[MS-WPRN]:
Web Point-and-Print Protocol Specification

Revision Summary

Date	Revision History	Revision Class	Comments
05/11/2007	0.1		MCPP Milestone 4 Initial Availability
08/10/2007	1.0	Major	Added normative reference.
09/28/2007	1.0.1	Editorial	Revised and edited the technical content.
10/23/2007	1.0.2	Editorial	Revised and edited the technical content.
11/30/2007	1.0.3	Editorial	Revised and edited the technical content.

Date	Revision History	Revision Class	Comments
01/25/2008	1.0.4	Editorial	Revised and edited the technical content.

Table of Contents

1 Introduction

This document is a specification of the Microsoft-proprietary Web Point-and-Print Protocol. The Web Point-and-Print Protocol is **Hypertext Transfer Protocol (HTTP)**–based (as specified in [RFC2616]) and is used by clients to download **printer driver** software either from a server in the client network or from a Web site, or directly by print devices. This allows distribution of printer driver software using standard Web technologies.

This document includes the following:

- How messages are transported and message syntax in section 2.
- Protocol details including abstract data models and message processing rules in section 3.
- Protocol examples in section 4.
- Security considerations for implementers in section 5.
- An appendix of Windows behavior in section 6.
- An index in section 7.

1.1 Glossary

The following terms are defined in [MS-GLOS]:

ASCII
DEVMODE
Driver Package
Hypertext Transfer Protocol (HTTP)
Hypertext Transfer Protocol over Secure Socket Layer (HTTPS)
INF File
Little-Endian
Print Server
Print System
Printer Driver
Proxy
Registry
Unicode
Universal Naming Convention (UNC)
Uniform Resource Locator (URL)
UTF-16LE

The following terms are specific to this document:

CAB File: See **cabinet file**.

Cabinet File: A file with the suffix .cab that acts as a container for other files. A **cabinet file** serves as a compressed archive for a group of files. For more information on the format of **cabinet files**, see [MSDN-CAB].

Digest: A number that uniquely identifies a given file such as a fingerprint. The **digest** is used for security and validation purposes to avoid multiple security warnings when downloading from the Internet.

Internet Printing Protocol (IPP): A standard protocol for printing and for the management of print jobs and printer settings over the Internet. It is built on the **Hypertext Transfer Protocol (HTTP)**.

Multi-String: An array of null-terminated 16-bit **Unicode** strings (**UTF-16LE**–encoded) with one additional null after the final string, which is null-terminated in it own right. Thus, there are actually two nulls at the end of a **multi-string** structure.

Multisz: A data type defining a **multi-string**.

MAY, SHOULD, MUST, SHOULD NOT, MUST NOT: These terms (in all caps) are used as specified in [RFC2119]. All statements of optional behavior use either MAY, SHOULD, or SHOULD NOT.

1.2 References

1.2.1 Normative References

We conduct frequent surveys of the normative references to assure their continued availability. If you have any issue with finding a normative reference, please contact dochelp@microsoft.com. We will assist you in finding the relevant information. Please check the archive site, http://msdn2.microsoft.com/en-us/library/E4BD6494-06AD-4aed-9823-445E921C9624, as an additional source.

[MS-GLOS] Microsoft Corporation, "Windows Protocols Master Glossary", March 2007.

[MS-RPRN] Microsoft Corporation, "Print System Remote Protocol Specification", June 2007.

[RFC793] Postel, J., "Transmission Control Protocol: DARPA Internet Program Protocol Specification", RFC 793, September 1981, http://www.ietf.org/rfc/rfc0793.txt

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, http://www.ietf.org/rfc/rfc2119.txt

[RFC2396] Berners-Lee, T., Fielding, R., and Masinter, L., "Uniform Resource Identifiers (URI): Generic Syntax", RFC 2396, August 1998, http://www.ietf.org/rfc/rfc2396.txt

[RFC2616] Fielding, R., et al., "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999, http://www.ietf.org/rfc/rfc2616.txt

[RFC2617] Franks, J., Hallam-Baker, P., Hostetler, J., Lawrence, S., Leach, P., Luotonen, A., and Stewart, L., "HTTP Authentication: Basic and Digest Access Authentication", RFC 2617, June 1999, http://www.ietf.org/rfc/rfc2617.txt

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, May 2000, http://www.ietf.org/rfc/rfc2818.txt

1.2.2 Informative References

[MSDN-CAB] Microsoft Corporation, "Microsoft Cabinet SDK", March 1997, http://msdn2.microsoft.com/en-us/library/ms974336.aspx

[MSDN-INF] Microsoft Corporation, "About INF Files", http://msdn2.microsoft.com/en-us/library/aa376858.aspx

[MSDN-PRNINF] Microsoft Corporation, "Printer INF Files", http://msdn2.microsoft.com/en-us/library/aa506008.aspx

[MSDN-RTV] Microsoft Corporation, "Registry Value Types", http://msdn2.microsoft.com/en-us/library/ms724884.aspx

[RFC2781] Hoffman, P. and Yergeau, F., "UTF-16, an encoding of ISO 10646", RFC 2781, February 2000, http://www.ietf.org/rfc/rfc2781.txt

[RFC2910] Herriot, R., Ed., Butler, S., Moore, P., Turner, R., and Wenn, J., "Internet Printing Protocol/1.1: Encoding and Transport", RFC 2910, September 2000, http://www.ietf.org/rfc/rfc2910.txt

[RFC2911] Hastings, T., Ed., Herriot, R., deBry, R., Isaacson, S., and Powell, P., "Internet Printing Protocol/1.1: Model and Semantics", RFC 2911, September 2000, http://www.ietf.org/rfc/rfc2911.txt

1.3 Protocol Overview (Synopsis)

The Web Point-and-Print Protocol provides a mechanism for clients to download printer driver software either from a server in the client network or from a Web site, or directly by print devices.<1>

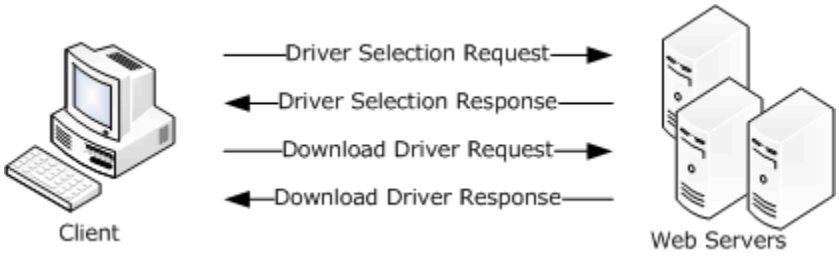


Figure 1: Client selection and download of printer driver

A Web Point-and-Print Protocol Web server maintains a list of printer drivers. A client makes a Driver Selection Request (section 2.3.2) to obtain a printer driver of a particular type and for a particular client configuration. If the server locates a printer driver that matches these requirements, the server redirects the client to the location of the printer driver through the Driver Selection Response (section 2.3.3).

Once a suitable printer driver has been found, the client downloads it by issuing a Download Driver Request (section 2.3.4). The driver is supplied in a Download Driver Response (section 2.3.5).

1.4 Relationship to Other Protocols

The Web Point-and-Print Protocol is dependent on HTTP specifications [RFC2616] and [RFC2617], and on TCP/IP specification [RFC793]. There are no protocols that are dependent on the Web Point-and-Print Protocol.

1.5 Prerequisites/Preconditions

The Web Point-and-Print Protocol is built on HTTP and therefore has the prerequisites specified in [RFC2616] that are common to HTTP communications.

It is a requirement that, before the protocol is invoked, a Web Point-and-Print Protocol client has obtained both the name of a server that supports the Web Point-and-Print Protocol and the name of a printer. How a client obtains these names is not addressed in this specification.

1.6 Applicability Statement

The Web Point-and-Print Protocol is applicable in environments that require distribution of printer driver software where other mechanisms (such as the Print System Remote Protocol; for more information, see [MS-RPRN]) are not available.

1.7 Versioning and Capability Negotiation

The Web Point-and-Print Protocol does not have versioning and does not contain locale-dependent information.

1.8 Vendor-Extensible Fields

The Web Point-and-Print Protocol does not support vendor-extensible fields.

1.9 Standards Assignments

The Web Point-and-Print Protocol uses the following default assignments:

Parameter	Value	Reference
TCP/IP port for Hypertext Transfer Protocol (HTTP)	80	As specified in [RFC2616] section 3.2.2.
TCP/IP port for **Hypertext Transfer Protocol over Secure Socket Layer (HTTPS)**	443	As specified in [RFC2818] section 2.3.

2 Messages

The following sections specify the syntax of Web Point-and-Print Protocol messages and how they are transported.

2.1 Transport

The Web Point-and-Print Protocol uses the following transports for HTTP sequences:

- HTTP over TCP/IP
- HTTPS over TCP/IP

2.2 Common Data Types

Unless otherwise noted, the following statements apply to this specification:

- All strings that are defined in this protocol MUST consist of characters that are encoded in **Unicode UTF-16LE** and MUST be null-terminated. Each code point in a string, including null terminators, MUST occupy 16 bits. For more information, see [RFC2781] section 2.1.
- A list of strings is referred to as a **multisz** in this protocol. In a multisz, the characters that make up the string N+1 MUST directly follow the terminating null character of string N. The last string in a multisz MUST be terminated by two null characters.
- All parameters or members that specify the number of characters in a string or multisz MUST specify the number of characters in the string, including the terminating null characters.
- All constraints that specify the maximum number of characters in a string or multisz MUST specify the number of characters in the string, including the terminating null characters.
- All parameters or members that specify the number of bytes in buffers that contain a string or multisz MUST specify the number of bytes in the buffer, including the terminating null characters.

2.3 Message Syntax

The Web Point-and-Print Protocol issues its requests from the client via **Uniform Resource Locators (URLs)** in an HTTP GET request and the associated query qualifier. The server returns a response via an HTTP response message, which returns either a Location header in a 302 (redirect) response or an HTTP error 500 if the original HTTP GET request cannot be processed by the server. For HTTP, see [RFC2616].

2.3.1 ClientInfo Type

There are four parameters passed in the **ClientInfo** field:

- Major version number of the client operating system.
- Minor version number of the client operating system.
- Value specifying the type of client platform.
- Value specifying the client processor architecture.

Each parameter is an 8-bit unsigned integer; possible values for parameters are shown in the tables that follow.

For transmission on the wire, a single unsigned 32-bit value called **ClientInfo** MUST be computed from these four values, according to the following equation:

```
ClientInfo = (Major Version) * (2^24) + (Minor Version) * (2^16) +
    (Client Platform) * (2^8) + (Client Processor Architecture)
```

The major version number of the client operating system SHOULD be one of the following:

Value	Major version
0x00000003	Windows NT 3.51
0x00000004	Windows 95, Windows 98, Windows Me, or Windows NT 4.0
0x00000005	Windows 2000 Server, Windows XP, or Windows Server 2003
0x00000006	Windows Vista

The minor version number of the client operating system SHOULD be one of the following:

Value	Minor version
0x00000000	Windows 95, Windows NT 4.0, Windows 2000 Server, or Windows Vista
0x00000001	Windows XP
0x00000002	Windows Server 2003
0x0000000A	Windows 98
0x00000033	Windows NT 3.51
0x0000005A	Windows Me

The client platform value SHOULD be one of the following:

Value	Client platform
0x00000000	Windows NT x86
0x00000001	Windows IA64
0x00000002	Windows x64

The client processor architecture value MUST be one of the following:

Value	Client processor architecture
0x00000000	Intel
0x00000006	IA64

Value	Client processor architecture
0x00000009	AMD64

2.3.2 Driver Selection Request

The Driver Selection Request sends the printer driver selection criteria to the server. The message is a standard HTTP GET request, as specified in [RFC2616]. The format of the HTTP request URL MUST be as specified by the grammar rule for **PrinterDriverQuery**:

```
PrinterDriverQuery = [PRINT_SERVER] PRINTER_RESOURCE "?" DRIVER_QUERY
PRINT_SERVER       = "http:" "//" host [":" port]
PRINTER_RESOURCE   = abs_path
DRIVER_QUERY       = "createexe" "&" CLIENT_INFO
CLIENT_INFO        = 1*DIGIT
```

For the definitions of **host**, **port**, **abs_path**, and **DIGIT**, see [RFC2616] section 2.2 and [RFC2396] sections 3.2.2 and 3.3. How the client determines the **host**, **port**, and **abs_path** is not specified by this protocol. Typically, the port is either the HTTP default port 80 or another port that is configured by an administrator.

DRIVER_QUERY: Forms the HTTP query component, as specified in [RFC2396] section 3.4, which constrains the selection of printer driver instances for the printer resource.

PRINT_SERVER: MUST specify the address of the Web Point-and-Print Protocol server. This value MUST be specified if the request is to be routed through a **proxy**.

PRINTER_RESOURCE: MUST specify the name of the printer resource.

CLIENT_INFO: MUST contain the environmental information on the client that is making the request, constructed as a **ClientInfo** value, and formatted as the **ASCII** representation of a decimal integer; and it MUST be computed as specified in section 2.3.1.

2.3.3 Driver Selection Response

The Driver Selection Response packet is a standard HTTP redirect response, in the HTTP **Location** field, to the URL of a file that contains the printer driver software and the information necessary to install the printer driver on the client. Specifically, the response MUST be a **Redirection 302** response, as defined in [RFC2616] section 10.3.3, and the HTTP **Location** field MUST point to the printer driver file.

2.3.4 Driver Download Request

This MUST be a standard HTTP GET request to the URL specified in the Location header received in the Driver Selection Response. For more information, see [RFC2616] section 10.3.3.

2.3.5 Driver Download Response

This MUST be a standard HTTP response, as specified in [RFC2616]. It MUST be returned in response to the driver download HTTP GET request. If the response return code is 200 (success), the response entity body MUST contain the requested driver.

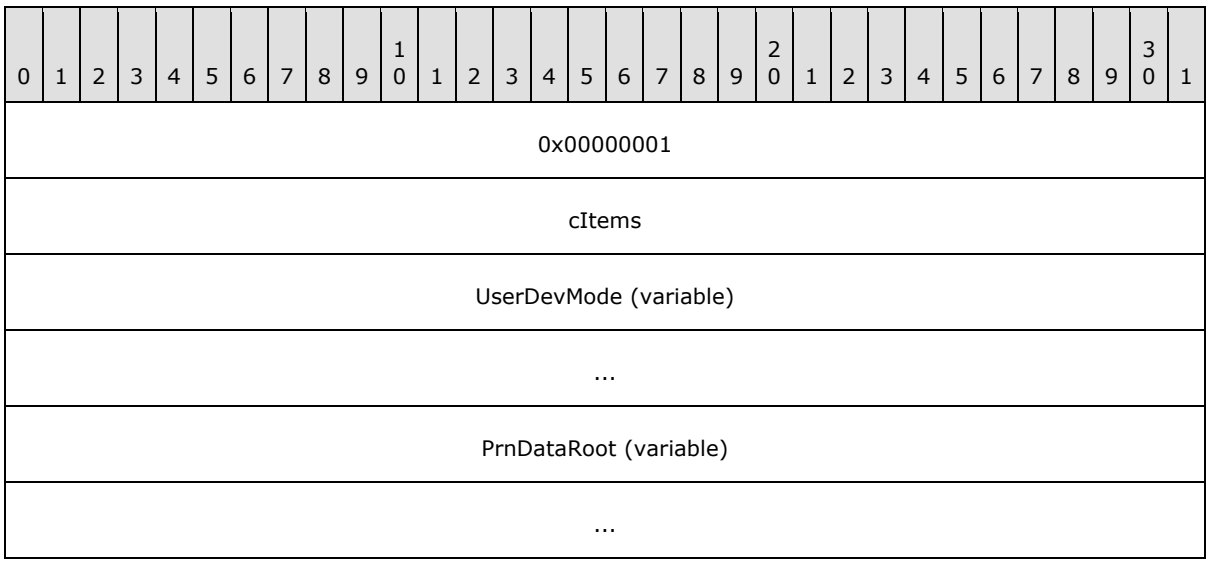
The HTTP redirect is to a URL with content-type set to application/octet-stream, which specifies a **cabinet file** (for more information, see [MSDN-CAB]) that MUST have a .webpnp extension. The cabinet file includes the following:

- Printer driver software files.
- An **INF file** that is used during installation of the driver software. For more information, see [MSDN-INF].
- A BIN file that provides the information needed to access printer setup/configuration data. For the structure of this file, see section 2.3.5.1.
- A DAT file that provides additional **print server** and printer setup/configuration data. For the structure of this file, see section 2.3.5.2.

2.3.5.1 BIN File Format

A BIN file contains the information needed to access printer setup and configuration data. The format of that information is shown below. All offsets are **little-endian** format.

File Header:



cItems (4 bytes): The number of PrnDataRoot structures contained in this structure. This value MAY be zero.

UserDevMode (variable): A UserDevMode (section 2.3.5.1.1) structure.

PrnDataRoot (variable): A PrnDataRoot (section 2.3.5.1.2) structure, which is repeated **cItems** times in this field.

2.3.5.1.1 UserDevMode Structure

A structure that is used for general printer information.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
cbSize																															
0x00000000																															
...																															
...																															
pDataOffset																															
cbData																															
Data (variable)																															
...																															

cbSize (4 bytes): Total number of bytes in this UserDevMode structure.

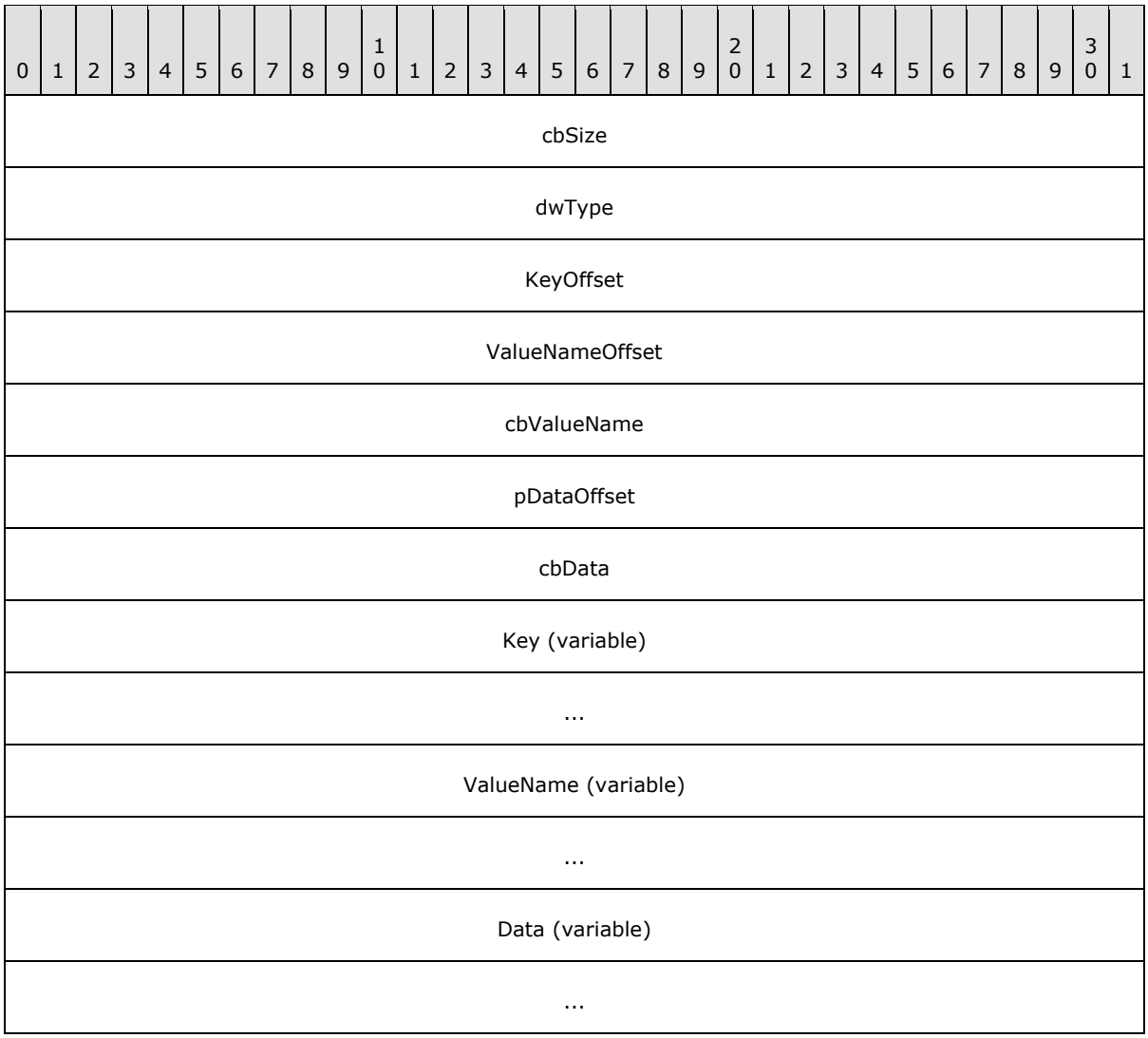
pDataOffset (4 bytes): A 32-bit unsigned integer that MUST specify the number of bytes from the start of the UserDevMode structure to the **Data** field.

cbData (4 bytes): Number of bytes in the **Data** field.

Data (variable): A **DEVMODE** structure; for more information, see [MS-RPRN] section 2.2.2.1. This field MUST be padded to the next 64-bit boundary.

2.3.5.1.2 PrnDataRoot Structure

A structure that is repeated for a count specified by the **cItems** member of the BIN file (section 2.3.5.1) header. PrnDataRoot describes **registry** structures related to the printer.



cbSize (4 bytes): Total number of bytes in this PrnDataRoot structure.

dwType (4 bytes): The value is a registry value type. For more information about this value, see [MSDN-RTV].

KeyOffset (4 bytes): A 32-bit unsigned integer that MUST specify the number of bytes from the start of the PrnDataRoot structure to the **Key** field.

ValueNameOffset (4 bytes): A 32-bit unsigned integer that MUST specify the number of bytes from the start of the PrnDataRoot structure to the **ValueName** field.

cbValueName (4 bytes): Number of bytes in **ValueName**, including the terminating NULL character.

pDataOffset (4 bytes): A 32-bit unsigned integer that MUST specify the number of bytes from the start of the PrnDataRoot structure to the **Data** field.

cbData (4 bytes): The number of bytes in the **Data** field.

Key (variable): A string that MUST contain the name of the key that corresponds to the printer configuration value. This field MUST be padded to the next 64-bit boundary.

ValueName (variable): A string that MUST contain the name of the printer configuration value. This field MUST be padded to the next 64-bit boundary.

Data (variable): A field containing data for the printer configuration value identified by **ValueName**. The registry value type for **Data** MUST be **dwType**. This field MUST be padded to the next 64-bit boundary.

2.3.5.2 DAT File Format

A DAT file contains the following options used for installing the printer driver on the client machine.

```
/if
/x
/Q<PackageList>
/b<ServerName>
/f<InfName>
/r<PrinterName>
/m<DriverName>
/n<UncName>
/a<BinName>
/q
```

PackageList: List of **CAB file** names containing **driver packages** included in the **.webpnp** file. The individual CAB file names are separated by ";".

ServerName: URL for the print server.

InfName: Name of the INF file included in the **.webpnp** file.

PrinterName: URL of the printer.

DriverName: Name of the printer driver.

UncName: **Universal Naming Convention (UNC)** path of the server.

BinName: Name of the BIN file included in the **.webpnp** file.

The following syntactic restrictions apply to these options:

- The name of the DAT file file MUST be "cab_ipp.dat".
- The individual options MAY appear in any order.
- The individual options MUST be separated by white space.
- White space MAY be present between the switch of an option (e.g. "/r") and the parameter.
- A parameter MAY be surrounded by quotes if it does not contain white space.
- The parameter MUST be surrounded by quotes if it contains any white space.
- Exactly one of the options "/x" and "/Q" MUST be specified.
- "/Q" MUST NOT be specified if **MajorVersion** of the **CLIENT_INFO** of the request is less than 6.

- If "/Q" is specified, "/q" MUST NOT be specified.

3 Protocol Details

The following sections specify details of the Web Point-and-Print Protocol, including abstract data models and message processing rules.

3.1 Client Details

3.1.1 Abstract Data Model

No abstract data model is required.

3.1.2 Timers

There are no timers required for the Web Point-and-Print Protocol.

3.1.3 Initialization

There is no protocol-specific initialization.

3.1.4 Higher-Layer Triggered Events

A client invocation is the result of local client application activity. The client application specifies the driver selection parameters. The syntax and semantics of these parameters are specified in section 2.3.2. No other higher-layer triggered events are defined.

3.1.5 Message Processing Events and Sequencing Rules

As specified in section 1.5, the client MUST be preconfigured with the URL of a Web Point-and-Print Protocol server. To initiate the Web Point-and-Print Protocol, the client MUST send a Driver Selection Request (section 2.3.2) message to the Web server with which it has been configured.

If the server returns an HTTP status other than 302 (found), the client MUST NOT process any Location header in the HTTP response, and MUST return an error code to the client application.

If the server returns an HTTP redirect (indicated by an HTTP status code of 302), the client MUST parse the received HTTP response to retrieve the Location header. If a Location header is not found in the response, the client MUST return an error to the invoking application. Otherwise, the client MUST send a Driver Download Request (section 2.3.4) to the server identified in the Location header of the received HTTP response.

If the HTTP response returned to the client has a status of 200 (success), the entity body received in the HTTP response MUST be returned to the invoking application as the printer driver package. If the HTTP response is anything other than success, the client MUST return an error to the invoking client application.

3.1.6 Timer Events

There are no timer events.

3.1.7 Other Local Events

No additional local events are used on the client beyond the events maintained in the underlying HTTP.

3.2 Server Details

3.2.1 Abstract Data Model

This section describes a conceptual model of a possible data organization that an implementation might need to maintain to participate in this protocol. The organization described in this section is provided to facilitate the explanation of how the protocol behaves. This document does not mandate that implementations adhere to this model as long as their external behavior is consistent with that described in this document.

The Web Point-and-Print Protocol depends on an abstract data model that maintains printers. The print server MUST behave as if it hosted the following objects in the specified hierarchy.

List of Printers: Each printer represents a physical print device or a number of homogeneous physical devices installed on the print server. Each printer object MUST maintain the following data elements:

- A name that uniquely identifies the printer.
- A reference to a printer driver object for the printer.
- Per-user DEVMODE settings (for more information, see [MS-RPRN] section 2.2.2.1).

List of Drivers: Each printer driver represents the software component responsible for converting print content submitted by applications into device-specific commands. Each printer driver object MUST maintain the following data elements:

- A name that uniquely identifies the printer driver.
- A list of well-known modules (that is, rendering module, configuration module, and data module).
- Additionally, each printer driver object SHOULD maintain the following optional data elements:
  - A list of dependent files.
  - Information on the printer driver manufacturer, printer driver time stamp, and version.

The abstract data model MUST be able to relate each printer to a single printer driver. For more information, see [MS-RPRN] section 3.1.1.

3.2.2 Timers

There are no timers required for the Web Point-and-Print Protocol.

3.2.3 Initialization

The server MUST listen for HTTP and HTTPS requests.

3.2.4 Higher-Layer Triggered Events

No higher-layer triggered events are used.

3.2.5 Message Processing Events and Sequencing Rules

The client initiates the Web Point-and-Print Protocol by sending a Driver Selection Request (section 2.3.2) message to the server. On receiving this message, the server MUST validate parameters as follows:

- **PRINTER_RESOURCE** parameter MUST be a valid absolute path of a printer resource.
- **CLIENT_INFO** section of the URL MUST be a ClientInfo type.

If parameter validation fails, the server MUST fail the operation immediately, returning HTTP error 500 (Internal Server Error). If parameter validation succeeds, the server MUST compose a response to the client as follows:

- The server MUST use the received **CLIENT_INFO** parameter to determine the location of the appropriate printer driver. The mechanism is implementation specific. If a matching driver is found, the server MUST create a Driver Selection Response, replying to the client with an HTTP 302 redirect.<2>
- If a matching driver cannot be found, the server MUST fail the operation, replying to the client with HTTP error 500 (Internal Server Error).

Processing of a Driver Download Request MUST follow the rules specified for GET processing, as specified in [RFC2616] section 9.3. The format of the data returned in the response is specified in section 2.3.5.

3.2.6 Timer Events

There are no timer events.

3.2.7 Other Local Events

No additional local events are used on the client beyond the events maintained in the underlying HTTP.

4 Protocol Examples

The following sections describe operations as used in common scenarios to illustrate the function of the Web Point-and-Print Protocol.

4.1 Client/Server Interaction

The following diagram illustrates the interaction sequence to obtain a printer driver using the Web Point-and-Print Protocol.

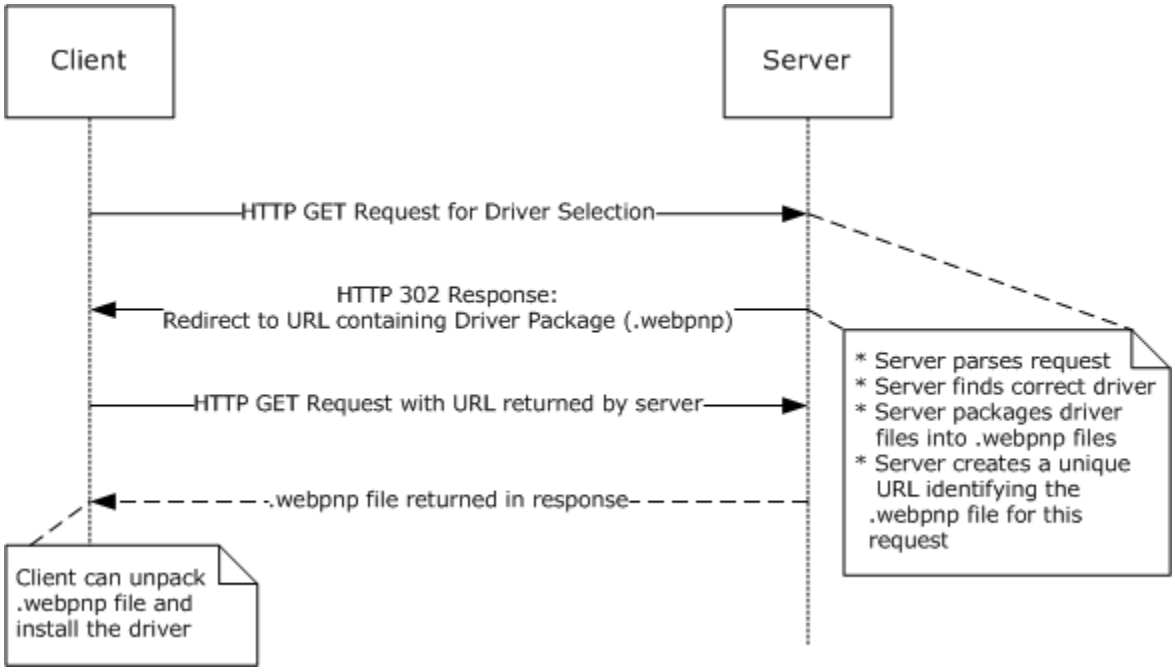


Figure 2: Obtaining a printer driver

4.2 Client/Server Messages

4.2.1 Client

The following text is a sample client Driver Selection Request, depicted in the figure of section 4.1. An HTTP **GET** request is used for sending printer driver selection criteria to the server. For more information, see section 2.3.2.

```
GET /printers/printerModelXXX/.printer?createexe&83952128
```

4.2.2 Server

The following text is a sample server Driver Selection Response to a client Driver Selection Request, as depicted in the figure of section 4.1. The server performs an HTTP **Redirection 302** response to the URL of a file that contains the requested printer driver and installation software. For more information, see section 2.3.3.

```
HTTP/1.1 302 Object Moved
Content-Length: 176
Content-Type: text/html
Location: http://1.1.1.2/printers/PrtCabs/85CCX862.webpnp
Server: Microsoft-IIS/6.0
X-Powered-By: ASP.NET
Date: Wed, 12 Jul 2006 21:44:42 GMT
Connection: close
<head><title>Document Moved</title></head>
<body><h1>Object Moved</h1>This document may be found
   <a HREF="http://1.1.1.2/printers/PrtCabs/85CCX862.webpnp">here</a>
</body>
<END OF ENTITY>
```

5 Security

The following sections specify security considerations for implementers of the Web Point-and-Print Protocol.

5.1 Security Considerations for Implementers

The Web Point-and-Print Protocol does not provide any facilities for mutual authentication, so server identities cannot be verified.

5.2 Index of Security Parameters

There are no security parameters in the Web Point-and-Print Protocol.

6 Appendix A: Windows Behavior

The information in this specification is applicable to the following versions of Windows:

- Windows 2000 Server
- Windows XP
- Windows Server 2003
- Windows Vista
- Windows Server 2008

Exceptions, if any, are noted below. Unless otherwise specified, any statement of optional behavior in this specification that is prescribed by using the terms SHOULD or SHOULD NOT implies Windows behavior in accord with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term MAY implies that Windows does not follow the prescription.

<1> Section 1.3: The Web Point-and-Print Protocol is used when the Print System Remote Protocol (for more information, see [MS-RPRN]) is not available due to policy restrictions, policy restrictions on the use of RPC, or other restrictions on printing. When the Web Point-and-Print Protocol is used to install the printer drivers, the print spooler uses the **Internet Printing Protocol (IPP)** (for more information, see [RFC2910] and [RFC2911]) to communicate with the actual print device.

<2> Section 3.2.5: The Windows implementation extracts information from printer driver INF files to determine the appropriate drivers to return to the client. For information on printer driver INF files, see [MSDN-PRNINF].

7 Index

S

T

U

V

W