Printer Working Group
White Paper

Roger deBry
Harry Lewis

Print Server-to-Device Protocol Proposal

Abstract

This paper addresses the issue before the Printer Working Group of defining an open standard for Print-Server-to-Device interchange. It summarizes the current environment and the requirements for a common Server-to-Device protocol, and then proposes a solution.

Table of Contents

The Current Environment

There is currently not a widely accepted, well defined, standard protocol for communications between a print-server and a device. Although such a standard [1] has been developed, it has not been widely adopted. As a result, each vendor still builds some control into the underlying page description language or data stream[2]. Many vendors, in an effort to support the largest possible customer set, implement multiple such interfaces, adding substantial cost and complexity to their products. This situation also requires that operating systems which support many different printing devices must develop software which allows for this multiplicity of interfaces. Frequently the burden falls back onto the printer vendor to write unique print submission and management components [3] for each operating system.

[1] IEEE P1284.1, Draft Standard for Information Technology for Transport Independent Printer System Interface (TIP/SI), February 1997

[2] Examples include PJL, Postscript device controls, and IPDS

[3] In Windows, these are called “Port Monitors”

Requirements

This proposal specifically addresses issues and definitions related to a standard printing protocol for a standalone IPP server connected via a communications channel to a rendering device. We will refer to this as the “SDP environment” or simply “SDP (Server-Device Protocol)”.

The Printer working group has defined a number of standards which address the interoperability of printing components. These include the SNMP Printer MIB, the SNMP Job Monitoring MIB, and the Internet Printing Protocol. In addition, the PWG has ongoing efforts to describe a common method for registration, filtering and transport of printer and print job related notifications. While, together, these standards would provide much of what is required in a Server-to-Device solution, we recognize that not every client will adopt SNMP, and IPP as currently defined may not be suitable for all embedded devices. Still, we feel that a key requirement must be to closely integrate the SDP environment with these, most recent, efforts of the PWG. Specifically, we feel it is imperative to preserve the IPPv1 encoding, the Printer MIB objects and OID references, and the Job MIB attributes.

There is a strong signal in the PWG that, for broad acceptance in the client market, all necessary SDP functions must be encapsulated into a single, transport independent protocol. Given that IPP already addresses job submission and query of some device and job characteristics, the SDP protocol will further require an alternate channel for control and unsolicited, real-time status. Also, SDP must easily support new IPP extensions as they occur.

Alternatives

Alternative I: Do nothing

An obvious alternative is to do nothing, that is, accept the status quo and continue to do business as usual. This will result in each device manufacturer continuing to enhance and develop whatever protocols they believe are required to give them a competitive edge in the marketplace. While this laissez-faire attitude requires the least amount of standards work, it does nothing to solve the problems of interoperability and the cost and complexity of supporting multiple interfaces.

Alternative 2: Adopt an existing protocol as the standard

Another alternative is to adopt an existing protocol as the standard. Since TIP/SI was created as a standard to fill this space, and meets many of the requirements, it would seem a likely choice. However, TIP/SI was completed prior to the development of IPP, the Job Monitoring MIB, and the completion of the Printer MIB, so even were we to adopt TIP/SI, it would be necessary to do the work required to rationalize these four standards. A variation on this proposal defines an IPP LU in IEEE1284.1 [4]

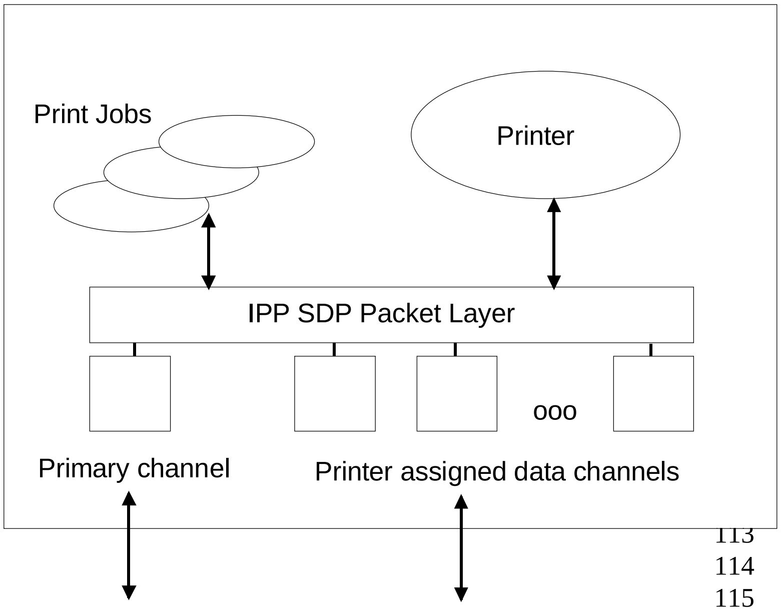
Alternative 3: Define an IPP server-to-device protocol

This approach recognizes IPP as the most contemporary model for print submission, and uses it as the basis for adding the additional functions needed to meet the requirements of a robust Server-to-Device protocol. In particular, this means adding a notification scheme to IPP to handle unsolicited real-time status information, defining alternate channels for control information and the flow of print data, and doing this on native TCP/IP, IEEE1284, and other device protocols. This paper proposes a definition for such an IPP based protocol.

[4] DonWright, “Internet Printing Protocol/1.0: IPP to 1284.1 Mapping”

The IPP Server-to-Device Proposal

This proposal takes IPP as currently defined and adds a packet structure to facilitate the transport independent flow of IPP operations and attributes between the IPP server and the device. The IPP SDP model and packet structure is derived from the IEEE1284.1 (TIP/SI) standard. The packet structure is a key piece of technology, providing both a data and control channel, acknowledgments, and asynchronous notifications as mandated by the requirements. The IPP SDP model differs from IEEE 1284.1 in that communication is differentiated between Printer object and Print Job object, to better fit the IPP paradigm. Changes are recommended to the Mandatory/Optional set of IPP operations, streamlining them for the SDP environment, but the IPPv1 encoding is completely preserved. Because this proposal specifically addresses the SDP environment, by choice and definition, this is a PUSH printing model only.



Packet Header Structure

The IPP SDP packet header is defined as follows:

Packet Header				
Start of Packet Byte	Packet Length	Flag	Reserved	Message

Start of Packet Byte

This byte is used to indicate that this is the start of a packet. When the device is in a state to receive a packet from the server, the first byte of the packet MUST be x’25’, (CR). If this byte is not the first byte, the device knows that it is not in sync with the server.

Packet Length Bytes

This field defines the number of bytes in this packet not including the packet length and the start of packet byte.

Command Flag Byte

The flag byte provides bit encoded flags which can be inspected to obtain control information. Flag bits are defined as:

Bit 7: Reserved
Bit 6: Data Channel Request. Set if a separate data channel is requested.
Bit 5: Continuation bit. Set if the next packet is a continuation of this message.
Bit 4: Reply Required. When set this bit tells the device that the server requires an acknowledgment.
Bits 3-0: Reserved

IPP provides for communication with two distinct classes of objects, printers and jobs. The destination (the printer or the job) is implied by the IPP operation, so a separate flag and command byte are not required.

Reserved

This field is reserved for future extensions.

Response Flag Byte

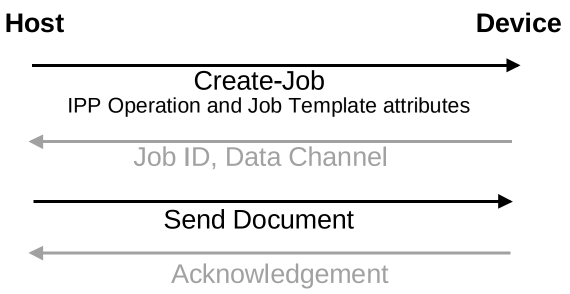
The flag bits in a response are different from those in a command, and are used by the server to correctly interpret the response.

Bit 7: Reserved
Bit 6: Data Channel. Set if a data channel is defined in this response message.
Bit 5: Continue bit. Set if the next packet is a continuation of this message.
Bit 4-3: Content Type
    00 - Standard IPP Response
    01 - Reserved
    10 - Unsolicited data - alert from Printer MIB table
    11 - Unsolicited data - alert from Job Monitor MIB table

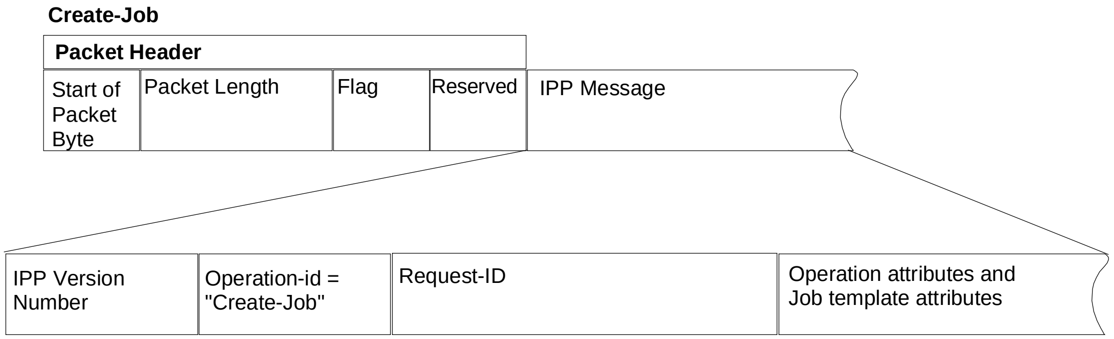
Message

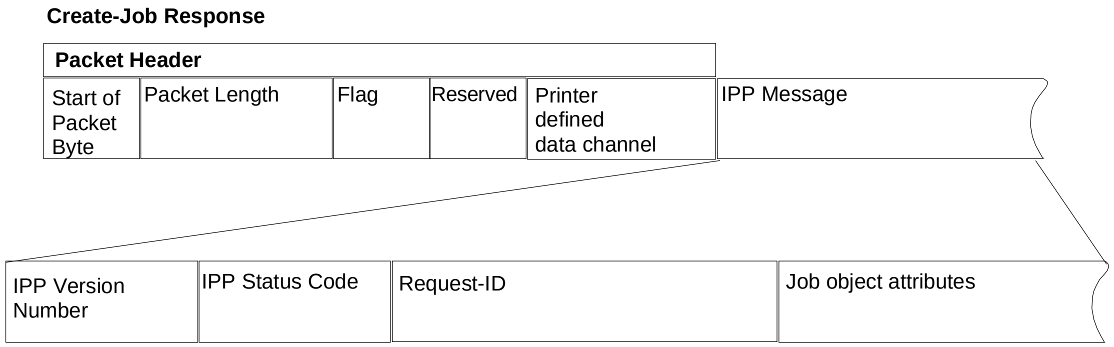
The content of the message field is always an standard IPP message, exactly as defined in section 3.0 of the IPP protocol Specification[5], in a command packet. Response messages may be standard IPP messages, or unsolicited data in the form of alerts. Using this scheme, a job submission flow would look like:

[5] Herriot, et al., <draft-ietf-ipp-protocol-05.txt>



Not surprisingly, this flow looks exactly like IPP on a client to server session.. The Create-Job request and the corresponding Create-Job Response would appear something like the following, within the packet structure that has been defined:





In order to preserve the distinction between device and job flows, Create-job and Send-document would be mandatory operations in this case, while Print-job would be an invalid operation. IPP notification registration would be required to set up the flow of alerts..

When a separate data channel is requested in the flag byte of a command, it is returned as the first two bytes following the packet header, and its presence is indicated by a flag bit.

SDP Notifications

The PWG is currently considering the topic of notifications as embodied in the documents "draft-ietf-ipp-not-01.txt" March 11, 1998 (deBry), "Notifications for the IPP print protocol" April 2 1998 (Hastings and Lewis), and with the SENSE PWG project for generalized event subscription. Registration methods have been described which, when adopted by IPP, will become part of SDP by virtue of their IPP operations and encoding.

IPP Sub-unit Objects

To address the needs of using one SDP protocol for printer management, the PWG has been presented with "IPP Sub-Unit Objects", April 7, 1998 (Isaacson, Hastings), which adds a new "Get-Sub-Units" printer operation to IPP. The intent of this new operation is to allow IPP (and thus SDP) to obtain values for all Printer MIB objects without requiring a separate protocol such as SNMP or HTTP. There are still choices to be made regarding extension of the list of IPP printer attributes vs. use of string representations of OIDs as attribute names. SDP will adopt the resolution of this effort. Key to any print management scenario, however, is the distinction between a list of sub-units (input tray 1,2,3 etc.) and or devices (printer1,2,3 etc). This proposal currently assumes the work in progress will address and resolve this issue.

Get GPD Operation

While the topic of a Universal Print Driver is not being actively engaged in the PWG at this time, it is an open topic which we feel needs to be accommodated by the SDP. Should a universal device description (such as the Microsoft GPD) ever be adopted, we propose that a new IPP printer operation called "Get-GPD" be defined which would invoke the device to respond, via the standard SDP protocol, with it’s GPD description.

IPP Server-to-Device conformance set (for device)

Because of the nature of Server-to-Device communications, the following are proposed as the IPP conformance definition for a device.

Printer Operations:
Print-Job (not supported)
Print-URI (not supported)
Validate-Job (not supported)
Create-Job (mandatory)
Get-Printer-Attributes (mandatory)
Get-Jobs (mandatory)

Job Operations:
Send-Document (mandatory)
Send-URI (not supported)
Cancel-Job (mandatory)
Get-Job-Attributes (mandatory)

Operation Attributes:
pinter-uri (not supported)
attributes-charset (optional - UTF8 unless indicated)
attributes-natural-language (optional - us english unless indicated)
requesting-user-name (mandatory)
job-name (mandatory)
ipp-attribute-fidelity (mandatory)
document-name (mandatory)
document-format (mandatory)

document-natural-language (optional)
compression (optional)
job-k-octets (optional)
job-impressions (optional)
job-media-sheets (optional)

Job Template Attributes
job-priority (optional)
job-hold-until (not supported, a server function)
job-sheets (not supported, a server function)
multiple-document-handling (optional)
copies (optional)
finishings (optional)
page-ranges (optional)
sides (optional)
number-up (optional)
orientation (optioanl)
media (optioanl)
printer-resolution (optional)
print-quality (optional)

Job Description Attributes
job-uri (not supported)
job-id (mandatory)
job-printer-uri (not supported)
job-more-info (not supported)
job-name (optional)
Job Description Attributes (continued)
job-originating-user-name (optional)
job-state (mandatory)
job-state-reasons (optional)
job-state-message (not supported)
number-of-documents (optional)
output-device-assigned (not supported)
time-at-creation (optional)
time-at-processing (optional)
time-at-completed (optional)
number-of-intervening-jobs (optional)
job-message-from-operator (optional)
job-k-octets (optional)
job-impressions (optional)
job-media-sheets (optional)
job-k-octets-processed (optional)
job-impressions-completed (optional)
job-media-sheets-completed (optional)
attributes-charset (optional)
attributes-natural-language (optional)

Printer Description Attributes
printer-uri-supported (not supported)
uri-security-supported (no supported)
printer-name (optional)
printer location (not supported)
printer-info (not supported)

printer-more-info (not supported)
printer-driver-installer (not supported)
printer-make-and-model (optional)
printer-more-info-manufacturer (not supported)
printer-state (mandatory)
printer-state-reasons (optional)
printer-state-message (not supported)
operations supported (not supported)
charset-configured (mandatory)
charset-supported (optional)
natural-language-configured (mandatory)
generated-natural-language-supported (optional)
document-format-default (mandatory)
document-default-supported (optional)
printer-is-accepting-jobs (mandatory)
queued-job-count (optional)
printer-message-from-operator (optional)
color-supported (optional)
reference-uri-schemes-supported (not supported)
pdl-override-supported (mandatory)
printer-up-time (optional)
printer-current-time (optional)
multiple-operation-timeout (optional)
compression-supported (optional)
job-k-octets-supported (optional)
job-impressions-supported (optional)
job-media-sheets-supported (optional)

Appendix: Detailed Requirements Statement

The following requirements come from discussions held at past PWG meetings.

1. The SDP protocol must provide a means to synchronize communications between the Server and Device, regardless of the underlying transport (i.e. start of packet indicator).

2. The SDP protocol must provide for packatized data flow with the ability to segment data for efficient use of underlying services with a means to indicate final segment. ("Chunking")

3. Must provide the ability for either Server or Device to mandate synchronization (ACK) to their message flow, if appropriate.

- Server -- Any message flow as Server sees appropriate based on data integrity needs, perceived communications reliability etc.
- - Device -- Limited message flow, mainly for the purpose of determining when a server may have lost connection. (**I’m not sure about this one**), or during the reverse transmission of FAX or SCAN data

4. The SDP protocol must asynchronous notifications including support for registration, de-registration and event type filtering.

5. The SDP protocol must support distinction between communications with Print Job Objects and Printer Objects, as defined in the IETF IPP Model standard.

6. The SDP protocol must accommodate the IPP encoding as defined in the IETF IPP Protocol standard.

(The following are from Portland and may be a bit redundant or need restating, rewording, or we may choose not to include some of them in our document)

7. The SDP protocol must be completely Transport independent.

8. Need way to send FAX or SCAN data from device to server (for MFPs only)

9. Control channel can't be blocked by data. Server can query and control with quick response.

10. Need configuration and status info like what's provided in the printer MIB.

11. Ability to recover job accounting information

12. Device can retrieve resources like fonts and forms

13. Server-to-Device protocol must not significantly limit the function of any major existing client to server protocols and must accommodate IPP without any loss.

14. Must Cover the case of spooling both in the server and the device or other multiple levels. The user should get the same functionality (CANCEL, MODIFY etc.).

15. Submit, Cancel, and list jobs (end-user and administrator)

16. Provide a means of client contention resolution (lunch counter ticket vs underlying protocol).

17. Allow printer to throttle data from the server.