r

HTTP Proxy Service (HPS)

Bluetooth® Service Specification

- **Date** 2015-Oct-06
- **Revision** v1.0
- **Group Prepared By** Internet Working Group
- **Feedback Email** int-main@bluetooth.org

Abstract:

This HTTP Proxy Service (HPS) Specification introduces a service that allows a Client device, typically a sensor, to communicate with a Web Server through a gateway device. The gateway device implements the HTTP Proxy Service and therefore provides the services available through the Internet to the Client sensor device.

Revision History

Revision Number	Date	Comments
v1.0	10/06/2015	Adopted by the Bluetooth SIG BoD

Contributors

Name	Company
Robin Heydon	CSR
Kanji Kerai	Nokia
Magnus Olsson	Ericsson
Marcel Holtmann	Intel
Mats Andersson	Connect Blue
Niclas Granqvist	Polar
Frank Berntsen	Nordic Semiconductor
Chris Hansen	Intel
Victor Zhodzishsky	Broadcom
Scott Walsh	Plantronics
Joe Decuir	CSR
Krishna Shingala	Nordic Semiconductor
Julien Gros	Dialog Semiconductor
YoungHwan Kwon	LG Electronics
Masaya Masuda	Toshiba

DISCLAIMER AND COPYRIGHT NOTICE

This disclaimer applies to all draft specifications and final specifications adopted by the Bluetooth SIG Board of Directors (both of which are hereinafter referred to herein as a Bluetooth "Specification"). Your use of this Specification in any way is subject to your compliance with all conditions of such use, and your acceptance of all disclaimers and limitations as to such use, contained in this Specification. Any user of this Specification is advised to seek appropriate legal, engineering or other professional advice regarding the use, interpretation or effect of this Specification on any matters discussed in this Specification.

Use of Bluetooth Specifications and any related intellectual property is governed by the Promoters Membership Agreement among the Promoter Members and Bluetooth SIG (the "Promoters Agreement"), certain membership agreements between Bluetooth SIG and its Adopter and Associate Members, including, but not limited to, the Membership Application, the Bluetooth Patent/Copyright License Agreement and the Bluetooth Trademark License Agreement (collectively, the "Membership Agreements") and the Bluetooth Specification Early Adopters Agreements (1.2 Early Adopters Agreements) among Early Adopter members of the unincorporated Bluetooth SIG and the Promoter Members (the "Early Adopters Agreement"). Certain rights and obligations of the Promoter Members under the Early Adopters Agreements have been assigned to Bluetooth SIG by the Promoter Members.

Use of the Specification by anyone who is not a member of Bluetooth SIG or a party to an Early Adopters Agreement (each such person or party, a "Member") is prohibited. The use of any portion of a Bluetooth Specification may involve the use of intellectual property rights ("IPR"), including pending or issued patents, or copyrights or other rights. Bluetooth SIG has made no search or investigation for such rights and disclaims any undertaking or duty to do so. The legal rights and obligations of each Member are governed by the applicable Membership Agreements, Early Adopters Agreement or Promoters Agreement. No license, express or implied, by estoppel or otherwise, to any intellectual property rights are granted herein.

Any use of the Specification not in compliance with the terms of the applicable Membership Agreements, Early Adopters Agreement or Promoters Agreement is prohibited and any such prohibited use may result in (i) termination of the applicable Membership Agreements or Early Adopters Agreement and (ii) liability claims by Bluetooth SIG or any of its Members for patent, copyright and/or trademark infringement claims permitted by the applicable agreement or by applicable law.

THE SPECIFICATION IS PROVIDED "AS IS" WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NONINFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, SATISFACTORY QUALITY, OR REASONABLE SKILL OR CARE, OR ANY WARRANTY ARISING OUT OF ANY COURSE OF DEALING, USAGE, TRADE PRACTICE, PROPOSAL, SPECIFICATION OR SAMPLE.

Each Member hereby acknowledges that products equipped with the Bluetooth wireless technology ("Bluetooth Products") may be subject to various regulatory controls under the laws and regulations applicable to products using wireless non licensed spectrum of various governments worldwide. Such laws and regulatory controls may govern, among other things, the combination, operation, use, implementation and distribution of Bluetooth Products. Examples of such laws and regulatory controls include, but are not limited to, airline regulatory controls, telecommunications regulations, technology transfer controls and health and safety regulations. Each Member is solely responsible for the compliance by their Bluetooth Products with any such laws and regulations and for obtaining any and all required authorizations, permits, or licenses for their Bluetooth Products related to such regulations within the applicable jurisdictions. Each Member acknowledges that nothing in the Specification provides any information or assistance in connection with securing such compliance, authorizations or licenses. NOTHING IN THE SPECIFICATION CREATES ANY WARRANTIES, EITHER EXPRESS OR IMPLIED, REGARDING SUCH LAWS OR REGULATIONS.

ALL LIABILITY, INCLUDING LIABILITY FOR INFRINGEMENT OF ANY INTELLECTUAL PROPERTY RIGHTS OR FOR NONCOMPLIANCE WITH LAWS, RELATING TO USE OF THE SPECIFICATION IS EXPRESSLY DISCLAIMED. To the extent not prohibited by law, in no event will Bluetooth SIG or its Members or their affiliates be liable for any damages, including without limitation, lost revenue, profits, data or programs, or business interruption, or for special, indirect, consequential, incidental or punitive damages, however caused and regardless of the theory of liability, arising out of or related to any furnishing, practicing, modifying, use or the performance or implementation of the contents of this Specification, even if Bluetooth SIG or its Members or their affiliates have been advised of the possibility of such damages. BY USE OF THE SPECIFICATION, EACH MEMBER EXPRESSLY WAIVES ANY CLAIM AGAINST BLUETOOTH SIG AND ITS MEMBERS OR THEIR AFFILATES RELATED TO USE OF THE SPECIFICATION.

If this Specification is an intermediate draft, it is for comment only. No products should be designed based on it except solely to verify the prototyping specification at SIG sponsored IOP events and it does not represent any commitment to release or implement any portion of the intermediate draft, which may be withdrawn, modified, or replaced at any time in the adopted Specification.

Bluetooth SIG reserves the right to adopt any changes or alterations to the Specification it deems necessary or appropriate.

Document Terminology

The Bluetooth SIG has adopted portions of the IEEE Standards Style Manual which dictate use of the words “shall’’, “should”, “may”, and “can” in the development of documentation, as follows:

The word *shall* is used to indicate mandatory requirements strictly to be followed in order to conform to the standard and from which no deviation is permitted (*shall* equals *is required to*).

The use of the word *must* is deprecated and shall not be used when stating mandatory requirements; *must* is used only to describe unavoidable situations.

The use of the word *will* is deprecated and shall not be used when stating mandatory requirements; *will* is only used in statements of fact.

The word *should* is used to indicate that among several possibilities one is recommended as particularly suitable, without mentioning or excluding others; or that a certain course of action is preferred but not necessarily required; or that (in the negative form) a certain course of action is deprecated but not prohibited (*should* equals *is recommended that*).

The word *may* is used to indicate a course of action permissible within the limits of the standard (*may* equals *is permitted*).

The word *can* is used for statements of possibility and capability, whether material, physical, or causal (*can* equals *is able to*).

The term *Reserved for Future Use (RFU)* is used to indicate Bluetooth SIG assigned values that are reserved by the Bluetooth SIG and are not otherwise available for use by implementations.

Contents

v1.0

1 Introduction

The HTTP Proxy Service (HPS) allows a device to expose HTTP Web services to a client of the HPS Server (HPS Client) using the Generic Attribute Profile (GATT). This enables an HPS Client device to program a set of characteristics that configures a Hyper Text Transfer Protocol (HTTP) request, initiate this request, and then read the response.

The HPS is designed to support HPS Client devices that are constrained in computing resources. The HPS Client software that uses HPS to reach remote HTTP Servers will be limited in the sizes of HTTP headers and entity bodies that they can generate and consume. From a practical standpoint, the HPS specification is intended for use by applications that will need relatively small headers and entity bodies, which can in turn fit within the GATT maximum characteristics size limits.

The device implementing the HPS server shall support the GATT Server role. As per Figure 1.1, the HPS Client would support the GATT Client role.

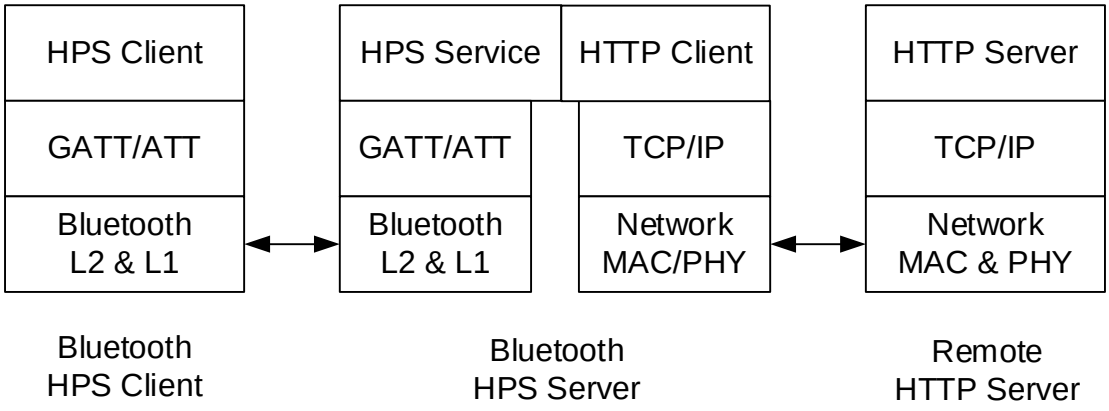

Figure 1.1: *HTTP Proxy Service model*

1.1 Conformance

If conformance to this service is claimed, all capabilities indicated as mandatory for this service shall be supported in the specified manner (process-mandatory). This also applies for all optional and conditional capabilities for which support is indicated. All mandatory capabilities, and optional and conditional capabilities for which support is indicated, are subject to verification as part of the Bluetooth qualification program.

1.2 Service Dependencies

This service is not dependent upon any other services.

1.3 Bluetooth Specification Release Compatibility

This specification is compatible with the Bluetooth Core Specification Version 4.0 or later [1].

1.4 GATT Sub-Procedure Requirements

Requirements in this section represent a minimum set of requirements for a Server.

Table 1.1 summarizes additional GATT sub-procedure requirements beyond those required by all GATT Servers.

GATT Sub-Procedure	Requirements
Read Long Characteristic Values	M
Write Characteristic Value	M
Write Long Characteristic Values	M
Notifications	M
Read Characteristic Descriptors	M
Write Characteristic Descriptors	M

Table 1.1: *GATT Sub-procedure requirements*

1.5 Transport Dependencies

There are no transport restrictions imposed by this service specification.

Where the term BR/EDR is used throughout this document, this also includes the use of AMP.

1.6 Error Codes

This service defines the following Attribute Protocol (ATT) Application error code:

Name	Error Code	Description
Invalid Request	0x81	A HTTP Control Point request cannot be serviced because content of the URI, the HTTP Headers or the HTTP Entity Body characteristics is not set correctly
Network Not Available	0x82	Network connection not available

Table 1.2: *ATT Application Error Code defined by this service.*

The HPS Server may reject the HPS Client's Write request to the URI (Universal Resource Identifier), HTTP Header, HTTP Entity Body, or HTTP Control Point with "Insufficient Resources" result code if it cannot originate an HTTP request to the HTTP Server.

1.7 Byte Transmission Order

All characteristics used with this service shall be transmitted with the least significant octet first (i.e., little endian). The least significant octet is identified in the characteristic definitions in [2].

2 Service Declaration

The HPS should be instantiated as a «Primary Service». There shall be only one instance of the HPS on the device per Client. The device may support multiple simultaneous connections originated from multiple HPS Clients. If multiple simultaneous connections are supported, operations defined in this specifications originated from one HPS Client shall not affect operations from another HPS Client.

The service UUID shall be set to «HTTP Proxy Service» defined in [2].

3 Service Characteristics

The following characteristics are exposed in the HTTP Proxy Service. Only one instance of each characteristic is permitted within this service. The characteristic formats and UUIDs are defined in [2].

Where a characteristic can be notified, a Client Characteristic Configuration Descriptor shall be included in that characteristic as required by the Core Specification [1].

Characteristic Name	Requirement	Mandatory Properties	Optional Properties	Security Permissions
URI	M	Write, Write Long		Optional
HTTP Headers	M	Read, Write, Read Long, Write Long		Optional
HTTP Entity Body	M	Read, Write, Read Long, Write Long		Optional
HTTP Control Point	M	Write		Optional
HTTP Status Code	M	Notify		Optional
HTTPS Security	M	Read		Optional

Table 3.1: *HPS Service Characteristic Requirements*

Notes:

- Security Permissions of “Optional” means that this service does not impose any requirements on security; the service can either require security or no security.

3.1 URI Characteristic

The URI characteristic is used to configure the URI for a subsequent request.

The URI characteristic is identified using the UUID «URI», as defined in [2].

The URI characteristic shall contain the URI to be used in the HTTP request as a UTF-8 string (utf8s), up to the maximum characteristic length defined.

This version of the specification supports HTTP requests with URI up to 512 octets in length.

3.1.1 Characteristic Behavior

The URI characteristic can be written while an HTTP request is not being executed. The behavior when this characteristic is written while an HTTP request is being executed is not defined.

3.2 HTTP Headers Characteristic

The HTTP Headers characteristic is used to hold the headers that would be sent with the HTTP Request or the headers contained within an HTTP response message from the HTTP Server.

The HTTP Header characteristic is identified using the UUID «HTTP Header», as defined in [2].

The HTTP Header characteristic shall contain the HTTP message headers to be used in the HTTP requests and responses as defined in [3], up to the maximum characteristic length defined.

This version of the specification supports HTTP requests with HTTP headers up to 512 octets in length.

3.2.1 Characteristic Behavior

The HTTP Headers characteristic can be read or written while an HTTP request is not being executed. The behavior when this characteristic is read or written while an HTTP request is being executed is not defined.

When written, the HTTP Headers shall be set to the general-header, request-header, and entity-header fields of an HTTP request.

When read, the HTTP Headers shall be set to the general-header, request-header and entity-header fields of an HTTP response; see Section 3.4.1.

The HPS Server shall notify the HPS Client that the HTTP Headers characteristic has been filled with data received from the HTTP Server using the Data Status field sent in the HTTP Status Code notification; see Section 3.5.

If HTTP Headers received by the HPS Server in the HTTP response is more than 512 octets in length, the HPS Server shall store the first 512 octets of the received header in the HTTP Headers characteristic. The fact that the data field has been truncated shall be reflected in the Data Status field sent in the HTTP Status Code notification; see Section 3.5.

Note: The Server may add additional headers as it requires.

Note: The management of cookies should not be performed in the Server of the HTTP Proxy Service.

3.3 HTTP Entity Body Characteristic

The HTTP Entity Body characteristic contains the contents of the message-body after any Transfer-Encoding has been applied.

The HTTP Entity Body characteristic is identified using the UUID «HTTP Entity Body», as defined in [2].

The HTTP Entity Body characteristic shall contain the HTTP message-body used in the HTTP requests and responses as defined in [3], up to the maximum characteristic length defined.

For HTTP requests that do not send an HTTP Entity Body, the HPS Server shall expect client to initialize it with a zero-length value.

This version of the specification supports HTTP requests with HTTP Entity Body up to 512 octets in length.

3.3.1 Behavior Description

This characteristic should not be written while an HTTP request is being executed, see section 3.4.1. The behavior of this characteristic, if written when an HTTP request is being executed, is not defined.

The HTTP Entity Body is the message-body of the HTTP request or the HTTP response.

The HPS Server shall notify the HPS Client that the HTTP Entity Body characteristic has been filled with data received from the HTTP Server using the Data Status field sent in the HTTP Status Code notification; see Section 3.5.

The value of the HTTP Entity Body characteristic when written by the HPS Client shall be sent out as an HTTP message-body. An HTTP message-body received from an HTTP Server shall be stored by the HPS Server in the HTTP Entity Body characteristic.

If the HTTP message-body received by the HPS Server in the HTTP response is more than 512 in length, the HPS Server shall store first 512 bytes of the response in the HTTP Entity Body characteristic. The fact that the data field has been truncated shall be reflected in the Data Status field sent in the HTTP Status Code notification; see Section 3.5.

Note: The Server may transform the entity-body into a message-body by applying a Transfer-Encoding. This may be implicitly performed by the Server to ensure safe and proper transfer of the message.

3.4 HTTP Control Point Characteristic

The HTTP Control Point is used to initiate a request to send an HTTP request message from the device containing the HTTP Proxy Service, acting as an HTTP Client.

The HTTP Control Point characteristic is identified using the UUID «HTTP Control Point», as defined in [2].

The HTTP Control Point characteristic is one octet long (unit8) and identifies the HTTP request type. The HTTP Control Point supports the following commands:

Op Code Value	Procedure
0x00	Reserved for Future Use

0x01	HTTP GET Request
0x02	HTTP HEAD Request
0x03	HTTP POST Request
0x04	HTTP PUT Request
0x05	HTTP DELETE Request
0x06	HTTPS GET Request
0x07	HTTPS HEAD Request
0x08	HTTPS POST Request
0x09	HTTPS PUT Request
0x0a	HTTPS DELETE Request
0x0b	HTTP Request Cancel
0x0c-0xff	Reserved for Future Use

Table 3.2: *HTTP Control Point supported commands.*

Note: Support for the OPTIONS, TRACE, and CONNECT methods is not supported by the HTTP Control Point.

3.4.1 Behavioral Description

The HTTP Control Point is used by an HPS Client to initiate an HTTP or HTTPS request as defined by [3], or to cancel an executing HTTP request.

When an HPS Client uses the Write Characteristic Value GATT sub-procedure with the HTTP GET Request, HTTP HEAD Request, HTTP POST Request, HTTP PUT Request, or HTTP DELETE Request opcode, then the HPS Server shall initiate an HTTP request, as defined by [3], and be considered to be executing the HTTP request. When an HPS Client uses the Write Characteristic Value GATT sub-procedure with the HTTPS GET Request, HTTPS HEAD Request, HTTPS POST Request, HTTPS PUT Request, or HTTPS DELETE Request opcode, then the Server shall initiate an HTTP request, as defined by [3] and [4], and be considered to be executing the HTTP request.

Only a single HTTP Request can be initiated by an HPS Client at the same time; only after a request has completed, can another request be initiated. The completion of an HTTP request is determined by the notification of the HTTP Status Code characteristic.

If the HPS was already executing an outstanding HTTP request when this sub-procedure is used, then the HPS Server shall reject the new HTTP Control Point write request with the Procedure Already In Progress error code [5]. The HPS Server shall continue processing the outstanding HTTP request.

Exception: when an HPS Client uses the Write Characteristic Value GATT sub-procedure with the HTTP Request Cancel opcode, then the HPS Server shall terminate any executing HTTP request from this HPS Client.

If the HPS Server cannot initiate HTTP Request due to network connection not being available, it should reject the Write Request with result code Network Not Available.

The HPS Client must configure HTTP Status Code characteristics before issuing any HTTP request. The HPS Server shall reject an attempt to Write Characteristic Value of the HTTP Control Point Characteristic with error code “Client Characteristic Configuration Descriptor Improperly Configured” if the request is received while the Client Characteristic Configuration descriptor of the HTTP Status Code characteristic is not configured for notifications.

The HPS Client must configure URI, HTTP Headers and HTTP Entity Body by using appropriate GATT write procedure each time before it issues any HTTP Request except the HTTP Request Cancel. The HPS Server shall reject an attempt to Write Characteristic Value of the HTTP Control Point Characteristic with error code “Invalid Request” if the request is received before URI, HTTP Headers and HTTP Entity Body characteristics have been configured after startup, or after the previous HTTP Request. If a particular characteristic is not used in executing an HTTP Request, then it shall be configured with a null (zero length) value.

Upon completing execution of an HTTP request, the Server shall notify the HTTP Status Code characteristic with the value of the Status-Code of the Status-Line of the first line of the HTTP Response message (for example 200 OK, or 404 RESOURCE NOT FOUND). The value of the associated HTTP characteristics shall contain the values from the HTTP Response message.

3.4.2 Implicit Behavior

The Server should process the following Status-Code values and continue with the request as directed by [3]:

- 100 – Continue
- 101 – Switching Protocols
- 301 – Moved Permanently
- 302 – Found
- 303 – See Other
- 305 – Use Proxy
- 307 – Temporary Redirect

The HPS Server may impose a timeout on any request sent to the Internet. If this timeout expires, then the Status-Code 504 shall be used to notify that the HTTP request timed-out. The HPS Server can manage connections to the HTTP Server in any way that it wants. The use of persistent connections and message transmission requirements is out of scope of this specification.

Note: This behavior is abstracted away from the behavior of this Service.
The HPS Server can perform caching of HTTP resources as defined in [3] §13.

3.5 HTTP Status Code Characteristic

The HTTP Status Code characteristic contains the Status-Code from the Status-Line of the first line of the HTTP Response message, followed by one octet indicating the Data Status bit field indicating the status of the data received as defined in Table 3.5.

Data Status	Value	Description
Headers Received	0x01	If set, the response-headers and entity-headers field were received in the HTTP response and was stored in the HTTP Headers characteristic for the Client to read
Headers Truncated	0x02	If set, the response-headers and entity-headers field exceeded 512 octets in length and first 512 octets were saved in the HTTP Headers characteristic
Body Received	0x04	If set, the entity-body field was received in the HTTP response and was stored in the HTTP Entity Body characteristic for the Client to read
Body Truncated	0x08	If set, the entity-body field exceeded 512 octets in length and first 512 octets were saved in the HTTP Headers characteristic
Reserved for Future Use	0xF0	Reserved for Future Use

Table 3.3: *HTTP Data Status Bit Field*

The HTTP Status Code characteristic is identified using the UUID «HTTP Status Code», as defined in [2].

The HTTP Status Code characteristic value is 3 octets in length with first two-octets representing the Status Code encoded as uint16 based on the HTTP Status Code [3] followed by 1 octet Data Status bit field.

The structure of the HTTP Status Code characteristic is defined below:

	Status Code	Data Status
Data type	uint16	uint8
Size	2 octets	1 octet
Units	None	None

Table 3.4: *Structure of the HTTP Status Code Characteristic.*

3.5.1 Behavior Description

If the HPS Client configured the Characteristic Client Configuration Descriptor to send notifications, the HPS server shall send the notification when the HTTP request execution, initiated by using the HTTP Control Point behavior, has completed.

3.6 HTTPS Security Characteristic

The HTTPS Security characteristic contains the known authenticity of the HTTP Server certificate for the URI.

The HTTP Security characteristic is identified using the UUID «HTTPS Security», as defined in [2].

3.6.1 Behavior Description

Once an HTTPS GET Request, HTTPS HEAD Request, HTTPS POST Request, HTTPS PUT Request, or HTTPS DELETE Request has been written to the HTTP Control Point, the HPS shall authenticate the HTTP Server certificate for the URI. The state of this authenticity shall be readable by the HPS Client before the HTTP Status Code is notified.

This GATT characteristic is Boolean:

- TRUE (0x01) = HTTP Server certificate for the URI is valid
- FALSE (0x00) = HTTP Server certificate for the URI is not valid

3.7 SDP Record

If this service is exposed over BR/EDR then it shall have the following SDP record.

Item	Definition	Type	Value	Status
Service Class ID Lists				M
Service Class #0		UUID	«HTTP Proxy Service»	M
Protocol Descriptor List				M
Protocol #0		UUID	L2CAP	M
Parameter #0 for Protocol #0		Uint16	PSM = ATT	M
Protocol #1		UUID	ATT	M
Parameter #0 for Protocol #1	GATT Start Handle	Uint16	First handle of this services in the GATT database	M
Parameter #1 for Protocol #1	GATT End Handle	Uint16	Last handle of this services in the GATT database	M
BrowseGroupList			PublicBrowseRoot*	M

Table 3.5: *HPS SDP Record*

* Public Browse Root shall be present; however, other browse UUIDs may also be included in the list.

4 LE Connection Procedure

The HPS Server shall support at least one of the GAP Connectable modes as defined in [1]. In the GAP Connectable Mode the HPS Server should include the Service UUID «HTTP Proxy Service» defined in [2] in the Service UUIDs AD type field of the advertising data.

The HPS Server may allow connections from any HPS Client or from previously bonded HPS Clients only. The HPS Server shall perform one of the GAP Connection Establishment procedures to the HPS Clients. If the HPS Server allows connections from any HPS Client it shall connect to any device which includes the Service UUID «HTTP Proxy Service» defined in [2] in the Service Solicitation UUIDs AD type field of the advertisement data.

The HPS Client can be in the GAP Undirected or Directed Connectable mode. In the GAP Undirected Connectable Mode the HPS Client includes the Service UUID «HTTP Proxy Service» defined in [2] in the Service Solicitation UUIDs AD type field of the advertisement data.

The HPS Client may also perform GAP Connection Establishment procedure as defined in [1].

5 Appendix A: Sample MSCs (Informative)

The GATT exchange represented in the example shown in Figure 5.1 illustrates the sample interaction between an HPS Client and an HPS Server to perform an HTTP transaction.

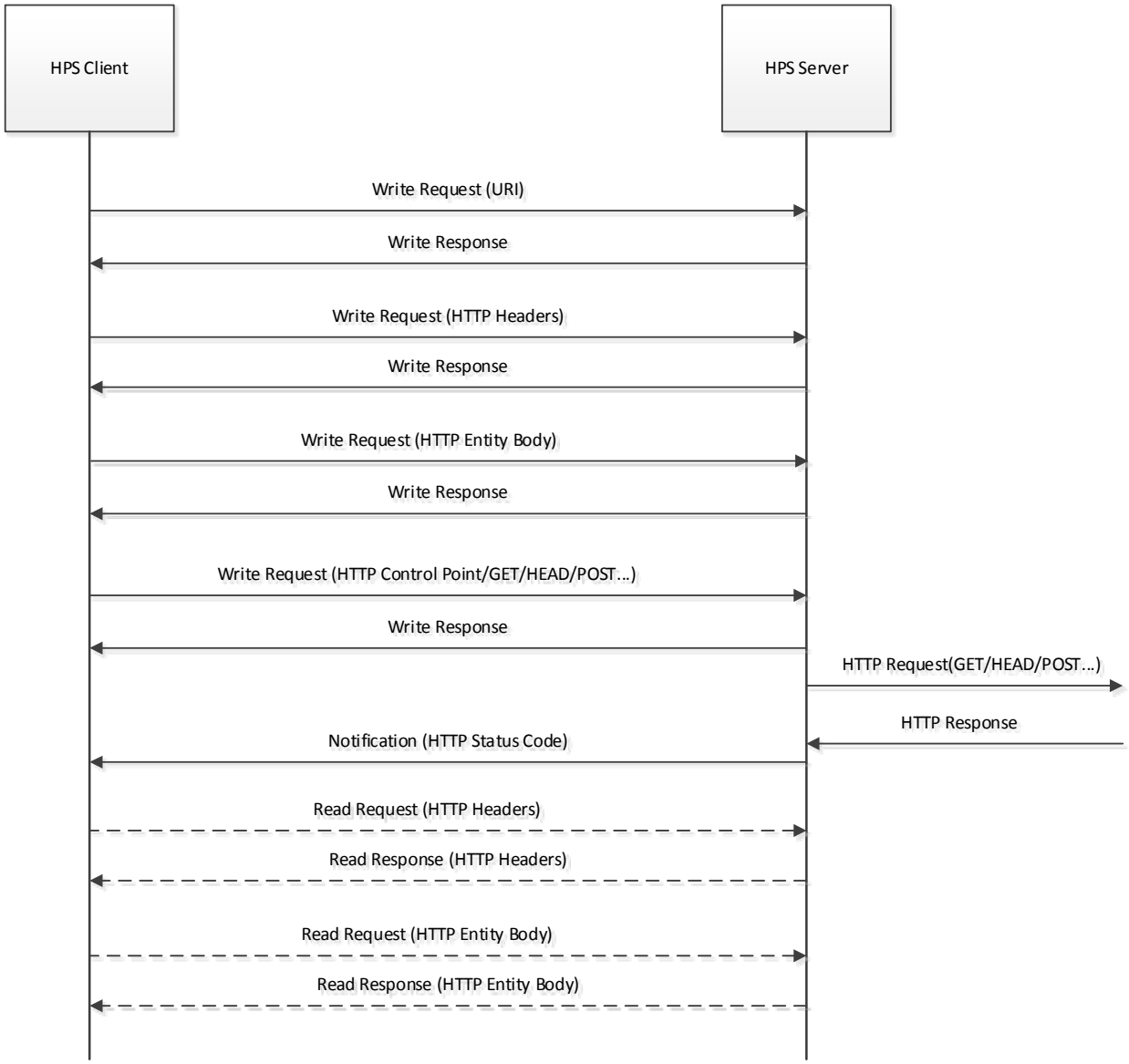

Figure 5.1: *Example MSC for executing an HTTP request*

Note: Dotted lines depict HPS Client requests required for processing of some but not all HTTP requests.

6 Appendix B: Using HTTP for RFC-2616 and RFC-2818 (Informative)

The following sections provide mapping between HPS characteristics and HTTP primitives when information is exchanged between an HPS and an HTTP Client implementing RFC-2616 [3], and to control use of a channel security entity defined in RFC-2818 [4].

6.1 URI - Universal Resource Identifier

The HTTP Client should use the contents of the URI characteristic (see Section 3.1) to construct the HTTP requests.

6.2 HTTP Headers

The HTTP Client should use the contents of the HTTP Headers characteristic (see Section 3.2) to construct the headers portion of HTTP requests. The HTTP Client should use this characteristic to capture incoming HTTP headers from HTTP responses.

6.3 HTTP Entity Body

The HTTP Entity Body characteristic may be used to construct or capture the HTTP Entity Body.

The HTTP Client may write data directly to the HTTP Entity Body characteristic. The HTTP Client should use the contents of the HTTP Entity Body characteristic to construct an HTTP Entity Body for HTTP requests. The HTTP Client should use this characteristic to capture an incoming HTTP Entity Body from an HTTP response.

6.4 HTTP Control Point

The HTTP Client should use the HTTP Control Point characteristic to command the HPS Server to execute the specified HTTP operation.

6.5 HTTP Status Code

The HTTP Client should capture the HTTP status code returned from the addressed HTTP Server, and load it into the Status Code part of the HTTP Status Code characteristic.

6.6 HTTP Security

RFC-2818 [4] implements Transport Layer Security (TLS) over Internet Protocol. The HTTP Client should implement the behavior specified in Section 3.6.1.

7 Acronyms and Abbreviations

Abbreviation or Acronym	Meaning
ER	Executing HTTP Request
GAP	Generic Access Profile
GATT	Generic Attribute Profile
HPS	HTTP Proxy Service
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
RB	Read Body
RH	Read Headers
TLS	Transport Layer Security
URL	Uniform Resource Locator
URI	Uniform Resource Identifier
UUID	Universally Unique Identifier

Table 6.1: *Abbreviations and Acronyms*

v1.0

8 References

[1] Bluetooth Core Specification, Version 4.0 or later

[2] Characteristic and Descriptor descriptions are accessible via the Bluetooth SIG Assigned Numbers

[3] IETF RFC 2616 Hypertext Transfer Protocol – HTTP/1.1
http://www.ietf.org/rfc/rfc2616.txt

[4] IETF RFC 2818 HTTP Over TLS
http://www.ietf.org/rfc/rfc2818.txt

[5] Bluetooth Core Specification Supplement, v5 or later