
# Free Standards Group OpenPrinting Reference Model

Version 0.01
April 22, 2005


## Abstract

The Free Standards Group (FSG) OpenPrinting Reference Model defines the overall system diagram for Linux open printing solution. The Reference Model provides a functional decomposition into and description of subsystems. It identifies the data and/or control and/or interfaces between subsystems and/or external systems.

# Copyright Notice

# Table of Contents

# 1. Introduction

This purpose of this document is to define the Reference Model for Linux open printing.

# 2. Background

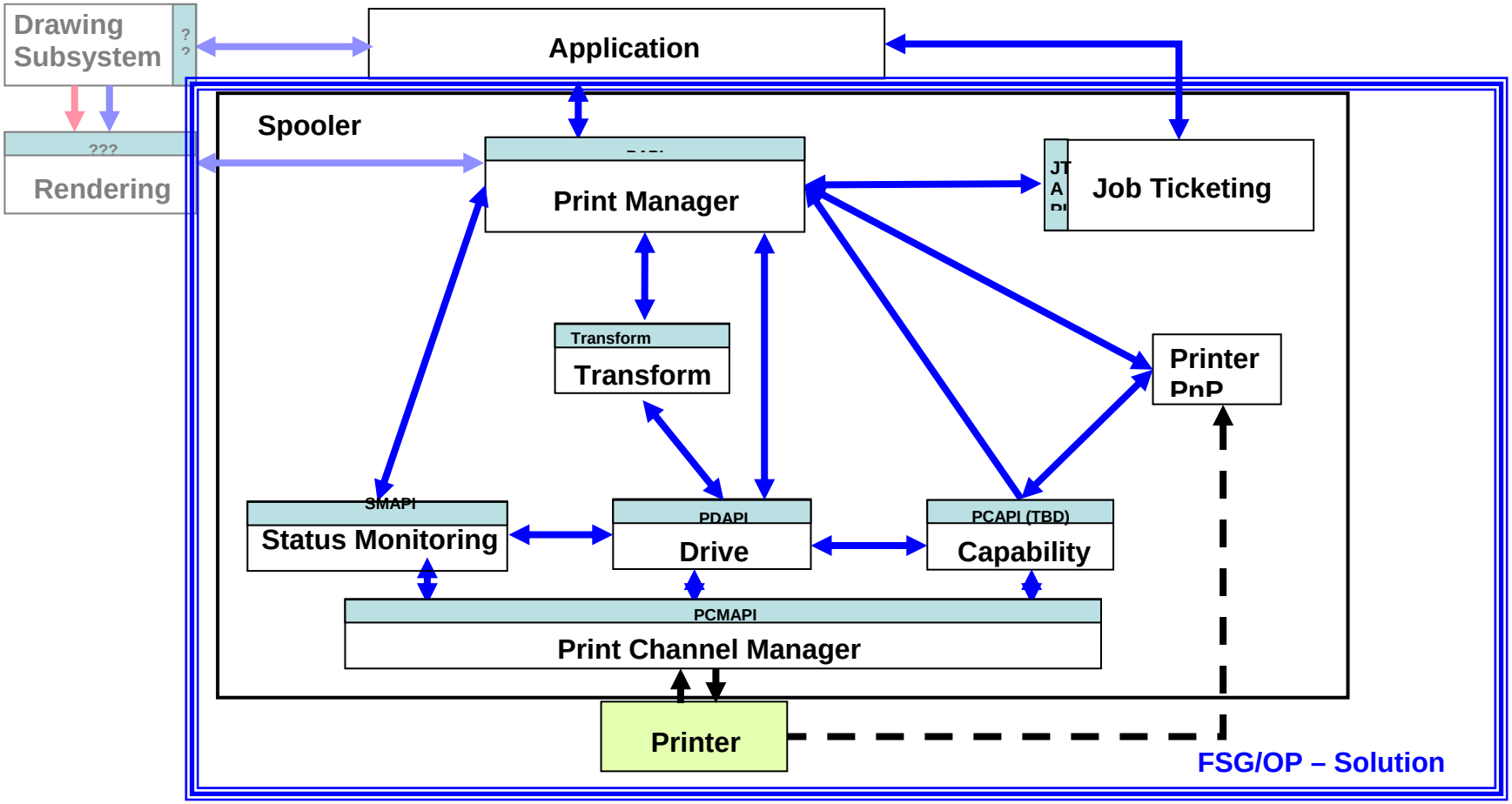
The elements of an architecture contain three major component; specifically,

- Reference Model (RM)
  - Overall system diagram
  - Functional decomposition into and description of subsystems.
  - Identification of data and/or control and/or interface between subsystems and/or external systems.
- High-Level Architecture (HLA)
  - A glossary of terminology.
  - Analysis and documented Use-Cases and requirements.
  - Identification of applicable/recommended standards for subsystem interfaces.
  - Identification of recommended infrastructure for integrating products.
- Detailed Architecture (DA)
  - Specification of system structure (classes, packages, associations – using UML).
  - Specification of system behavior (activity and sequence diagrams – using UML).
  - Identification of the system process structure.
  - Identification of inter-process communication mechanisms.

## 2.1 A Reference Model

A **Reference Model (RM)** provides definitions and a formal structure for describing the implicit and explicit concepts and relationships used in Linux open printing.

## 3. Simplified Reference Model Diagram



# 4. Reference Model Description of Subsystems

## 4.1 Spooler

The Spooler subsystem provides the generic function for temporary storage of data for internal use by individual printing subsystems and/or for transfer (by reference) between printing subsystems. In general, the Spooler will not provide external, outside of the printing subsystems, access. Memory management and memory transfer shall be controlled by the Spooler. Memory management shall include, but not be limited to, allocation and de-allocation of Spooler buffers. Memory transfer shall be determined by the type of physical and logical storage used by the Spooler.

From the above definition, the Spooler is accessible by all printing subsystems for bi-directional data interfacing and uni-directional commands. To support this scope of interaction, the Spooler shall, at a minimum, support the following functionality.

- Create a spool buffer
  - Provide automatic release of buffer after a condition or time limit.
- Assign access (open/read/write/close/free) rights for other printing subsystems
- Store or retrieve a single byte or a buffer of data
- Provide simultaneous store and retrieve from different requestors.
- Maintain status on individual spool buffers

---

Maybe this should an example?????????

The Spooler subsystem provides a single subsystem for saving (writing) and retrieving (reading) print jobs on behalf of the Print Manager subsystem. The Spooler provides buffering of incoming print jobs (including job tickets and job data) for later scheduling and forwarding to other FSG/OP Architecture subsystems. There is a single Application Programming Interface (API) for Spooler access.

For each incoming print job, the Print Manager subsystem may open an output stream to the Spooler subystem (using a URI to specify the destination file for portability), copy the incoming job to the Spooler subsystem, and then close the output stream. Later, the Print Manager subsystem will schedule the saved job, open an input stream from the Spooler subsystem for the saved job (using a URI), forward the saved job to another FSG/OP Architecture subsystem for processing, and then close the input stream.

See: Section 1.1 'Simplified Printing Model' and section 2.2 'Job Object' in IPP/1.1 Model and Semantics, RFC 2911, September 2000.

See: Document Printing Application (DPA), ISO 10175, June 1996.

## 4.2 Print Manager

The Print Manager subsystem provides the functionality of accepting, managing and processing print jobs from Software Applications and/or Gateways. Additionally, the Print Manager accepts and manages print content, controls and manages printing within the print system, and monitors and manages print devices. The Print Manager is the single (only) entry Application Programming Interface (API) for the print system.

The Print Manager after accepting a job ticket shall transform the job ticket into the print systems native job description or job ticket format (this can be done by the Job Ticket API subsystem). Upon successful transformation of the job ticket content, the Print Manager shall identify and locate all of the resources (transforms, drivers, I/O controls, etc) necessary to complete the print job. Upon successful resource identification, the Print Manager will schedule the print job and report to the caller the print job was received and scheduled else the Print Manager will report on why the print job failed. Next the Print Manager sequentially executes individual printing processes while maintaining status on each process.

The Print Manager subsystem shall use the Spooler subsystem to support rate adaptation across different communications paths and/or additional temporary job storage for limited Printer subsystems.

### 1.1.1 Print Manager Subsystem Interfaces

The Print Manager Subsystem interfaces with all other subsystem in the print system. Therefore, subsystem interface discussion will be given in the section for each subsystem.

---

The Print Manager subsystem provides a single subsystem for accepting, monitoring, scheduling, and canceling print jobs for Applications and Gateways. The Print Manager provides an abstraction of Servers, Queues, Jobs, Documents, and Resources (see ISO 10175). There is a single Application Programming Interface (API) for Print Manager access.

An Application or Gateway may open an output stream to the Print Manager subsystem, send the job/document processing and description attributes,

(optionally) the job ticket, and the job data (by reference or by value) to the Print Manager subsystem, and then close the output stream.

The Print Manager subsystem may use the Spooler subsystem to support rate adaptation across different communications paths and/or additional temporary job storage for limited Printer subsystems.

See: Section 1.1 'Simplified Printing Model' and section 2.1 'Printer Object' in IPP/1.1 Model and Semantics, RFC 2911, September 2000.

See: Document Printing Application (DPA), ISO 10175, June 1996.

## 4.3 Job Ticketing

The Job Ticket Application Programming Interface (JTAPI) shall provide an abstract interface for applications to create, read, edit, and write print job tickets. The syntax of a specific print job ticket (industry standard or proprietary format) shall not addressed by the JTAPI. The JTAPI shall define an abstract model of objects (print jobs, documents, etc.) and their operations and attributes for printing.

The JTAPI shall isolate an application and print system from the syntax of a job ticket to hide details and the structural complexity of a specific job ticket along with interoperability between different job ticket file formats. The JTAPI abstract model shall be programming language, operating system and print job ticket neutral. The JTAPI shall be an object oriented extensible API that contains objects that are well known in the print industry.

---

Put this in next level architecture document

A print job ticket information object (JobTicketInfo) contains one job object (Job) where the job contains zero or more document objects (Document). The JobDocumentPage object is abstract and contains functionality that is common to jobs, documents, and specific pages (PageOverrides) in a job or document. Each of the other objects in the JTAPI represents functionality that can be specified for the Job, Document, and PageOverrides objects. For example, the Media object represents the media that the job, document, or specific pages in the job or document is to be imaged/printed on.

Some examples of industry standard job tickets are as follows:

The International Cooperation for the Integration of Process in Prepress, Press and Postpress (CIP4) is a joint initiative of vendors for the graphical arts industry. CIP4 has published a Job Definition Format (JDF) specification. JDF is a comprehensive XML-based file format proposed industry standard for end-to-end job ticket specifications combined with a message description standard and message interchange protocol to cover all aspects of the commercial printing workflows.

The Printer Working Group (PWG) is a joint initiative of printer vendors and print system providers to develop printing protocol standards for use on the Internet and within enterprises on their intra nets [pwg]. The PWG has published the Internet Printing Protocol (IPP) in September 2000 [rfc2910, rfc2911]. The PWG is in the process of publishing the PWG Semantic Model which summarizes the printing semantics common to a number of printing protocols, centered on the IPP semantics. The PWG Semantic Model includes an XML Schema definition. Therefore, an XML Job Ticket using the semantics of the PWG Semantic Model is possible.

### 4.4 Print Dialog

### 4.5 Application Interface

### 4.6 Transform

## 4.7 Driver

A printer driver (driver) is the interface between the operating system (OS) and a physical printer (printer). There are many responsibilities for a driver. The primary responsibility is to generate print jobs from an application.

One interface that the driver provides deals with properties. There is a set of properties that can change between print jobs. These properties are called job properties. The job properties are divided into three sets, 1) those properties that are defined by the OS and are supported by the printer driver, 2) those properties that are defined by the OS and understood by the driver but not necessarily supported by a driver and 3) properties that are specific to each driver. Some properties might be supported if an optional feature of the printer is installed. Defining if these properties are installed/activated or not are called printer properties. Through the interface, applications (and spoolers) may enumerate the properties (either one or many). The interface also may tell the application that constraints exist between different properties.

Another interface is dealing with print jobs. Print jobs under Linux are generated using the Postscript language. When the jobs are despooled, then a rendering engine that understands Postscript will convert the Postscript commands into a API set for the driver. The driver will then create a printer specific data stream that corresponds to the Postscript commands. Drivers may support either a set of vector drawing calls or accept a raster bitmap or support both sets of drawing calls.

Drivers may optionally support device fonts. When device fonts are supported, then applications can query and use these fonts as part of the print job. The font metrics are provided so that the application can position the characters on the screen using a system font if an exact font is not available.

Drivers are asked what devices that they support. Drivers are told to install or remove a specific device. They are also told that a new version is being installed.

Drivers are responsible for bidirectional communications protocols to their device.

Drivers may provide a color profile for their device given the current job properties.

## 4.8 Capability

The Capabilities Subsystem provides a single subsystem for the collection, acquisition, summarization, dissemination and/or reporting of the printer's physical and logical capabilities. Collection of capability information comes from the printer, a file(s) and/or network location(s). There is a single Application Programming Interface (API) that can accept capability information in to-be-determined formats and report specified capability information. The API will provide specified information for a specified print device and/or can provide the printer or list of printers with the specified capability(ies). "Standard" capability information (to be determined) is formatted for efficiency while formatting for extension capability information is represented by key-value pairs. The Capability Subsystem does not interpret capability information nor validate the information beyond syntax and "reasonableness" bounds checking for "standard" capability information.

## 4.9 Print Channel Manager

The Print Channel Manager subsystem provides a single subsystem for opening, managing, transferring data over and closing Channels (using datagrams or connections, local or network) between any of the FSG/OP Architecture subsystems and a Printer subsystem. There is a single Application Programming Interface (API) for Channel access.

An FSG/OP Architecture subsystem may open a Channel to a Printer (using a URI to specify the destination protocol endpoint for portability) in order to support a job submission, job management, status monitoring, or other application protocol.

Note: A single transport connection may simultaneously support more than one Channel (e.g., IETF BEEP, RFC 3080).

A Channel includes a transport protocol (e.g., TCP or UDP), and may include a session protocol (e.g., HTTP), a presentation protocol (e.g., SOAP), and/or layer security protocols (e.g., TLS). A Channel is bound (for the lifetime of the Channel) to a particular Interface.

An Interface includes a communications port (local or network) or an OS pipe/shared-memory port, and may include a datalink protocol (e.g., Ethernet), a network protocol (e.g., IP or IPX), and/or layer security protocols (e.g., IPsec).

See: Section 2.2.10 'Print Job Delivery Channels' and Channel group in
section 6 of Printer MIB v2, RFC 3805, June 2004.

See: Blocks Extensible Exchange Protocol Core, RFC 3080, March 2001.

## 4.10 Status Monitoring

The Status Monitoring subsystem provides a single subsystem for the collection, acquisition, summarization, dissemination and/or reporting of the physical and/or logical print driver, print communication and/or printer status and/or state information. There is a single Application Programming Interface (API) for the setup and retrieval of specified status information. In addition to polling support for specified status information, the API shall include a callback mechanism to report specified status information based on an events and/or times. The subsystem shall include a dead-man timer capability for monitoring the general operation of the print driver, print communication and/or printer. Status information, for example includes, but is not limited to, the printer's current state, ink/toner levels, percentage of a page, document and/or job completed and the state/condition of all I/O channels. "Standard" status information (to be determined) is formatted for efficiency while formatting for extension status information is not specified. Status Monitoring is not to be confused with Job Monitoring associated with the Print Manager.

## 4.11 Drawing Subsystem

## 4.12 Rendering

A renderer is a program that takes a recording of drawing commands from an application and interfaces with a printer driver (driver) to create a print job in printer specific language. The driver may support high level drawing commands, accept a raster image, or handle a mixture of some high level drawing commands with a raster image fallback. The renderer is also responsible for presenting the printer driver's device fonts to the operating system. If an application uses device fonts, then the renderer will notify the printer driver during the print job.

The operating system may support the concept of device independent drawing contexts. Once these contexts are created, they can be used for drawing on the video screen or on hardcopy output with no change to the API set except for the initial creation.

### 4.13 Printer

Print Data

### 4.14 Printer Plug-aNd-Play (PNP)

## Appendix A: Organizational Structure

The diagram below provides an understanding of OpenPrinting organization to understanding of the relationship be the various OpenPrinting working groups.

# Editor

Glen Petrie [glen.petrie@eitc.epson.com ] - EPSON

# Authors

Claudia Alimpich [alimpich@us.ibm.com ] - International Business Machines
Mark Hamzy [hamzy@us.ibm.com ] - International Business Machines
Norm Jacobs [Norm.Jacobs@Sun.com ] - SUN Micro Systems
Till Kamppeter [till.kamppeter@gmx.net ] – MandrakeSoft
Ira McDonald [imcdonald@sharplabs.com ] - High North
Glen Petrie [glen.petrie@eitc.epson.com ] - EPSON

# Contributors

## Change Log

| *Date* | *Affected Version* | *Author* | *Change* |
|---|---|---|---|
| 2004.09.01 | ----- | Glen Petrie | Original Document |

Ж Ж Ж Ж Ж Ж Ж Ж Ж Ж **END OF DOCUMENT** Ж Ж Ж Ж Ж Ж Ж Ж Ж Ж