HP | DesignJet Printers | PJL Passthrough and SNMP

PJL Passthrough to PML and SNMP
User's Guide

First edition, September 1997
Second edition, March 1998
Third edition, September 1998
Fourth edition, January 1999
Fifth edition, May 2000

This document explains how to use the HP DesignJet 1000, 2000 and 3000 Series printers PJL Passthrough to PML protocol (called simply “PJL Passthrough” in this document) and these and the HP DesignJet 500, 800 and 5000 Series printers with SNMP, so that ISVs can get the required bidirectional information that allows them to remotely manage the printer.

Notice

The information contained in this document is subject to change without notice and should not be construed as a commitment by the Hewlett-Packard Company.

Hewlett-Packard assumes no responsibility for any errors that may appear in this document nor does it make expressed or implied warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The Hewlett-Packard Company shall not be liable for incidental or consequential damages in connection with, or arising out of the furnishing, performance, or use of this document and the program material which it describes.

Hewlett-Packard Company
InkJet Commercial Division
Avda. Graells, 501
08190 Sant Cugat del Vallès
Barcelona, Spain

Contents

1. Introduction

This document describes

- The PML (Peripheral Management Language) objects that provides remote printer status information.
- The means of accessing these objects using the PJL (Printer Job Language) passthrough to PML.
- SNMP (Simple Network Management Protocol) protocols.

The PJL passthrough to PML protocol is called "PJL Passthrough" in this document, and gives access to PML objects through the use of PJL commands. SNMP (Simple Network Managemen Protocol) provides access to PML objects through a simple conversion. This document provides concrete examples and usage tips.

References

The following documents contain additional information necessary to understanding this guide:

- *PJL Technical Reference Manual*, edition 9, Hewlett-Packard
- *PML Protocol Specification*, version 2.2, Hewlett-Packard.

Remote Information Needed from the Printer

There are three types of information that drivers and control programs need from the printer.

1. Job control information

This is available through the following PJL commands:

- COMMENT
- ECHO
- ENTER LANGUAGE
- PJL
- UEL
- RESET (to Front Panel settings)
- JOB/EOJ (with the NAME option).

HP recommends the use of these PJL commands.

2. Printer status

Printer status is supported by PML only. This includes errors, media type and ink level status, and job and page processing. Status can be static; status can be **solicited** (host request printer for status) or **unsolicited** (printer notifies host when status changes). Unsolicited status can be obtained by using PML **traps**. PML traps are supported via PJL Passthrough only. SNMP through HP JetDirect cards does not support traps. Instead, the printer is periodically **polled** for status.

3. Printer control

This causes the printer to do something, and includes ink refill and pen check; it currently does not exist in PJL. In this case also, HP recommends the use of either PJL Passthrough to PML or SNMP.

Why Does HP Recommend the Use of PJL Passthrough for Direct Connections? (HP DesignJet 1000, 2000 and 3000 Series Printers)

The information of type **2** or **3** above is typically what PML manages in the printer. Most of the function is already implemented and PML is *the* recommended way of getting remote information, whereas PJL in DesignJets is dedicated to switching personalities, setting job boundaries and specifying job-related settings. This is why some PJL commands are becoming obsolete, and PML is being developed instead.

PML is supported for both direct connections (IEEE-1284-compatible and ECP) and network connections. For direct connections, an additional protocol MLC (Multiple Logical Channels) is needed to support PML. For network connections, the SNMP protocol is used.

As HP does not want to duplicate information in both PJL and PML in the printer, but still is open to the use of PML by third parties that would not want to implement MLC, PJL Passthrough has been defined.

PJL Passthrough is also modular in the sense that if some more information is required in a future product, an existing PML object can be used, or a new PML object implemented, which is a relatively simple process.

What about Network Connections?

On the network, either PJL Passthrough (for the HP DesignJet 1000, 2000 and 3000 Series printers), or PML data translated into SNMP, is supported. For SNMP, an SNMP prefix needs to be added to the PML object ID. The translation between the prefixed PML object ID and SNMP

is done at the SNMP driver level. Because of the availability of off-the-shelf, multi-platform SNMP drivers, and the minimal effort required to obtain printer status, SNMP is the recommended solution for networks. Refer to Appendix D for details on using SNMP, including some questions and answers.

PJL Passthrough Requirements

In this document, we assume that the communication layer already provides physical bidirectionality, in any connection type (parallel or MIO), that is, that the port driver allows the higher layers to read from and write to this physical channel. Only one application at a time is allowed to use this driver. To support traps, the port driver needs a socket which periodically reads the back channel from the printer.

- The PJL Passthrough is **connection-based**, which means that status is available only within an established connection. Only one connection at a time can be established. Since an I/O switch creates a new connection, the old connection is closed and traps for that connection are switched off. This means that the device's PJL parser has to be initialized after every I/O switch, and the traps need to be turned on (as described later in the document)
- PJL Passthrough is **synchronous**; that is, it shares the same I/O path as the data. This means that while sending a file, any host-to-printer request is appended to the end of the file and is processed only after the file has been processed. Note that traps can be sent from the printer to the host at any time since it is using the back channel.
- Some third-party PostScript spoolers do not work with PJL Passthrough and PML because they have problems parsing binary sequences. To use PJL Passthrough, the PostScript spooler must not attempt to parse the PJL Passthrough sequence.

PJL Passthrough Host Requirements

It is necessary to implement some code to physically have bidirectionality, like an **ECP driver** or a **network port driver**. IEEE-1284-compatible only is enough for a direct connection.

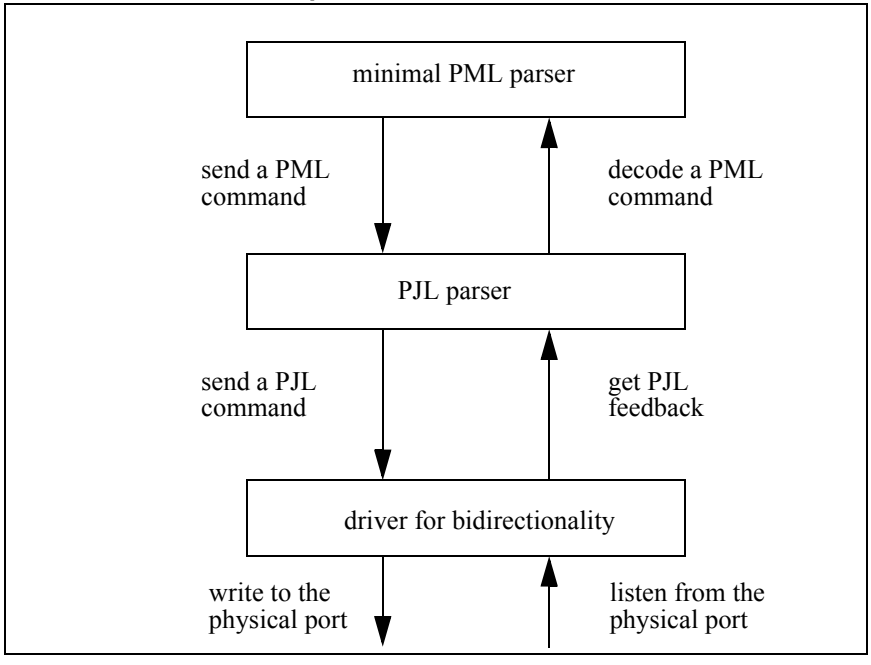
At the level of the information itself (rather than communications), a **PJL parser** is required, as well as a **minimal PML parser** for PJL Passthrough to PML. The PML parser only has to be minimal because just a subset of PML is used to get the required information.

Why is a Minimal PML Parser on the Host Enough?

Only a subset of PML commands is used, and there are few objects. Therefore you can use simple macros instead of coding each command. The commands sent by the device to the host must be decoded. To simplify both requests and replies, you are recommended to limit your code to using only one object per command.

PML errors also can be treated in a minimal way: if the device sends an error code with the reply (that is, the command was unsuccessful), the host can avoid decoding the error to know what happened. As an error is quite rare (after debugging), and is likely to be “action cannot be performed now”, the host can simply try again.

TABLE 1. **Host Side, Implementation Overview**



SNMP Requirements

SNMP is supported in the HP JetDirect cards and is available across any of the physical layers or protocols supported by the HP JetDirect products. This includes Ethernet and Token-ring units, and Novell, TCP/IP, Appletalk, and DLC/LLC. On the host side, an SNMP client or manager is needed to translate the PML object to SNMP for outgoing

requests, and to translate from SNMP to PML for incoming replies. Since SNMP and PML formats are very similar, the PML-to-SNMP translation for most cases requires only adding the SNMP prefix. SNMP drivers and source code are commercially available for most platforms.

SNMP is not connection-based. A network connection does not have to be maintained while sending and receiving SNMP packets. In addition, SNMP requests are **asynchronous**; that is, requests can be sent to the printer any time, independent of other data being sent to it at the same time. SNMP, as supported through HP JetDirect, does not support traps. Instead, **polling** is done. Polling is the process in which the host periodically queries the printer for status of particular PML objects. Depending on the PML object, the host may query for additional objects to get more detail. This way the host can keep track of the changing states of the printer. In general, polling is preferable to traps because connections can be unreliable and trap packets may be lost. It is up to the host to support polling.

To set PML objects using SNMP, HP JetDirect cards must be configured appropriately. Use the HP JetAdmin utility, which is included with the HP JetDirect card or can be accessed from HP's web site.

PJL Passthrough and SNMP Compared

Table 2 compares the benefits and restrictions of PJL Passthrough and SNMP.

TABLE 2. PJL Passthrough and SNMP Compared

	PJL Passthrough	SNMP
Connections	Direct connects and network	Network only
Unsolicited status from printer	Traps (both direct connects and network)	None. Periodic polling is used to get printer status
Printer multiple I/O environment	Traps need to be reinitialized after every I/O switch	No restrictions

TABLE 2. **PJL Passthrough and SNMP Compared (continued)**

	PJL Passthrough	SNMP
Host requirements	For direct connects, either an IEEE-1284-compatible bidirectional driver or an ECP in nibble mode; For network connects, a TCP/IP socket driver. Minimal PJL parser. Minimal PML parser.	SNMP client/manager (commercially available) No parser development
Host-to-printer channel restriction	Synchronous (connection-based)	Asynchronous (not connection-based

Chapter 3 focuses on PJL Passthrough. Refer to Chapter 4 for details of SNMP, and there are some questions and answers in Appendix C.

For both SNMP and PJL Passthrough, only one pending request at a time is supported. Additional requests may be lost.

2. PML Overview

PML is an object-oriented request-reply printer management protocol. It is composed PML commands and PML objects. Each PML object is associated with a unique piece of printer information, while the PML commands specify the way the objects are accessed.

PML Objects

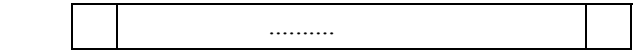
Each object is uniquely specified by its object identifier, which is represented in either text or numeric format. For example, the object which provides the ink level of the black ink supply has the text representation AGENT1_LEVEL and the numeric representation 1.4.1.5.3.1.2. The encoded version of the numeric representation is used to communicate with the printer.

The data type of the object can be one of the following: Enumeration, Collection, Signed Integer, Real, String, and Binary. The table below describes each of the data types; the type may be abbreviated as shown:

TABLE 3. **PML Object Types**

Type	***Description***
Enumeration (“Enum.”)	Finite set of values 0, 1, 2, 3, ...
Collection (“Coll.”)	Consists of a finite set of items. Each item is represented as one bit in a 32-bit word. The bit ordering is specified below.
Signed Integer (“Integer”)	2’s complement signed 32-integer value.
Real	32-bit binary floating point value.
String	Consists of a symbol set and the characters that make up the string.
Binary	Each byte of binary string has value between 0 and 255. The length of the binary string is specified in bytes.

Ordering of Collection Bits



bit 31
Least Significant Bit

bit 0
Most Significant Bit

PML Commands

The PML commands provide the means of accessing PML objects. The **PML commands** that are sent by the host to the device are:

- PmlGetRequest
- PmlSetRequest
- PmlEnableTrapRequest (PJL Passthrough only)
- PmlDisableTrapRequest (PJL Passthrough only).

The **PML commands** that are sent by the device to the host are:

- PmlGetReply
- PmlSetReply
- PmlEnableTrapReply (PJL Passthrough only)
- PmlDisableTrapReply (PJL Passthrough only)
- PmlTrapRequest. No acknowledgement required from the host (PJL Passthrough only).

Execution Outcome of PML Commands

If the device encounters a problem executing the command sent by the host, it reports an error in the reply sent back to the host. It is part of the "execution outcome" that immediately follows the PML command in the reply. The "execution outcome" has the following values:

00 = OK: command executed with no errors

81 = reply buffer overflow error: some results may be lost

82 = command execution error: device encountered a problem while obtaining the result

84 = action unsupported error: attempting to execute a command that the object doesn't support (for example, attempting to SET the ink level of the printer)

85 = invalid value error: attempting to SET an object to an unsupported value.

87 = action cannot be performed now error: the action is supported, but the device is unable to either apply the action or provide a reply in a timely manner. The values of the object queried for, if any, are invalid.

88 = syntax error: the command sent is syntactically incorrect.

3. PJL Passthrough in Detail

In this section, only the information used with PJL Passthrough to PML is described. Note that the HP DesignJet 500, 800 and 5000 Series printers do not support this way of sending and receiving information

In the rest of the document, we use the notation **CR** for carriage return, **LF** for line feed and **FF** for form feed; the escape character is denoted by **ESC**.

PJL Passthrough consists of two commands sent by the host:

> @PJL DMINFO ASCIIHEX="PmlRequest", where DM indicates Device Management, for which the PJL parser on the device sends back the PmlReply to the host
>
> @PJL USTATUS TRAP=ON/OFF, to let the device know that the host wants or no longer wants to receive PML traps.

ASCIIHEX is a keyword meaning that the bytes of the PML command are encoded into an ASCII (non-null terminated) string. The maximum size for a PML command for DesignJets is 64 bytes, so the maximum length for the ASCIIHEX string is 128 bytes.

PJL Passthrough commands sent by the device in response to those from the host are:

> @PJL DMINFO ASCIIHEX="PmlRequest"**CR LF**
> ASCIIHEX="PmlReply"**CR LF FF**
>
> @PJL USTATUS TRAP**CR LF**
> ASCIIHEX="PmlTrapRequest"**CR LF FF**

Retrieving Information Using PmlGetRequest

Below are some types of information that can be retrieved from the printer. Note that the information supported is device-dependent:

- Model number/name
- Printer NOT READY status
- Printer WARNING status
- Printer NOT IDLE (activity) status
- Media type
- Media size
- Ink refill status
- Pen check status
- Page length accuracy
- Ink type
- List of nozzle outs (one per pen)
- Ink level (one per pen or cartridge).

The host sends:

@PJL DMINFO ASCIIHEX="PmlGetRequest"**CR LF**

and receives from the device:

@PJL DMINFO ASCIIHEX="PmlGetRequest"**CR LF**
ASCIIHEX="PmlGetReply"**CR LF FF**

"PmlGetReply" contains the value of the object, as well as a code indicating whether the request was successful or not.

Starting Printer Activity Using PmlSetRequest

These actions can be started in the printer:

- Ink refill
- Maximum nozzle-out threshold to exit pen servicing
- Pen check.

The host sends:

@PJL DMINFO ASCIIHEX="PmlSetRequest"**CR LF**

and receives from the device:

@PJL DMINFO ASCIIHEX="PmlSetRequest"**CR LF**
ASCIIHEX="PmlSetReply"**CR LF FF**

Requesting Unsolicited Information (PML Traps) Using PmlEnableTrapRequest

Traps can be set for these categories of information:

- Printer status/errors/activity (one or more PML objects depending on the level of detail needed)
- Job name (corresponding to the start and end of printing the job)
- Current page printing
- Ink refill status
- Pen check status.

At **initialization only** (that is, at the beginning of the connection and after each I/O switch)**,** the host needs to send:

@PJL USTATUS TRAP=ON**CR LF**

to enable the usage of traps in general.

For all the objects listed above the host must send a PmlEnableTrap command to indicate the objects whose traps he wants to receive:

@PJL DMINFO ASCIIHEX="PmlEnableTrapRequest"**CR LF**

The reply from the device is:

@PJL DMINFO ASCIIHEX="PmlEnableTrapRequest"**CR LF**
ASCIIHEX="PmlEnableTrapReply"**CR LF FF**

Then, as long as the connection is maintained, each time the value of the object changes, the device sends to the host the corresponding PML traps with the following format:

@PJL USTATUS TRAP**CR LF**
ASCIIHEX="PmlTrapRequest"**CR LF FF**

Note: The PJL code inside the device sends the PmlTrapReply back to the PML parser in the device, so the host does not have to acknowledge the trap.

Before closing connection, to disable traps in general, the host sends:

@PJL USTATUS TRAP=OFF**CR LF**

and disables the PML traps for each object:

@PJL DMINFO ASCIIHEX="PmlDisableTrapRequest"**CR LF**

The host receives from the device:

@PJL DMINFO ASCIIHEX="PmlDisableTrapRequest"**CR LF**
ASCIIHEX="PmlDisableTrapReply"**CR LF FF**

4. SNMP In Detail

As already mentioned, off-the-shelf SNMP drivers are available for various platforms, and they can help minimize the amount of development on the host application by preparsing the PML objects and checking for syntax problems.

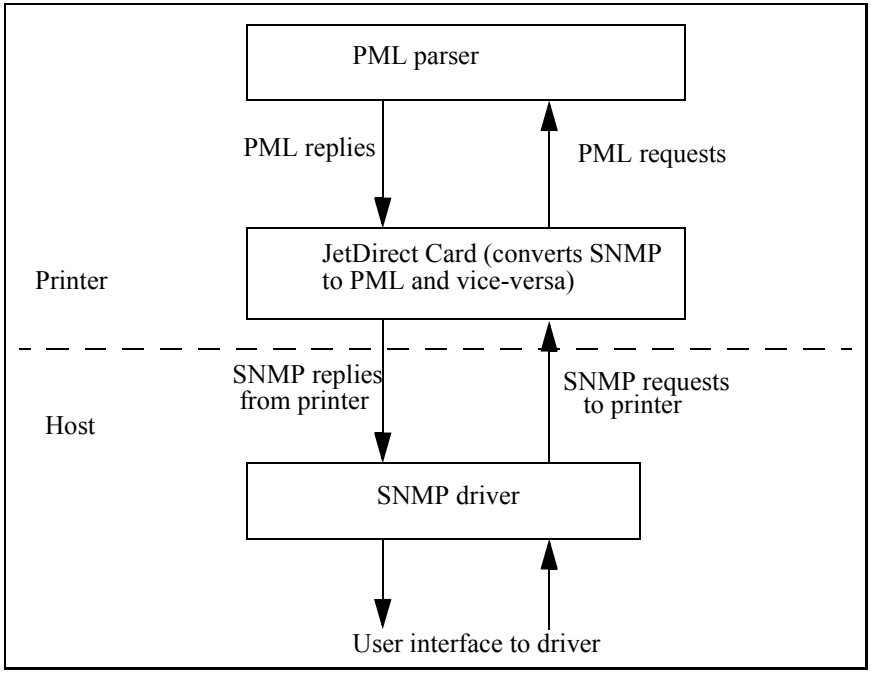
The SNMP commands provide the means of accessing SNMP objects. These SNMP commands are used by the host to access SNMP objects from the device:

- snmpset: this is similar to the PMLSetRequest and allows for starting activities on the device.
- snmpget: this is similar to the PMLGetRequest and is used to get information from the device

The exact syntax of the commands depends on the SNMP driver implementation.

SNMP to PML Implementation

TABLE 4. SNMP to PML Translation, Overview



PML to SNMP Conversion

SNMP objects, similar to PML objects, are assigned unique object IDs. The HP JetDirect imlementation supports a simple conversion from PML to SNMP objects, which consists of adding an SNMP prefix and a suffix to the PML object ID.

The prefix is the same for all objects:

1.3.6.1.4.1.11.2.3.9.4.2

and the suffix is always “0”.

For example, the SNMP object ID for AGENT1_LEVEL (with PML object ID of 1.4.1.5.3.1.2) is 1.3.6.1.4.1.11.2.3.9.4.2.1.4.1.5.3.1.2.0.

Refer to the specific SNMP driver implementation for details for accessing SNMP objects and the syntax of the SNMP replies.

Because of its ease of use and fewer restrictions, SNMP is recommended for network connections.

- Since the HP JetDirect implementation currently does not support SNMP traps, it is recommended that polling is done instead. As previously mentioned, polling is simply sending a status request at fixed internals (for example, every 5 or 10 seconds) for specific SNMP objects. If more detail is needed, additional queries can be made for these objects. Refer to the SNMP example for more detail.
- Since SNMP does not go through the PJL parser, SNMP requests can be processed by the printer even if the PJL parser is blocked.

5. PML Objects Used with HP DesignJet Printers

Below are the PML objects supported by the different DesignJet printers (indicated by •), classified according to the type of information or activity. Note that not all the objects or status bits of an object have to be supported in the host, only for those which are deemed important by the developer.

Tables 5 through 11 are only applicable to the HP DesignJet 1000, 2000 and 3000 Series printers. (See page 31 for the HP DesignJet 500, 800 and 5000 Series.)

TABLE 5. Printer Miscellaneous Information (HP DesignJet 1000, 2000, 3000 Series)

PML object Object ID	Object type Access right	Values taken, comments	1050C 1055CM	2000CP 2500CP	3000CP 3500CP
MODEL_NUMBER 1.1.3.1	String Get	Returns the device identification: C4723A (for the HP DesignJet 3000CP) C4724A (for the HP DesignJet 3500CP C6074A (for the HP DesignJet 1050C C6075A (for the HP DesignJet 1055CM)	•		•
FW_ROM_-REVISION 1.1.3.6	String Get	Returns the firmware level in the form a.xx.xxn where n is alphabetic	•		•
TAKE_UP_REEL_-INSTALLED 1.4.1.4.1.5	Enum. Get	Returns the value of the Take-up reel setting on the front panel: 1=OFF 2=ON Returns an “ObjectNotSupported” error for devices that don’t support take-up reels			•
TOTAL_RAM_SIZE 1.1.2.21	Integer Get	Returns the base RAM memory configuration in the printer	•		

TABLE 5. **Printer Miscellaneous Information (HP DesignJet 1000, 2000, 3000 Series) (continued)**

PML object Object ID	Object type Access right	Values taken, comments	1050C 1055CM	2000CP 2500CP	3000CP 3500CP
PAGE_LENGTH_-ACCURACY 1.4.1.1.7	Enum. Get, set	Returns front-panel setting of page length accuracy (PAD): 1 = exact (no PAD) 2 = optimized (SMART PAD) 3 = constant (fixed PAD) Refer to the *User's Guide* for your HP DesignJet for more detail on page length accuracy.		•	•

TABLE 6. **Pen Information (HP DesignJet 1000, 2000, 3000 Series)**

PML object Object ID	Object type Access right	Values taken, comments	1050C 1055CM	2000CP 2500CP	3000CP 3500CP
MARKING_-AGENT_REFILL 1.4.1.5.1.5	Coll. Set, get, trap	Triggers printer to refill the ink cartridges. Each bit corresponds to a pen. Since refill does a refill for all four pens, all four bits must be set. If no bit is set, no refill is pending or being done (refer to the examples for more details).		•	•
AGENT1_REFILL_-STATUS 1.4.1.5.3.1.8	Enum. Get, trap	Returns status of ink refill: 1 = waiting for refill to begin 2 = unknown. This is returned when no refill has been done since boot-up 3 = refill in progress 4 = refill has completed successfully 6 = refill has failed.		•	•
MARKING_-AGENT_TEST 1.4.1.5.1.6	Coll. Get, set, trap	Trigger printer to do a pen check. The printer prints a nozzle pattern, scans it for failed nozzles, and does pen servicing if required. Since a pen check applies to all four pens, all four bits must be set. Refer to "Remote Error Hiding for DesignJet 2xxxCP and 3xxxCP" on page 41 for more detail.		•	•

TABLE 6. **Pen Information (HP DesignJet 1000, 2000, 3000 Series) (continued)**

PML object Object ID	Object type Access right	Values taken, comments	1050C 1055CM	2000CP 2500CP	3000CP 3500CP
AGENT1_TEST_-STATUS 1.4.1.5.3.1.9	Enum. Get, trap	Returns status of pen check: 1 = waiting for pen check to begin 2 = unknown. This is returned when no pen check has been done since boot-up 3 = pen check in progress 4 = pen check successful 6 = pen check failed.		•	•
MARKING_-AGENT_NOZZLE_-SERVICE_-THRESHOLD 1.4.1.5.1.7	Integer Set, get	This is used with the pen check. Specifies the maximum bad-nozzle threshold allowed. If the number of bad nozzles exceeds this threshold after pen servicing has been done, user intervention is required. Refer to “Remote Error Hiding for DesignJet 2xxxCP and 3xxxCP” on page 41 for more detail.		•	•
AGENT1_BAD_-NOZZLE_STATUS_-PART1 1.4.1.5.3.1.10	Binary Get	Returns first part of list of nozzle outs for the **black** pen. Refer to “Remote Error Hiding for DesignJet 2xxxCP and 3xxxCP” on page 41 for more detail.	•	•	•
AGENT1_BAD_-NOZZLE_STATUS_-PART2 1.4.1.5.3.1.11	Binary Get	Returns second part of list of nozzle outs for the **black** pen.	•	•	•
AGENT1_BAD_-NOZZLE_STATUS_-PART3 1.4.1.5.3.1.16	Binary Get	Returns third part of list of nozzle outs for the **black** pen.	•		
AGENT2_BAD_-NOZZLE_STATUS_-PART1 1.4.1.5.3.2.10	Binary Get	Returns first part of list of nozzle outs for the **cyan** pen.	•	•	•
AGENT2_BAD_-NOZZLE_STATUS_-PART2 1.4.1.5.3.2.11	Binary Get	Returns second part of list of nozzle outs for the **cyan** pen.	•	•	•

TABLE 6. **Pen Information (HP DesignJet 1000, 2000, 3000 Series) (continued)**

PML object Object ID	Object type Access right	Values taken, comments	1050C 1055CM	2000CP 2500CP	3000CP 3500CP
AGENT2_BAD_-NOZZLE_STATUS_-PART3 1.4.1.5.3.2.16	Binary Get	Returns third part of list of nozzle outs for the **cyan** pen.	•		
AGENT3_BAD_-NOZZLE_STATUS_-PART1 1.4.1.5.3.3.10	Binary Get	Returns first part of list of nozzle outs for the **magenta** pen.	•	•	•
AGENT3_BAD_-NOZZLE_STATUS_-PART2 1.4.1.5.3.3.11	Binary Get	Returns second part of list of nozzle outs for the **magenta** pen.	•	•	•
AGENT3_BAD_-NOZZLE_STATUS_-PART3 1.4.1.5.3.3.16	Binary Get	Returns third part of list of nozzle outs for the **magenta** pen.	•		
AGENT4_BAD_-NOZZLE_STATUS_-PART1 1.4.1.5.3.4.10	Binary Get	Returns first part of list of nozzle outs for the **yellow** pen.	•	•	•
AGENT4_BAD_-NOZZLE_STATUS_-PART2 1.4.1.5.3.4.11	Binary Get	Returns second part of list of nozzle outs for the **yellow** pen.	•	•	•
AGENT4_BAD_-NOZZLE_STATUS_-PART3 1.4.1.5.3.4.16	Binary Get	Returns third part of list of nozzle outs for the **yellow** pen.	•		

TABLE 6. **Pen Information (HP DesignJet 1000, 2000, 3000 Series) (continued)**

PML object Object ID	Object type Access right	Values taken, comments	1050C 1055CM	2000CP 2500CP	3000CP 3500CP
MARKING_-AGENT1_LEVEL 1.4.1.5.3.1.14	Integer Get	Returns the amount of usable ink in the **black** (K) pen before a refill is needed. The value is in microliters (10^{-6} liters). The error "ActionCannotBePerformedNow" is returned if the printer is initializing a new ink delivery system, printing or refilling (and the request will need to be repeated at a later time). The total ink usage, including spitting, is allowed for in the value.			•
MARKING_-AGENT2_LEVEL 1.4.1.5.3.2.14	Integer Get	Returns the amount of usable ink in the **cyan** pen before a refill is needed.			•
MARKING_-AGENT3_LEVEL 1.4.1.5.3.3.14	Integer Get	Returns the amount of usable ink in the **magenta** pen before a refill is needed.			•
MARKING_-AGENT4_LEVEL 1.4.1.5.3.4.14	Integer Get	Returns the amount of usable ink in the **yellow** pen before a refill is needed.			•

TABLE 7. **Media Information (HP DesignJet 1000, 2000, 3000 Series)**

PML object Object ID	Object type Access right	Values taken, comments	1050C 1055CM	2000CP 2500CP	3000CP 3500CP
TRAY1_MEDIA_-SIZE_LOADED 1.4.1.3.3.1.1	Enum. Get	Discrete media size in TRAY1. For the 1050C and 1055CM, TRAY1 always refers to the roll feeder and TRAY2 to the sheet feeder. HP DesignJets return 101 (custom) for sheet media and 32766 (custom roll) for roll media. To determine the actual media size, use TRAYx_CUSTOM_MEDIA_LENGTH and TRAYx_CUSTOM_MEDIA_WIDTH.	•		

TABLE 7. **Media Information (HP DesignJet 1000, 2000, 3000 Series) (continued)**

PML object Object ID	Object type Access right	Values taken, comments	1050C 1055CM	2000CP 2500CP	3000CP 3500CP
TRAY1_MEDIA_-TYPE 1.4.1.3.3.1.6	Enum. Get	Values are product-specific. See "Appendix B. PML Media Type Values Supported" on page 60.	•	•	•
TRAY1_CUSTOM_-MEDIA_WIDTH 1.4.1.3.3.1.10	Integer Get	Width of media (in decipoints) loaded in TRAY1. (1 inch = 720 decipoints).	•	•	•
TRAY1_CUSTOM_-MEDIA_LENGTH 1.4.1.3.3.1.11	Integer Get	Length of media (in decipoints) loaded in TRAY1.	•		
TRAY2_MEDIA_-SIZE_LOADED 1.4.1.3.3.2.1	Enum. Get	Discrete media size in TRAY2. For the 1050C and 1055CM, TRAY1 always refers to the roll feeder and TRAY2 to the sheet feeder.	•		
TRAY2_MEDIA_-TYPE 1.4.1.3.3.2.6	Enum. Get	Values are product-specific. See "Appendix B. PML Media Type Values Supported" on page 60.	•		•
TRAY2_CUSTOM_-MEDIA_WIDTH 1.4.1.3.3.2.10	Integer Get	Width of media (in decipoints) loaded in TRAY2. (1 inch = 720 decipoints).	•		
TRAY2_CUSTOM_-MEDIA_LENGTH 1.4.1.3.3.2.11	Integer Get	Length of media (in decipoints) loaded in TRAY2.	•		

TABLE 8. **Ink Supply Information (HP DesignJet 1000, 2000, 3000 Series)**

PML object Object ID	Object type Access right	Values taken, comments	1050C 1055CM	2000CP 2500CP	3000CP 3500CP
AGENT1_LEVEL 1.4.1.5.3.1.2	Integer Get	Percentage of ink remaining in reservoir of **black** ink. Note that there is potentially a 20% inaccuracy for all four inks.		•	
AGENT2_LEVEL 1.4.1.5.3.2.2	Integer Get	Percentage of ink remaining in reservoir of **cyan** ink.		•	
AGENT3_LEVEL 1.4.1.5.3.3.2	Integer Get	Percentage of ink remaining in reservoir of **magenta** ink.		•	

TABLE 8. **Ink Supply Information (HP DesignJet 1000, 2000, 3000 Series) (continued)**

PML object Object ID	Object type Access right	Values taken, comments	1050C 1055CM	2000CP 2500CP	3000CP 3500CP
AGENT4_LEVEL 1.4.1.5.3.4.2	Integer Get	Percentage of ink remaining in reservoir of **yellow** ink.		•	
AGENT3_CLASS_ID 1.4.1.5.3.3.1	String Get	Type of ink in the third cartridge (assuming all cartridges have the same ink): "Dye" = dye-based ink "UV-Pigmented" = UV-pigmented-based ink. The first two characters (always 0x0115) correspond to the symbol set.		•	•
AGENT_SUPPLY1_-USED 1.4.1.5.4.1.5	Integer Get	Returns the total amount of ink, in microliters, spent by the ink delivery system for the **black** ink system. This object is used for accounting. The driver should request the amount of ink used before and after a print, and use the difference for tracking. There is a 20% inaccuracy. The error "ActionCannotBePerformedNow" is returned if the printer is initializing or is initializing a new ink delivery system. The total ink usage, including spitting and priming, is allowed for in the value.			•
AGENT_SUPPLY2_-USED 1.4.1.5.4.2.5	Integer Get	Returns the total amount of ink, in microliters, spent by the ink delivery system for the **cyan** ink system.			•
AGENT_SUPPLY3_-USED 1.4.1.5.4.3.5	Integer Get	Returns the total amount of ink, in microliters, spent by the ink delivery system for the **magenta** ink system.			•
AGENT_SUPPLY4_-USED 1.4.1.5.4.4.5	Integer Get	Returns the total amount of ink, in microliters, spent by the ink delivery system for the **yellow** ink system.			•
AGENT_SUPPLY1_-LEVEL 1.4.1.5.4.1.1	Integer Get	Returns the total amount of ink in the **black** ink delivery system as a percentage. There is a 20% inaccuracy.	•		•

TABLE 8. **Ink Supply Information (HP DesignJet 1000, 2000, 3000 Series) (continued)**

PML object Object ID	Object type Access right	Values taken, comments	1050C 1055CM	2000CP 2500CP	3000CP 3500CP
AGENT_SUPPLY2_-LEVEL 1.4.1.5.4.2.1	Integer Get	Returns the total amount of ink in the **cyan** ink delivery system as a percentage. There is a 20% inaccuracy.	•		•
AGENT_SUPPLY3_-LEVEL 1.4.1.5.4.3.1	Integer Get	Returns the total amount of ink in the **magenta** ink delivery system as a percentage. There is a 20% inaccuracy.	•		•
AGENT_SUPPLY4_-LEVEL 1.4.1.5.4.4.1	Integer Get	Returns the total amount of ink in the **yellow** ink delivery system as a percentage. There is a 20% inaccuracy.	•		•

TABLE 9. **Printer NOT READY Information (HP DesignJet 1000, 2000, 3000 Series)**

PML object Object ID	Object type Access right	Values taken, comments	1050C 1055CM	2000CP 2500CP	3000CP 3500CP
NOT_READY_-PRINTER 1.1.2.2	Coll. Get, trap	Reminder: collections start at bit 0! . bit 3 = parser error . bit 4 = destination print engine error (if bit 4 is set, more detail about the error can be retrieved from the NOT_READY_DESTIN-ATION_PRINT_ENGINE object, see next) . bit 7 = system error (firmware error)	all bits	all bits	all bits

TABLE 9. **Printer NOT READY Information (HP DesignJet 1000, 2000, 3000 Series) (continued)**

PML object Object ID	Object type Access right	Values taken, comments	1050C 1055CM	2000CP 2500CP	3000CP 3500CP
NOT_READY_- DESTINATION_- PRINT_ENGINE 1.4.1.2.1	Coll. Get, trap	Reminder: collections start at bit 0! . bit 0 = door open . bit 1 = internal media jam . bit 6 = pen missing . bit 8 = incorrect pen installed . bit 11 = tray media jam . bit 13 = requested media unavailable . bit 14 = out of media . bit 15 = unknown print engine error . bit 16 = pen test failure (bad pen) . bit 20 = media lever in wrong position . bit 26 = media misaligned, reload . bit 28 = media in wrong format . bit 29 = media mispositioned, reload . bit 30 = media edge not detected . bit 31 = if this bit is set, refer to the NOT_READY_DESTINATION_-PRINT-ENGINE_PART2 object.	all bits	bits 0, 1, 6, 8, 11, 14, 20, 25	bits 0, 1, 6, 8, 11, 14, 20, 25
NOT_READY_- DESTINATION_- PRINT_ENGINE_- PART2 1.4.1.2.28	Coll. Get, trap	This is only referred to if bit 31 of NOT_READY-DESTINATION-PRINT_ENGINE is set. Reminder: collections start at bit 0! . bit 0 = ink supply empty . bit 1 = ink supply missing . bit 2 = incorrect ink supply installed . bit 3 = ink supply failure (bad ink supply) . bit 8 = pen cleaner missing . bit 9 = pen cleaner incorrect . bit 10 = pen cleaner failure (bad pen cleaner)	all bits		

TABLE 10. **Printer Status Information (HP DesignJet 1000, 2000, 3000 Series)**

PML object Object ID	Object type Access right	Values taken, comments	1050C 1055CM	2000CP 2500CP	3000CP 3500CP
STATUS_PRINTER 1.1.2.22	Coll. Get, trap	Reminder: collections start at bit 0! . bit 3 = memory out warning . bit 4 = destination print engine warning (if bit 4 is set, more detail about the warning can be obtained from STATUS_DESTINATION_PRINT_-ENGINE object, see next) . bit 7 = continuable system error	all bits		
STATUS_-DESTINATION_-PRINT_ENGINE 1.4.1.2.8	Coll. Get, trap	Reminder: collections start at bit 0! . bit 14 = ready for media . bit 18 = replace pen . bit 31 = if this bit is set, refer to STATUS_DESTINATION_PRINT_-ENGINE_PART2 object (next).	all bits		
STATUS_-DESTINATION_-PRINT_ENGINE_-PART2 1.4.1.2.29	Coll. Get, trap	This object is only referred to if bit 31 of STATUS_DESTINATION_PRINT_ENGIN E is set. Reminder: collections start at bit 0! . bit 6 = agent supply low (less than 15% ink left) . bit 7 = agent supply nearly out (less than 5% ink left) . bit 12 = design life of pen reached.	all bits		

TABLE 11. **Printer Not Idle (Activity) Information (HP DesignJet 1000, 2000, 3000 Series)**

PML object Object ID	Object type Access right	Values taken, comments	1050C 1055CM	2000CP 2500CP	3000CP 3500CP
NOT_IDLE 1.1.2.4	Coll. Get, trap	Reminder: collections start at bit 0! . bit 3 = parsing . bit 4 = destination print engine activity (if bit 4 is set, more detail about the activity can be obtained from NOT_IDLE_DESTINATION_-PRINT_-ENGINE object, see next)	all bits		
NOT_IDLE_-DESTINATION_-PRINT_ENGINE 1.4.1.2.2	Coll. Get, trap	Reminder: collections start at bit 0! . bit 0 = drying media . bit 1 = printing . bit 2 = accessing pen . bit 3 = aligning pen . bit 6 = loading media . bit 7 = unloading media . bit 10 = checking pens . bit 11 = nesting plots . bit 12 = cancelling print . bit 15 = busy (general) . bit 16 = replacing print kit (pens, supplies, leaners)	all bits		

TABLE 12. **Printer Job Information (HP DesignJet 1000, 2000, 3000 Series)**

PML object Object ID	Object type Access right	Values taken, comments	1050C 1055CM	2000CP 2500CP	3000CP 3500CP
CURRENT_JOB_-PRINTING1_NAME1 1.1.6.2.2.1.2.1	String, Get, trap	Updated when the job starts printing; the first two characters (always 0x0115) correspond to the symbol set	•	•	•
CURRENT_-PRINTING_JOB_-PAGES_PRINTED 1.1.6.1.7	String, Get, trap	Updated when a page starts printing; the integer value is equal to the job ID number times 65536 plus the page number. For example, if page 4 in job 10 just started printing, then the value would be 65536*10+4 = 655364. Recommendation: extract the page number and use the job ID	•	•	•

Tables 12 through 19 are only applicable to the HP DesignJet 500, 800 and 5000 Series printers.

TABLE 13. **Printer Miscellaneous Information (HP DesignJet 500, 800, 5000 Series)**

PML object Object ID	Object type Access right	Values taken, comments	500 Series	800 Series	5000 Series
MODEL_NUMBER 1.1.3.1	String Get	Returns the device identification: C6096A (for the HP DesignJet 5000PS 60”) C6091A (for the HP DesignJet 5000PS 42”) C6095A (for the HP DesignJet 5000 60”) C6090A (for the HP DesignJet 5000 42”) C7769B (for the HP DesignJet 500 24”) C7770B (for the HP DesignJet 500 42”) C7779B (for the HP DesignJet 800 24”) C7780B (for the HP DesignJet 800 42”) C7779C (for the HP DesignJet 800PS 24”) C7780C (for the HP DesignJet 800PS 42”) C7769C (for the HP DesignJet 500PS 24”) C7770C (for the HP DesignJet 500PS 24”)	•	•	•
FW_ROM_-REVISION 1.1.3.6	String Get	Returns the firmware level in the form a.xx.xxn where n is alphabetic	•	•	•
MIO1_MODEL_-NUMBER 1.1.4.3.1.1	String Get	Returns the product number of the formatter card plugged into the MIO slot.	•	•	
MIO1_MANUFACT-URING_INFO 1.1.4.3.1.3	String Get	Returns the firmware version of the formatter card plugged into the MIO slot.	•	•	
TAKE_UP_REEL_-INSTALLED 1.4.1.4.1.5	Enum. Get	Returns the status of the Take-up reel: 1=OFF (not installed, or installed but not used) 2=ON (installed and used) Returns an “ObjectNotSupported” error for devices that don’t support take-up reels			•

TABLE 13. **Printer Miscellaneous Information (HP DesignJet 500, 800, 5000 Series) (continued)**

PML object Object ID	Object type Access right	Values taken, comments	500 Series	800 Series	5000 Series
TOTAL_RAM_SIZE 1.1.2.21	Integer Get	Returns the base RAM memory configuration in the printer in bytes	•	•	•

TABLE 14. **Media Information (HP DesignJet 500, 800, 5000 Series)**

PML object Object ID	Object type Access right	Values taken, comments	500 Series	800 Series	5000 Series
MEDIA_PROFILE_-VENDOR_n 1.4.1.12.1.1.n	String Get	It returns the media vendors' name placed in the "n" position currently downloaded in the printer, in the same language than the printer has been set up. Uses ASCII characters 32 through 122. See footnote (a).			•
MEDIA_PROFILE_-NAME_n 1.4.1.12.1.2.n	String Get	It returns the media name placed in the "n" position currently downloaded in the printer, in the same language than the printer has been set up. Uses ASCII characters 32 through 122. See footnote (a).			•
MEDIA_PROFILE_-LOCALIZED_-VENDOR_n 1.4.1.12.1.3.n	String Get	It returns the media vendors' name placed in the "n" position currently downloaded in the printer, in the original language the media vendor uses. Uses the Unicode symbol set. See footnote (a).			•
MEDIA_PROFILE_-LOCALIZED_-NAME_n 1.4.1.12.1.4.n	String Get	It returns the media name placed in the "n" position currently downloaded in the printer in the original language the media vendor uses. Uses the Unicode symbol set. See footnote (a).			•
MEDIA_PROFILE_-CONSUMPTION_-VENDOR_n 1.4.1.12.2.1.n	String Get	It returns the media vendors' name placed in the "n" position currently downloaded in the printer, in the same language than the printer has been set up. Uses ASCII characters 32 through 122. See footnote (a).			•

TABLE 14. **Media Information (HP DesignJet 500, 800, 5000 Series) (continued)**

PML object Object ID	Object type Access right	Values taken, comments	500 Series	800 Series	5000 Series
MEDIA_PROFILE_-CONSUMPTION_-NAME_n 1.4.1.12.2.2.n	String Get	It returns the media name placed in the "n" position currently downloaded in the printer, in the same language than the printer has been set up. Uses ASCII characters 32 through 122. See footnote (a).			•
MEDIA_PROFILE_-LOCALIZED_-CONSUMPTION_-VENDOR_n 1.4.1.12.2.3.n	String Get	It returns the media vendors' name placed in the "n" position currently downloaded in the printer, in the original language the media vendor uses. Uses the Unicode symbol set. See footnote (a).			•
MEDIA_PROFILE_-LOCALIZED_-CONSUMPTION_-NAME_n 1.4.1.12.2.4.n	String Get	It returns the media name placed in the "n" position currently downloaded in the printer in the original language the media vendor uses. Uses the Unicode symbol set. See footnote (a).			•
MEDIA_PROFILE_-CONSUMPTION_-AMOUNT_n 1.4.1.12.2.5.n	Integer Get	Accumulated consumption for the whole life of the printer (in square feet). See footnote (a).			•
TRAY1_MEDIA_-VENDOR 1.4.1.3.3.1.13	String Get	Name of the media vendor who owns the current loaded media. Uses ASCII characters 32 through 122.	•	•	•
TRAY1_MEDIA_-NAME 1.4.1.3.3.1.4	String Get	Name of the current loaded media. Uses ASCII characters 32 through 122.	•	•	•
TRAY1_MEDIA_-SIZE_LOADED 1.4.1.3.3.1.1	Enum Get	Tells if the loaded media has a standard size or a custom size. Return Values: 101 → Sheet loaded. 32766 → Roll loaded.	•	•	•
TRAY1_CUSTOM_-MEDIA_WIDTH 1.4.1.3.3.1.10	Integer Get	Width (in decipoints) of media loaded in TRAY1. (1 inch = 720 decipoints).	•	•	•

TABLE 14. **Media Information (HP DesignJet 500, 800, 5000 Series) (continued)**

PML object Object ID	Object type Access right	Values taken, comments	500 Series	800 Series	5000 Series
TRAY1_CUSTOME_-MEDIA_LENGTH 1.4.1.3.3.1.11	Integer Get	Length (in decipoints) of media loaded in TRAY1. (1 inch = 720 decipoints).	•	•	•

a. There are two lists of MEDIA_PROFILE_... information; the list with _CONSUMPTION_ keeps records of deleted media, the other doesn't. If a deleted media definition is downloaded again, its consumption amount resumes at the previously saved value.

TABLE 15. **Ink Supply Information (HP DesignJet 500, 800, 5000 Series)**

PML object Object ID	Object type Access right	Values taken, comments	500 Series	800 Series	5000 Series
NUMBER_OF_-MARKING_AGENTS 1.4.1.5.2.1	Integer Get	It will return the number of pens that the printer has (4 or 6)	•	•	•
AGENTx-CLASS-ID 1.4.1.5.3.x.1	String Get	It will return what kind of ink is being used by pen "x": *Example:* AGENT3-CLASS-ID = 1.4.1.5.3.3.1 will return what kind of ink is used by pen number 3, which, as we can see in the instruction below corresponds to the black pen. Possible Return Values: "10": Black ink for the HP DesignJet 500 and 800 Series printers. "82": Dye inks for the HP DesignJet 500 and 800 Series printers. "81": Dye inks for HP DesignJet 5000 Series "83": UV inks for HP DesignJet 5000 Series	•	•	•

TABLE 15. **Ink Supply Information (HP DesignJet 500, 800, 5000 Series) (continued)**

PML object Object ID	Object type Access right	Values taken, comments	500 Series	800 Series	5000 Series
AGENTx-COLORS 1.4.1.5.3.x.3	Coll. Get	It will return which colour is associated with each number. *Example*: AGENT2-COLORS=1.4.1.5.3.2.3 and will return 2, Yellow). Values: Cyan=0 Magenta=1 Yellow=2 Black=3 LightCyan=4 (Only for 5000 Series) LightMagenta=5 (Only for 5000 Series)	•	•	•
AGENT-SUPPLYx-LEVEL 1.4.1.5.4.x.1	Integer Get	It returns the total amount of ink in the delivery system, as a percentage. There may be an inaccuracy of 20%.	•	•	•
AGENT-SUPPLYx-USED 1.4.1.5.4.x.5	Integer Get	It will return the amount of used ink for the ink delivery system (color, **x**), in microliters. If you check this value before and after starting the job, you will know how much ink has been wasted in the job.	•	•	•
AGENT_-CONSUMPTION_-CLASS_ID_x_n 1.4.1.5.6.1.x.n	String Get	For each class of ink (**n**) and for each ink delivery system (color, **x**) there is a record of the ink consumption during the whole life of the printer.	•	•	•
AGENT_-CONSUMPTION_-COLORS_x_n 1.4.1.5.6.2.x.n	Coll. Get	For each class of ink (**n**) and for each ink delivery system (color, **x**) there is a record of the ink consumption during the whole life of the printer.	•	•	•
AGENT_-CONSUMPTION_-AMOUNT_x_n 1.4.1.5.6.3.x.n	Integer Get	For each class of ink (**n**) and for each ink delivery system (color, **x**) there is a record of the ink consumption during the whole life of the printer. Measured in milliliters.	•	•	•

TABLE 16. **Printer NOT READY Information (HP DesignJet 500, 800, 5000 Series)**

PML object Object ID	Object type Access right	Values taken, comments	500 Series	800 Series	5000 Series
NOT_READY_-PRINTER 1.1.2.2	Coll. Get	Reminder: collections start at bit 0! . bit 3 = parser error . bit 4 = destination print engine error (if bit 4 is set, more detail about the error can be retrieved from the NOT_READY_DESTINATION_-PRINT_-ENGINE object, see next) . bit 7 = blocking system error	all bits	all bits	all bits
NOT_READY_-DESTINATION_-PRINT_ENGINE 1.4.1.2.1	Coll. Get	Reminder: collections start at bit 0! . bit 0 = door open . bit 1 = internal media jam . bit 6 = pen missing . bit 8 = incorrect pen installed . bit 11 = tray media jam . bit 13 = requested media unavailable . bit 14 = out of media . bit 15 = unknown print engine error . bit 18 = pen test failure (bad pen) . bit 20 = media lever in wrong position . bit 26 = media misaligned, reload . bit 28 = media in wrong format . bit 29 = media mispositioned, reload . bit 30 = media edge not detected . bit 31 = if this bit is set, refer to the NOT_READY_DESTINATION_-PRINT-ENGINE_PART2 object.	all bits	all bits	all bits

TABLE 16. **Printer NOT READY Information (HP DesignJet 500, 800, 5000 Series) (continued)**

PML object Object ID	Object type Access right	Values taken, comments	500 Series	800 Series	5000 Series
NOT_READY_- DESTINATION_- PRINT_ENGINE_- PART2 1.4.1.2.28	Coll. Get	This is only referred to if bit 31 of NOT_READY-DESTINATION-PRINT_ENGINE is set. Reminder: collections start at bit 0! . bit 0 = ink supply empty . bit 1 = ink supply missing . bit 2 = incorrect ink supply installed . bit 3 = ink supply failure (bad ink supply) . bit 8 = pen cleaner missing . bit 9 = pen cleaner incorrect	all bits	all bits	all bits

TABLE 17. **Printer Status Information (HP DesignJet 500, 800, 5000 Series)**

PML object Object ID	Object type Access right	Values taken, comments	500 Series	800 Series	5000 Series
STATUS_PRINTER 1.1.2.22	Coll. Get	Reminder: collections start at bit 0! . bit 3 = memory out warning . bit 4 = destination print engine warning (if bit 4 is set, more detail about the warning can be obtained from STATUS_DESTINATION_PRINT_-ENGINE object, see next) . bit 7 = continuable or blocking system error	all bits	all bits	all bits

TABLE 17. **Printer Status Information (HP DesignJet 500, 800, 5000 Series) (continued)**

PML object Object ID	Object type Access right	Values taken, comments	500 Series	800 Series	5000 Series
STATUS_-DESTINATION_-PRINT_ENGINE 1.4.1.2.8	Coll. Get	Reminder: collections start at bit 0! . bit 0 = Door Open . bit 1 = Internal Media Jam . bit 6 = Missing pen . bit 8 = Incorrect pen installed . bit 14 = ready for media . bit 15 = Unknown print engine error . bit 18 = replace pen . bit 20 = Media lever in wrong position . bit 26 = Media alignment required . bit 28 = Media in wrong format . bit 29 = Media mispositioned . bit 30 = Media edge not detected . bit 31 = if this bit is set, refer to STATUS_DESTINATION_PRINT_-ENGINE_PART2 object (next).	all bits	all bits	all bits
STATUS_-DESTINATION_-PRINT_ENGINE_-PART2 1.4.1.2.29	Coll. Get	This object is only referred to if bit 31 of STATUS_DESTINATION_PRINT_ENGIN E is set. Reminder: collections start at bit 0! . bit 0 = agent supply out . bit 1= agent supply missing . bit 2 = agent supply incorrect . bit 3= agent supply failure . bit 6 = agent supply low (less than 15% ink left) . bit 7 = agent supply nearly out (less than 5% ink left) . bit 8 = pen cleaner missing . bit 9 = pen cleaner incorrect . bit 12 = design life of pen reached.	all bits	all bits	all bits

TABLE 18. **Printer Not Idle (Activity) Information (HP DesignJet 500, 800, 5000 Series)**

PML object Object ID	Object type Access right	Values taken, comments	500 Series	800 Series	5000 Series
NOT_IDLE 1.1.2.4	Coll. Get	Reminder: collections start at bit 0! . bit 3 = parsing . bit 4 = destination print engine activity (if bit 4 is set, more detail about the activity can be obtained from NOT_IDLE_DESTINATION_-PRINT_-ENGINE object, see next)	all bits	all bits	all bits
NOT_IDLE_-DESTINATION_-PRINT_ENGINE 1.4.1.2.2	Coll. Get	Reminder: collections start at bit 0! . bit 0 = drying media . bit 1 = printing . bit 2 = accessing pen . bit 3 = aligning pen . bit 6 = loading media . bit 7 = unloading media . bit 10 = checking pens . bit 11 = nesting plots . bit 12 = cancelling print . bit 15 = busy (general) . bit 16 = replacing print kit (pens, supplies, leaners)	all bits	all bits	all bits

TABLE 19. **Printer Job Information (HP DesignJet 500, 800, 5000 Series)**

PML object Object ID	Object type Access right	Values taken, comments	500 Series	800 Series	5000 Series
CURRENT_JOB_-PRINTING1_NAME1 1.1.6.2.2.1.2.1	String, Get	Updated when the job starts printing; the first two characters (always 0x0115) correspond to the symbol set	•	•	•
CURRENT_JOB_-PRINTING1_NAME2	String, Get	Updated when the job starts printing; the first two characters (always 0x0115) correspond to the symbol set If the job name is too long, this object returns the last part of the name.	•	•	•

TABLE 19. **Printer Job Information (HP DesignJet 500, 800, 5000 Series) (continued)**

PML object Object ID	Object type Access right	Values taken, comments	500 Series	800 Series	5000 Series
CURRENT_-PRINTING_JOB_-PAGES_PRINTED 1.1.6.1.7	String, Get	Updated when a page starts printing; the integer value is equal to the job ID number times 65536 plus the page number. For example, if page 4 in job 10 just started printing, then the value would be 65536*10+4 = 655364. Recommendation: extract the page number and use the job ID	•	•	•

PML General Guidelines

For NOT-READY, STATUS, and NOT-IDLE objects, more than one bit in the collection can be set, depending on the event detected by the printer.

The pens, supplies, and cleaners have the following convention: PEN1 = black, PEN2 = cyan, PEN3 = magenta, and PEN4 = yellow.

Guidelines for 3000CP and 3500CP

Since the values returned by the ink delivery system may have errors of up to 20%, there are cases when the value of **AGENT_SUPPLYx_USED** may exceed the 380 cc capacity of the ink system.

Users should not switch ink delivery systems between printers; doing so incurs a 25% reduction in calculated ink levels.

For images larger than A0 or E-size, and for print modes with coverages of more than 100% per color, it is recommended that ink levels in the pens are checked between jobs and that a refill is triggered if the print heads are not full. See *Mid-print refilling* in the *Product Comparison Guide*. The estimated printable areas at 100% ink density are:

TABLE 20. **Estimated printable areas at 100% density before refill**

	Imaging Ink	UV Ink
Ink level not checked, printheads not topped up before starting	8.5 sq ft or 0.8 sq m.	8 sq ft or 0.75 sq m.
Printheads topped up before starting	12 sq ft or 1.1 sq m.	9.2 sq ft or 0.85 sq m.

Remote Error Hiding for DesignJet 2xxxCP and 3xxxCP

By using the MARKING_AGENT_TEST, AGENT1_TEST_STATUS, MARKING_AGENT_NOZZLE_SERVICE_THRESHOLD, and the AGENT1_BAD_NOZZLE_STATUS_PART1 (and other similar) objects, the host has the capability to trigger the printing of a nozzle pattern, do pen servicing, and receive a list of nozzles that have failed. These objects provide enough information for drivers to do error hiding on the host through the substitution of bad nozzles. Details of how this can be done is beyond the scope of this document. Note that error hiding is done by Varware only, where the mask can be specified. Below is a detailed description of each object.

MARKING_AGENT_TEST:

This object triggers the printer to do a **pen check**: that is, the printer prints and scans a nozzle pattern for each pen. If any nozzle is out, the printer enters **pen servicing**. Once in servicing mode, the printer repeats the print and scan of the pattern and continues servicing the pens until all nozzles work or three pen servicings have been done. If the former occurs, the pen check terminates successfully. If the latter occurs, and the number of nozzle outs exceeds the threshold specified by the object MARKING_AGENT_NOZZLE_SERVICE_-THRESHOLD, the pen check fails and the printer changes into NOT READY status with a print quality warning requiring user intervention to bring it back to READY state.

MARKING_AGENT_NOZZLE_SERVICE_THRESHOLD:

This object specifies the maximum number of failed nozzles that are acceptable on leaving pen servicing (as a result of a pen check). This applies only in the case of leaving a pen check after three servicings have been done and there is at least one nozzle out. If the number of nozzle outs exceeds the threshold value, the pen check fails and the printer goes into NOT READY state, requiring user intervention. As a guideline, this object is sent prior to a pen check and should be set to the maximum number of failed nozzles that the host application can substitute through error hiding. If the object is not set, the default value is 3.

Note: The maximum threshold value is 16 for dye-based ink and 24 for UV-pigment ink. Once set, the threshold value is good until the printer reboots.

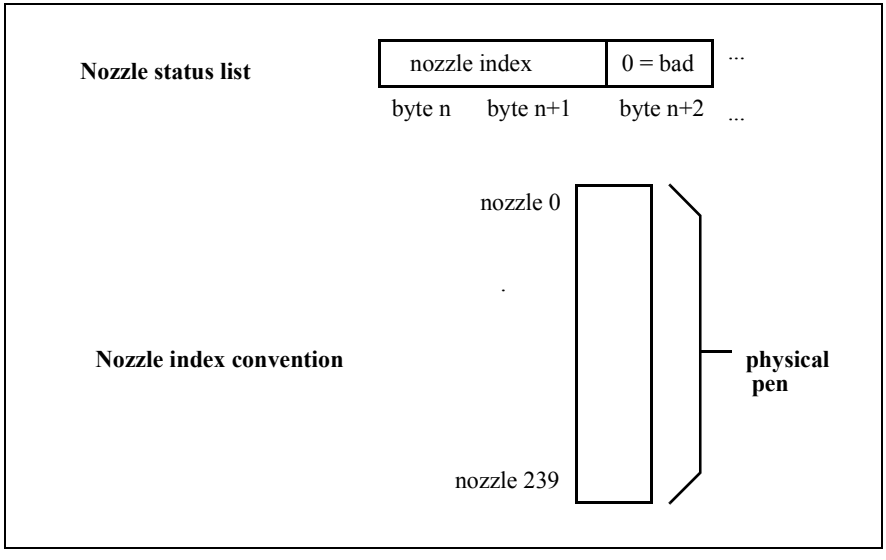
AGENT1_TEST_STATUS:

Returns the status of the pen check process. Since a pen check affects all four pens, the status of only one pen (in this case, the black pen) is needed. Refer to the PML object table for more detail.

AGENT1_BAD_NOZZLE_STATUS_PART1:
AGENT1_BAD_NOZZLE_STATUS_PART2:

These objects contain the list of failed nozzles in the black pen as a result of doing a pen check. Up to 28 failed nozzles are reported, and both objects have to be referenced to get all 28 nozzles: PART1 contains the first 14 failed nozzles, and PART2 contains the second 14 failed nozzles. Table 21 on page 42 shows the format of the list (in binary) and the nozzle index with respect to the actual pen:

TABLE 21. Format of the Nozzle Status List



AGENT2_BAD_NOZZLE_STATUS_PART1:
AGENT2_BAD_NOZZLE_STATUS_PART2:
AGENT3_BAD_NOZZLE_STATUS_PART1:
AGENT3_BAD_NOZZLE_STATUS_PART2:
AGENT4_BAD_NOZZLE_STATUS_PART1:
AGENT4_BAD_NOZZLE_STATUS_PART2:

These are the same as for the black pen, but they specify the list of nozzle outs for the cyan, magenta, and yellow pens, respectively.

Note: the nozzle index may be byte-swapped—check for whichever byte contains significant information.

Guidelines for 1050C and 1055CM

TRAY1 always refers to the roll feed, and TRAY2 always refers to the sheet feed, regardless of the media currently loaded.

"Door" is the servicing station door, "cover" is the top cover which is opened to load sheet media, and "lever" is the the roll lever. The *cDoorOpen(0)* bit in NOT-READY-DESTINATION-PRINT-ENGINE is set for the below cases:

- Door open (always)
- Cover open if media is already loaded (both when printing and idle)

The *PaperLeverInWrongPosition(20)* bit is set if the lever is lifted when media is already loaded, both when printing and idle. If no media is loaded and the cover or lever is lifted, the *LoadingMedia(6)* bit is set in the NOT-IDLE-DESTINATION-PRINT-ENGINE.

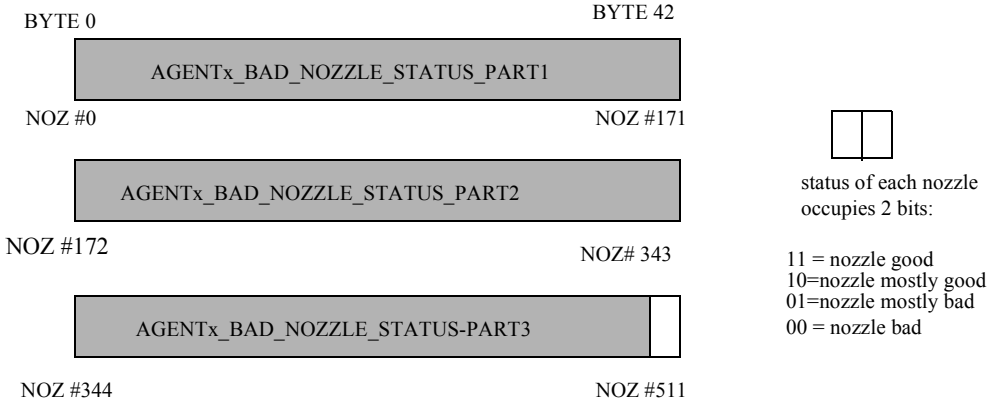
If no media is loaded in a tray, querying its SIZE and TYPE will result in an *ActionCannotBePerformedNow* error. Querying CUSTOM-MEDIA-WIDTH and CUSTOM-MEDIA-LENGTH will result in values of 0.

The AGENTx_BAD_NOZZLE_STATUS_PARTy objects provide the list of bad nozzles. Since the pens for these printers have 512 nozzles, three objects are needed in order to return the status of all the nozzles of a pen. No servicing or pen check is provided due to the complexity of these algorithms. Remote error hiding is supported by Varware only. Below is the usage:

Before sending a plot:

- Host requests printer for the list of bad nozzles (determined by the printer's last pen check).
- Printer returns the list of nozzle outs (the format is specified below).
- The host assembles the replies (in order PART1, PART2, and PART3 for each pen) and changes the masks depending on the nozzle outs.
- Host sends Varware job.

Format of list of nozzle outs (as returned by printer):



6. PJL Passthrough Examples

Refer to the PML Protocol Specification for details about the syntax of PML commands.

Get Media Width

The host sends:

@PJL DMINFO ASCIIHEX="0000070104010303010A"**CR LF**

00: get request
00: object ID follows
07: object ID length
0104010303010A: the object ID
(TRAY1_CUSTOM_MEDIA_WIDTH, 1.4.1.3.3.1.10).

and receives from the device:

@PJL DMINFO ASCIIHEX="0000070104010303010A"**CR LF**
ASCIIHEX="800000070104010303010A08025FA0"**CR LF FF**

80: get reply
00: no error
00: object ID follows
07: object ID length
0104010303010A: the object ID (TRAY1_CUSTOM_MEDIA_WIDTH)
08: type is integer
02: the length of the value field (that follows) is 2 bytes
5FA0: value = 24480 decipoints <=> 34 inches (width of E-size media).

Enable Status Objects

The host sends:

@PJL DMINFO ASCIIHEX="05000401010202"**CR LF**

05: enable trap request
00: object ID follows
04: object ID length
01010202: the object ID (NOT_READY_PRINTER).

and receives from the device:

@PJL DMINFO ASCIIHEX="05000401010202"**CR LF**
ASCIIHEX="85000004010102022000"**CR LF FF**

85: enable trap reply
00: no error
00: object ID follows
04: object ID length
01010202: the object ID (NOT_READY_PRINTER)
20: type is collection
00: value = 0 (no error at all, the printer is ready).

Similarly with NOT_READY_DESTINATION_PRINT_ENGINE:

the PmlEnableTrapRequest is 0500050104010201
and the PmlEnableTrapReply is 8500000501040102012000.

Receiving PML traps

In the example, the printer goes from the ready status to out-of-ink error. When the status changes, traps are sent for the NOT_READY_PRINTER and NOT_-READY_DESTINATION_PRINT_ENGINE objects.

@PJL USTATUS TRAP**CR LF**
ASCIIHEX="0700000401010202200110"**CR LF FF**

07: trap request
00: no error
00: object ID follows
04: object ID length
01010202: the object ID (NOT_READY_PRINTER)
20: type is collection
01: length of the value field is 1 byte
10: bit 4 is set, there is a destination print engine error.

@PJL USTATUS TRAP**CR LF**
ASCIIHEX="0700000501040102012001 80"**CR LF FF**

07: trap request
00: no error
00: object ID follows
05: object ID length
0104010201: the object ID
(NOT_READY_DESTINATION_PRINT_ENGINE)

20: type is collection
01: length of the value field is 1 byte
80: bit 7 is set; the printer is out of ink.

Note: sometimes if two different traps are returned at the same time, they can be appended to each other:

@PJL USTATUS TRAP**CR LF**
ASCIIHEX="0700000401010202200110000501040102
01200180"**CR LF FF**

Reminder: the host does not have to send any PmlTrapReply, the device PJL parser itself acknowledges the trap.

Ink Refill

The MARKING_AGENT_REFILL object allows the host to trigger a refill of the pens on the printer, and the AGENT1_REFILL_STATUS object provides status of the refill activity. A refill request is queued until the printer is ready to do a refill (for example, until it finishes printing a job). The example below shows how the host can trigger a refill and keep track of its status by using both traps and polling.

To set the traps, the host enables the trap capability (note that this only has to be done once for each I/O switch):

@PJL SET USTATUS TRAP=ON**CR LF**

Then it sends the enable trap requests for the refill request and its status:

@PJL DMINFO ASCIIHEX="050006010401050105"**CR LF**

05: enable trap request
00: object ID follows
06: object ID length
010401050105: the object ID (MARKING_AGENT_REFILL).

@PJL DMINFO ASCIIHEX="0500070104010503010 8"**CR LF**

05: enable trap request
00: object ID follows
07: object ID length
01040105030108: the object ID (AGENT1_REFILL_STATUS).

The printer returns the enable trap replies:

@PJL DMINFO ASCIIHEX="050006010401050105"**CR LF**
ASCIIHEX="850000060104010501052000"**CR LF**

85: enable trap reply
00: no error
00: object ID follows
06: object ID length
010401050105: the object ID (MARKING_AGENT_REFILL)
20: type is collection
00: value = 0 (no refill is pending or being done).

@PJL DMINFO ASCIIHEX="05000701040105030108"**CR LF**
ASCIIHEX="850000070104010503010804010 2"**CR LF**

85: enable trap reply
00: no error
00: object ID follows
07: object ID length
01040105030108: the object ID (AGENT1_REFILL_STATUS)
04: type is enumeration
01: length of the value field is 1 byte
02: status is "unknown" (no refill has been done since boot-up).

The host now triggers a set request for refill:

@PJL DMINFO ASCIIHEX="040006010401050105 20010F"**CR LF**

04: set request
00: object ID follows
06: object ID length
010401050105: the object ID (MARKING_AGENT_REFILL)
20: type is collection
01: length of value is 1 byte
0F: set collection bits for all 4 pens 00001111(binary)

and the printer returns a set reply:

@PJL DMINFO ASCIIHEX="040006010401050105 20010F"**CR LF**
ASCIIHEX="840000060104010501052 0010F"**CR LF**

84: set reply
00: no error
00: object ID follows
06: object ID length
010401050105: the object ID (MARKING_AGENT_REFILL)
20: type is collection
01: length of value is 1 byte
0F: set collection bits for all 4 pens 00001111(binary)

Since the REFILL and REFILL_STATUS values have changed, the printer sends a trap request (in this case concatenated; note that two 0s separate the concatenated traps):

@PJL USTATUS TRAP**CR LF**
ASCIIHEX="0700060104010501052001 0F00
0701040105030108040101"**CR LF**

07: trap request
00: object ID follows
06: object ID length
010401050105: the object ID (MARKING_AGENT_REFILL)
20: type is collection
01: length of value is 1 byte
0F: set collection bits for all 4 pens 00001111(binary)
00: object ID follows
07: object ID length
01040105030108: the object ID (AGENT1_REFILL_STATUS)
04: type is enumeration
01: length of value is 1 byte
01: waiting for refill to begin

When the actual refill commences, the printer sends another trap request:

@PJL USTATUS TRAP**CR LF**
ASCIIHEX="07000701040105030108040103"**CR LF**

07: trap request
00: object ID follows
07: object ID length
01040105030108: the object ID (AGENT1_REFILL_STATUS)
04: type is enumeration

01: length of data is 1 byte
03: ink refill is in progress.

At the end of refill, if it has completed successfully, the printer sends the following trap requests:

@PJL USTATUS TRAP**CR LF**
ASCIIHEX="07000701040105030108040104"**CR LF**

07: trap request
00: object ID follows
07: object ID length
01040105030108: the object ID (AGENT1_REFILL_STATUS)
04: type is enumeration
01: length of data is 1 byte
04: ink refill has completed successfully.

@PJL USTATUS TRAP**CR LF**
ASCIIHEX="0700060104010501052000"**CR LF**

07: trap request
00: object ID follows
06: object ID length
010401050105: the object ID (MARKING_AGENT_REFILL)
20: type is collection
00: value = 0 (no refill is pending or being done).

Note: Instead of traps, the host can periodically send get requests for both objects to determine the status of the refill operation.

Pen Check

The process for a pen check is similar to that of a refill. Before producing printout that requires high ink quality, the host sends a pen check to check whether there are any nozzle outs, and if so, which ones they are so that remote error hiding can be done.

The host first specifies the nozzle-out threshold. As a guideline, the threshold corresponds to the number of nozzle outs that can be supported by the host error-hiding application (for this example, assume that the host supports up to 6 nozzle outs and the black pen initially has eight nozzle outs).

Note: The threshold is set only after a boot-up or for changing the existing threshold.

The host sets the nozzle-out threshold to 8:

@PJL DMINFO ASCIIHEX="040006010401050107080108"**CR LF**

04: set request
00: object ID follows
06: object ID length
010401050107: the object ID
(MARKING_AGENT_NOZZLE_SERVICE_THRESHOLD)
08: type is integer
01: length of value is 1 byte
08: threshold value.

The printer returns a set reply:

@PJL DMINFO ASCIIHEX="040006010401050107080108"**CR LF**
ASCIIHEX="840000060104010501070801 08"**CR LF**

84: set reply
00: no error
00: object ID follows
06: object ID length
010401050107: the object ID
(MARKING_AGENT_NOZZLE_SERVICE_THRESHOLD)
08: type is integer
01: length of value is 1 byte
08: threshold value.

To set the traps, the host enables the trap capability (note that this only has to be done once for each I/O switch):

@PJL SET USTATUS TRAP=ON**CR LF**

Then it sends the enable trap requests for the pen check request and its status:

@PJL DMINFO ASCIIHEX="050006010401050106"**CR LF**

05: enable trap request
00: object ID follows
06: object ID length
010401050106: the object ID (MARKING_AGENT_TEST).

@PJL DMINFO ASCIIHEX="05000701040105030109"**CR LF**

05: enable trap request
00: object ID follows
07: object ID length
01040105030109: the object ID (AGENT1_TEST_STATUS).

The printer returns the enable trap replies:

@PJL DMINFO ASCIIHEX="050006010401050106"**CR LF**
ASCIIHEX="8500000601040105010620000"**CR LF**

85: enable trap reply
00: no error
00: object ID follows
06: object ID length
010401050106: the object ID (MARKING_AGENT_TEST)
20: type is collection
00: value = 0 (no pen check in progress).

@PJL DMINFO ASCIIHEX="05000701040105030109"**CR LF**
ASCIIHEX="850000070104010503010904010 2"**CR LF**

85: enable trap reply
00: no error
00: object ID follows
07: object ID length
01040105030109: the object ID (AGENT1_TEST_STATUS)
04: type is enumeration
01: length of the value field is 1 byte
02: status is "unknown" (no refill has been done since boot-up).

The host now triggers a set request for pen check:

@PJL DMINFO ASCIIHEX="04000601040105010620010F"**CR LF**

04: set request
00: object ID follows
06: object ID length
010401050106: the object ID (MARKING_AGENT_TEST)

20: type is collection
01: length of value is 1 byte
0F: set collection bits for all 4 pens 00001111(binary)

and the printer returns a set reply:

@PJL DMINFO ASCIIHEX="040006010401050106200 10F"**CR LF**
ASCIIHEX="840000060104010501062001 0F"**CR LF**

84: set reply
00: no error
00: object ID follows
06: object ID length
010401050106: the object ID (MARKING_AGENT_TEST)
20: type is collection
01: length of value is 1 byte
0F: set collection bits for all 4 pens 00001111(binary)

Since the TEST and TEST_STATUS values have changed, the printer sends a trap request (in this case concatenated; note that two 0s separate the concatenated traps):

@PJL USTATUS TRAP**CR LF**
ASCIIHEX="070006010401050106200 10F00
070104010503010904010 1"**CR LF**

07: trap request
00: object ID follows
06: object ID length
010401050106: the object ID (MARKING_AGENT_TEST)
20: type is collection
01: length of value is 1 byte
0F: set collection bits for all 4 pens 00001111(binary)
00: object ID follows
07: object ID length
01040105030109: the object ID (AGENT1_TEST_STATUS)
04: type is enumeration
01: length of value is 1 byte
01: waiting for pen check to begin

When the actual pen check commences, the printer sends another trap request:

@PJL USTATUS TRAP**CR LF**
ASCIIHEX="0700070104010503010904010 3"**CR LF**

07: trap request
00: object ID follows
07: object ID length
01040105030109: the object ID (AGENT1_TEST_STATUS)
04: type is enumeration
01: length of data is 1 byte
03: pen check is in progress.

After the first pattern is printed, the printer scans it and finds that the magenta pen (pen 3) has 10 nozzle outs. The printer enters servicing mode (because there is at least 1 nozzle out). After three servicings, there are still 7 nozzle outs, and the printer exits servicing mode. Since this is less than the threshold, the pen check completes successfully.

The printer sends a trap request indicating pen check has succeeded.

@PJL USTATUS TRAP**CR LF**
ASCIIHEX="0700070104010503010904010 4"**CR LF**

07: trap request
00: object ID follows
07: object ID length
01040105030109: the object ID (AGENT1_TEST_STATUS)
04: type is enumeration
01: length of data is 1 byte
04: pen check successful.

@PJL USTATUS TRAP**CR LF**
ASCIIHEX="0700060104010501062000"**CR LF**

07: trap request
00: object ID follows
06: object ID length
010401050106: the object ID (MARKING_AGENT_TEST)
20: type is collection
00: value = 0 (no pen check in progress).

Now the host sends a get request to get the list of nozzle outs for each of the pen, and receive a corresponding reply from the printer.

The host sends a get request for pen #1 (black pen):

@PJL DMINFO ASCIIHEX="000007010401050301 0A"**CR LF**

00: get request
00: object ID follows
07: object ID length
0104010503010A: the object ID (AGENT1_BAD_NOZZLE_STATUS_PART1).

and receives from the printer:

@PJL DMINFO ASCIIHEX="000007010401050301 0A"**CR LF**
ASCIIHEX="8000000701040105030 10A1C00"**CR LF**

80: get reply
00: no error
00: object ID follows
07: object ID length
0104010503010A: the object ID (AGENT1_BAD_NOZZLE_STATUS_PART1)
1C: type is null
00: length is 0 (no bad nozzles).

The host sends a get request for pen #2 (cyan pen):

@PJL DMINFO ASCIIHEX="000007010401050302 0A"**CR LF**

and receives from the printer:

@PJL DMINFO ASCIIHEX="000007010401050302 0A"**CR LF**
ASCIIHEX="8000000701040105030 20A1C00"**CR LF**

The host sends a get request for pen #3 (magenta pen):

@PJL DMINFO ASCIIHEX="000007010401050303 0A"**CR LF**

and receives from the printer:

@PJL DMINFO ASCIIHEX="000007010401050303 0A"**CR LF**
ASCIIHEX="8000000701040105030 30A14150000000D0000500000
5D0000830000A00000AD0000"**CR LF**

The host sends a get request for pen #4 (yellow pen):

@PJL DMINFO ASCIIHEX="0000070104010503040A"**CR LF**

and receives from the printer:

@PJL DMINFO ASCIIHEX="0000070104010503040A"**CR LF**
ASCIIHEX="800000070104010503040A1C00"**CR LF**

From the results returned by the printer, the black, cyan, and yellow pens have no nozzle outs; the magenta pen has seven nozzle outs:

80: get reply
00: no error
00: object ID follows
07: object ID length
0104010503030A: the object ID
(AGENT3_BAD_NOZZLE_STATUS_PART1)
14: type is binary
15: length of the binary sequence (21 bytes)
(remaining bytes): nozzles 0 (0000), 13(0D00), 80 (5000), 93 (5D00), 131 (8300),
160 (A000), and 173 (AD00) are out (00).

Note: The nozzle index may be byte-swapped (as in this case).

7. SNMP Examples

As mentioned in Chapter 1, the SNMP utilities provide the SNMP commands to access the SNMP objects, and the HP JetDirect card converts the SNMP objects to PML objects. Some utilities will parse the syntax of the response and provide only the data itself. Since each utility behaves differently, no concrete examples can be given for SNMP. However, guidelines for using specific features are provided.

Polling Guidelines in SNMP

Polling can be used in order to keep track of the current printer status. This consists of querying status of specific PML objects that keep track of the printer state such as NOT-READY, STATUS, and NOT-IDLE. The polling frequency depends on which status is being queried, the number of the objects queried, and the network connection being used. In general, to minimize network traffic, query for as few objects as possible and as infrequently as possible while maintaing a current status of the printer.

For example, to detect when the printer goes NOT READY, the host queries for the NOT-READY-PRINTER object once every 10 seconds. If a bit is set in the response, the printer is in NOT READY state. Depending on the bit set, the host may query for the NOT-READY-DESTINATION-PRINT-ENGINE object to get more detail on why the printer went NOT READY (media out, etc.).

Appendix A. PJL Passthrough Usage Tips

- It it is recommended that the PJL parser be reinitialized prior to sending individual or group of PJL Passthrough commands. This is to ensure that the current parser is in the correct state to parse PJL commands. For example, to get the printer NOT-READY status, prefix and append the UEL (Universal End of Language) command:

  ESC%-12345X@PJL**CR LF**
  @PJL DMINFO ASCIIHEX=”00000401010202”**CR LF**
  ESC%-12345X@PJL

- As already mentioned, every time there is an I/O switch, the traps need to be reinitialized. If the printer has multiple connections (for example, MIO and parallel), traps should be reinitialized before each I/O session.
- In general, the usage of traps should not be considered reliable, because the underlying I/O may not be reliable and may lose packets. When possible, use polling to get status instead of traps.
- QUEUEING should be set to ON, since this allows for PJL to be parsed and passthough commands to be processed when the printer is busy (for example, doing a pen check or pen refill). For certain conditions such as system errors and printing KCMY files, the parsing of PJL commands are blocked and status requests are not processed until the parsing is unblocked. In this case traps are useful in indicating the current status of the printer. In general, PJL Passthrough commands cannot be processed if the PJL parser is blocked.
- Duplicate traps requests may be returned by the printer for certain printer states. It is recommended that the application on the host filters out duplicate trap messages.
- The threshold can be set higher than the number of failed nozzles the error-hiding software is capable of supporting. This is because some prints may be acceptable even with some nozzle outs (for example, for printing drafts). It is your responsibility to determine what image quality is acceptable.
- The host should have only one pending get request at a time. It should wait for a get reply from the printer before sending another get request.

- A pen check request should be sent before each print that requires the highest ink quality possible to ensure that there are no nozzle outs, or if there are, that they can be covered by error hiding from the host application.

Appendix B. PML Media Type Values Supported

Below are the supported HP media (indicated by •) and their corresponding PML values (in hexadecimal). Different HP DesignJets support different media types.

TABLE 22. Media Types for Dye-Based Ink

Media Type	PML values	1050C 1055CM	2000CP 2500CP	3000CP 3500CP
Plain Paper	0x00000003	[2]		
White Inkjet Paper	0x00007FDF	[2]		
Thin Natural Tracing Paper	0x00007FE0	[2]		
Natural Tracing Paper	0x00007FF0	[2]		
Coated Paper	0x00007FF1	[2]	[2]	[2]
Clear Film	0x00007FF2	[2]		
High Gloss Photo	0x00007FF3	[2]	[2]	[2]
Semi-Gloss Photo	0x00007FF4		[2]	[2]
Heavy Coated Paper	0x00007FF6	[2]	[2]	[2]
Vellum	0x00007FFC	[2]		
Translucent Bond	0x00007FFD	[2]		
Matte Film	0x00007FFE	[2]		
Imaging Film Backlit	0x00007FEE		[2]	[2]
High Gloss Film	0x00007FF5		[2]	[2]
Opaque Vinyl	0x00007FED		[2]	[2]

TABLE 23. Media Types for UV-Pigment-Based Inks

Media Type	PML values	1050C 1055CM	2000CP 2500CP	3000CP 3500CP
UV-Opaque Vinyl	0x00007FE3		[2]	[2]
Heavy Coated Paper	0x00007FF6		[2]	[2]
UV Custom Media A	0x00007FE6		[2]	[2]
UV Custom Media B	0x00007FE7		[2]	[2]
UV Custom Media C	0x00007FE8		[2]	[2]
UV Custom Media D	0x00007FE9		[2]	[2]

Appendix C. SNMP Usage Tips

Some Questions and Answers for PJL Passthrough and SNMP.

What are the Advantages of Using SNMP (Compared with PJL Passthrough) on a Network?

SNMP is asynchronous, that is, the printer can be queried at any time, even while a job is being sent. PJL is synchronous, so that PJL Passthrough requests are queued until parsing and printing of data is finished. Furthermore, if the printer is in a “not ready” state (for example, there is a paper jam or system error), PJL commands are blocked, whereas SNMP requests are not.

For IEEE-1284-compatible bidirectional mode interfaces, the printer-to-host channel is always open. This means that unsolicited status can always be sent from the printer, even while a job is being sent.

Any Disadvantages?

SNMP does not support traps through the JetDirect card. The remedy is to use a polling technique. In fact, for networked connections, this may be more reliable than using traps, since network packets may be lost. Polling is more secure, because the host will always be expecting a response from the printer once a request is sent.

Does PJL Passthrough Support Traps over a Network?

Yes, with some restrictions. Traps are supported only while a connection is established between the host and the printer. In that case, traps can be enabled and received while the connection is active. At the end of the connection, all traps are set off and would need to be reenabled for another connection. I/O switches or time-outs automatically close the connections and default the trap settings. The I/O time-out duration depends on the network protocol.

Since the printer accepts only one connection at a time, you should keep the connection open only when necessary, that is, only when sending and printing files. It is only at that time that traps are needed anyway, as status requests can be sent at other times.

How can I Obtain SNMP Drivers?

SNMP is a standard protocol for managing devices. SNMP publications are widely available on the World Wide Web (refer to the Internet Engineering Task Force RFC 1157 as a starting point; see `http://www.ietf.org`). SNMP libraries and utilities exist for various platforms,

including Macintosh, UNIX and Windows NT. SNMP libraries are available with the latest version of Microsoft's Windows NT Developer's Package.

The effort required to implement an SNMP solution varies depending on the operating system and interfaces used; generally the effort required is slightly less than that for a PJL Passthrough solution.