from Plug-in.

  Reading Thread
  - Manages the dataflow due to;
  - Data reading request by PCM.
  - Status reading ready from Plug-in.

- Plug-in for each Printer / Protocol
  - Manages the printer / protocol dependent;
  - Packet control.
  - Timing Control for reading / writing.
  - Data translation.


Writing Printing Data
Reading Status for User "A"
Reading Status for User "B"
PCM
fork
Reading / Writing Control Process
Writing Thread
Reading Thread
control
fork or link
printer / protocol independent layer
printer / protocol dependent layer
Plug-in for each Printer / Protocol
Writing Thread
control
Reading Thread
Printer

Requirements for Plug-in

Basic Requirements

- Plug-in must;

  A-1) Provide the API for;

  - Writing a printing data to the printer.
  - Reading a printer status data from the printer.

  A-2) Conceal the printer/protocol dependencies.

  - PCM can deal with all the printers/protocols (e.g. IPP, etc..) as an unified object via each Plug-in.

  A-3) Provide the API that can be applied on both;

  - Big memory system.
  - Small memory system.

Port Control Requirements

- Plug-in must;

  B-1) Conceal the port control procedures based on each printer/network protocol dependencies.

  - Open / Close a USB port, etc...
  - Open / Close an IPP port etc...

- Port must;

  B-2) Be specified by the generic device/network protocol identifier.

  - URI ?

Data Reading Requirements

- Plug-in must;

  C-1) Provide a generic API for sending back the variable length printer status.

  - Printer status data length depends on each printer model and conditions of the printer.
  - Plug-in may separate a single very long printer status data into a set of several short data for sending them back into a short restricted buffer prepared by the upper layer.

  C-2) Provide the API to know the top of the printer status data and/or to reset the reading position of it.

  - The upper layer doesn’t know what the status data includes(binary, XML, etc…), but needs to know the top of it.
  - Reading a printer status data may be interrupted by a signal.

Data Reading Requirements (Cont.)

- Plug-in must;

  C-3) Keep the printer status data consistent until all the status data are read by the upper layer.

  - Plug-in needs a trigger to allow for reading a printer status from the printer/network port.

  C-4) Provide the API which gives a trigger to the Plug-in for dealing with reading a printer status data and writing a printing data exclusively.

  - Some printers/protocols have to deal with reading a printer status data and writing a printing data exclusively, and the Plug-in needs a trigger for doing it.

Data Writing Requirements

- Plug-in must;

  D-1) Provide the API which gives a trigger to the Plug-in for dealing with reading the printer status data and writing the printing data exclusively.

  - Same as C-4) of the Data Reading Requirements.
  - Writing a printing data may be interrupted by a signal.

Dataflow Control Requirements

- Plug-in must;

E-1) Let the upper layer know;

- If the Plug-in is ready to receive a printing data.
- If the Plug-in is ready to send back a printer status data.

- Plug-in should;

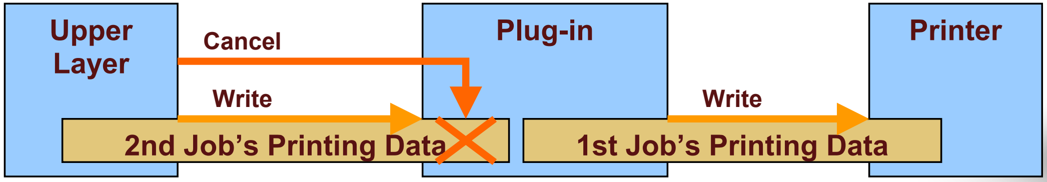
E-2) Provide the API being not affected by OS platform dependencies.

Printer Control Requirements

- Plug-in must;

F-1) Provide the API to cancel a printing.

- The upper layer keeps alive the Plug-in for obtaining the printer status continuously across several printing jobs.
- Each printing job's printing data might be specified by an identifier for the cancellation. (TBD)
  - Example) In case of the upper layer sends two job's printing data to the Plug-in. After the upper layer finished to send the first job's printing data to the Plug-in, the upper layer can start to send the second job's printing data while the Plug-in may send the first job's printing data to the port. During this period, the first job has not been finished, and if the user wants to cancel the second job after noticed the mistake, the upper layer needs an identifier to cancel the second job's data buffering, or cancel the first job.



Printer Control Requirements (Cont.)

- Plug-in might (TBD);

F-2) Provide the API for the common control functions.

- “Pause”, “Resume” for local port printers as well as IPP ?
  - IPP needs a job identifier to control a job.

F-3) Provide the API for tracking printing jobs.

- Plug-in for IPP will be provided, while IPP provides job attributes.
- If PCM is the “only” interface to access a printer, PCM might provide the interface for tracking printing jobs in a printer to satisfy some applications needs.

Plug-in API Design

Object / Port Control

Object Control

New

- Conceal the printer/protocol dependencies in the created object.
- Object keeps the URI specified by the upper layer until destroyed.

Destroy

- Destroy the object.

Port Control

Open

- Open the printer / network port specified by the object.

Close

- Close the port / network port specified by the object.

Data Reading

StartRead

- Declare the start of reading a printer status.
- Reset the reading position of the printer status data in the object.

Read

- Read the specified byte length of a printer status data from the object, and store them in the buffer given by the upper layer.
- Returns the status data byte length read from the object, or zero when no status data remains in the object.

* The data format of the printer status is out of scope of this API.

EndRead

- Declare the end of reading a printer status.

Data Writing

StartWrite

- Declare the start of writing a printing data.

Write

- Write the specified byte length of a printing data in the buffer given by the upper layer to the object.
- Returns the printing data byte length written to the object, or zero when no printing data written to the object.

EndWrite

- Declare the end of writing a printing data.

Dataflow Control

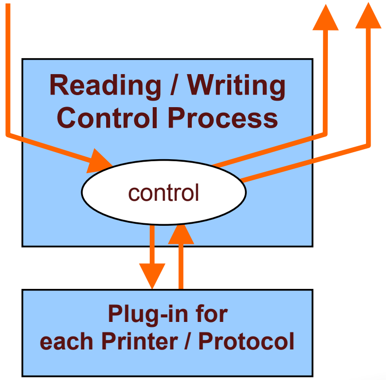
GetReadFD

- Obtain the reading file descriptor for select() use.
- The upper layer know by select() if the Plug-in is ready to send back a printer status data.

GetWriteFD

- Obtain the writing file descriptor for select() use.
- The upper layer know by select() if the Plug-in is ready to receive a printing data.

The upper layer deals with all the reading/writing request by select().


Reading / Writing Control Process
control
Plug-in for each Printer / Protocol

Printer Control

StartJobData

- Declare the beginning of the printing data of each job to the object.

CancelJobData

- Cancel the printing data specified by the job identifier.

EndJobData

- Declare the end of the printing data to the object.

ExtCtrl

- Execute the extended printer control,
  such as “Pause”, “Resume”...

Status Monitoring Interface

Just an Idea

- Requirements
  - Status Monitoring Interface must;
    Translate the printer status data read from each Plug-in into the generic format data that does not depend on each printer/protocol.
  - Generic format is XML? IPP? other?

Just an Idea (Cont.)

- API Design
  - New
    - Conceal the printer/protocol dependencies in the created object.
    - Object keeps the “language” and “character encoding” specified by the caller (module which calls this API) until destroyed.
  - StartTranslate, Translate, EndTranslate
    - Translate the printer status data given by the caller to the generic format data, and store it into the buffer given by the caller.
  - Destroy
    - Destroy the object.

Issues/Concerns, Next and Info.

Issues/Concerns, Next

- Plug-in Interface
  - Need to collect the requirements for handling events.
- Status Monitoring Interface
  - Need to collect the requirements for the Status Monitoring functionalities.
- Next
  - Define the draft Plug-in API.
  - Define the draft Status Monitoring API.

Info.

Contributors

TORATANI Yasumasa	Canon Inc.
Osamu MIHARA	FUJI XEROX Printing Systems Co. Ltd.
Ide Kentaro	SEIKO EPSON CORPORATION
Nomura Kazuo	EPSON SOFTWARE DEVELOPMENT LABORATORY, INC.
KANJO Hidenori	BBR INC.
YOSHIDA Mikio	BBR INC.
Shinpei KITAYAMA	EPSON KOWA CORPORATION
YAMAGISHI Toshihiro	Turbolinux, Inc.
Hisao NAKAMURA	E&D
Koji OTANI	AXE

Thank you.