BLUETOOTH DOC	Date / Year-Month-Day	Approved	Revision	Document No
	2002-09-23	Final	1.0a	
Prepared	e-mail address			N.B.
Bluetooth Printing Working Group	bt-printing-feedback@bluetooth.org			Confidential

HARDCOPY CABLE REPLACEMENT PROFILE

Interoperability Specification

Abstract

This application profile defines the application requirements for Bluetooth™ devices necessary for the support of the Hardcopy Cable Replacement usage model. The requirements are expressed in terms of end-user services and by defining the features and procedures that are required for interoperability between Bluetooth devices in the Hardcopy Cable Replacement usage model.

Revision History

Revision	Date	Comments
0.1 draft	2000-10-16	First draft for PWG review.
0.11 draft	2000-10-30	Changed name to Hardcopy Cable Replacement Profile. Added flow control and notifications. Minor rewording in some sections.
0.2	2000-11-13	Changed version number and made changes in Sections 5, 6, and 7 to rework the notifications a little and clarify some issues.
0.3	2000-11-26	Fixed how control and data channels match up in the SD record. Fixed most outstanding highlighted comments in the document and removed some unnecessary references.
0.4	2000-12-22	Rewrote control channel flow control mechanism. Added credit query and return PDUs. Stressed mandatory and optional features. Changed PDU format to be simpler. Simplified notification section. Miscellaneous changes.
0.5 voting draft	2001-01-25	Explained notifications more eloquently. Added contributors. Grammatical edits.
0.6	2001-03-25	Updated from F2F in Milan and miscellaneous comments.
0.7	2001-04-02	Updated from F2F in Tokyo and miscellaneous reorganization.
0.8	2001-05-14	A few clarifications from the 0.5 feedback.
0.85	2001-05-28	Changes from feedback and discussions.
0.86	2001-06-03	Included new Service Classes and protocol names. Added the CR_NotificationConnectionAlive PDU. Miscellaneous clarifications.
0.87	2001-06-09	Changes from F2F discussions in Monte Carlo.
0.88	2001-06-21	Miscellaneous corrections, word-smithing, and minor functionality changes proposed for Version 0.9.
0.89	2001-06-26	Miscellaneous changes from spec walk-through.
0.895	2001-07-08	Changed SD records to use AdditionalProtocolDescriptorLists fields.
0.896	2001-07-11	Final changes to SD records for 0.9 vote.
0.897	2001-08-07	Incorporated changes from BARB review.
0.9	2001-08-09	Updated version number following vote.
0.91	2001-08-15	Updated with ServiceID and notifications clarifications.
0.94	2001-08-29	Modified notification language. Modified device\service class bits language. Cleaned up comments about status field in PDUs.
0.945	2001-09-05	Modified device/service class bits to include capturing for scanning.
0.95	2001-09-10	Promoted to 0.95 for release.
0.95a	2001-10-11	Added wording to transaction ID initial values. Reordered SD record entries for clarification. Added words about suggested method to use for HCRP device discovery.
1.0	2002-07-19	Includes Errata 2281 (section 7), Errata 2282 (section 9), Errata 2374 (section 8.2), Errata 2391 (Section 6.6.2)
1.0a	2002-09-23	Errata 2474 (Section 6.4.3), Errata 2477 (Section 6), Errata 2478 (Section 7.1, 7.2)

Contributors

Olof Larsson	Axis Communications
Wladyslaw Bolonawski	Ericsson, Inc.
Alan Berkema	Hewlett-Packard Company
Melinda Grant	Hewlett-Packard Company
John Waters	Hewlett-Packard Company
Henrik Holst	i-data International
Jim Combs	Lexmark International, Inc.
Jerry Thrasher	Lexmark International, Inc.
Don Wright	Lexmark International, Inc.
Tom Green	Microsoft Corporation
Joby Lafky	Microsoft Corporation
Thomas Nielson	Microsoft Corporation
Patrick Vine	Microsoft Corporation
Don Levinstone	Motorola, Inc.
Martin Roter	Nokia
Goro Ishida	Seiko Epson Corporation
Brad Emerson	Toshiba Corporation

Disclaimer and Copyright Notice

Contents

List of Tables

Foreword

This document together with the Generic Access Profile forms the Hardcopy Cable Replacement usage model.

Interoperability between devices from different manufacturers is provided for a specific service and usage model if the devices conform to a Bluetooth SIG-defined profile specification. A profile defines a selection of messages and procedures (generally termed *capabilities*) from the Bluetooth SIG specifications and gives an unambiguous description of the air interface for specified service(s) and usage model(s).

All defined features are process-mandatory. This means that if a feature is used, it is used in a specified manner. Whether the provision of a feature is mandatory or optional is stated separately for both sides of the Bluetooth air interface.

1 Introduction

1.1 Scope

The Hardcopy Cable Replacement Profile defines the requirements for the protocols and procedures that shall be used by applications providing the Hardcopy Cable Replacement usage model. This profile makes use of GAP [5] to define the interoperability requirements for the protocols needed by applications. The most common devices using these usage models are mobile laptops and desktop computers, although other devices are not excluded. The usage model includes, but is not limited to, printing and scanning any type of document. The data is rendered through the use of a driver on the client device.

This profile does not include the printing of pure images such as those created by cameras and similar devices; this application is covered by the Still Image Profile [10]. Driverless printing for mobile devices such as mobile phones, pagers, and PDAs is defined in the Basic Printing Profile [9].

There are other existing Bluetooth profiles which could possibly be used for printing and scanning: the Serial Port Profile, Generic Object Exchange Profile, Object Push Profile, File Transfer Profile, and Personal Area Network Profile, as well as potentially others.

However, the Hardcopy Cable Replacement Profile is specialized for hardcopy applications and thus has the following advantages over other more generic profiles:

It supports a 1284ID string, which is a transport-agnostic method for identifying a host driver; this allows the host to leverage existing solutions and extend them cleanly to include Bluetooth.

It provides a very lightweight flow control mechanism appropriate for the high data volume of printing and scanning.

It provides a method for simple asynchronous notifications.

It is connection-oriented [see Section 6]; this should ensure more reliable behavior when a client moves out of range: a printer or scanner can recognize this and abort the current job, rather than losing data in the middle of a job.

At the same time, HCRP is expected to be relatively inexpensive to implement because:

It has a small list of relatively simple control commands.

It is implemented at a low level in the Bluetooth stack, avoiding the overhead of layers such as OBEX, RFCOMM, or PAN. Hopefully this should allow some devices to achieve a data throughput rate near the theoretical limits of the Bluetooth radio channel.

1.2 Bluetooth Profile Structure

In Figure 1 the Bluetooth profile structure and the dependencies of the profiles are depicted. A profile is dependent upon another profile if it re-uses parts of that profile by implicitly or explicitly referencing it. Dependency is illustrated in the figure: a profile has dependencies on the profile(s) in which it is contained – directly and indirectly. For example, the Hardcopy Cable Replacement Profile is dependent on the Generic Access Profile.

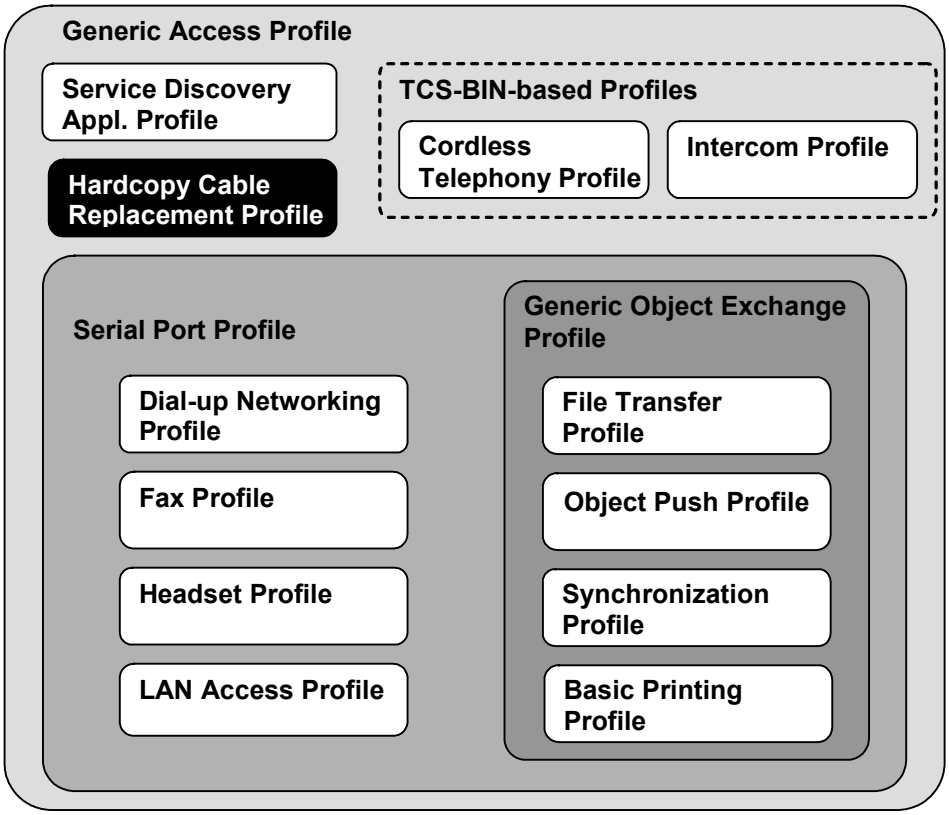


Figure 1: Bluetooth Foundation Specification Profiles plus the Hardcopy Cable Replacement Profile

1.3 Related Specifications

Bluetooth Basic Printing Profile Specification

Application Profile for Basic Printing applications.

Defines the interoperability requirements for the applications within the Basic Printing application profile.

Does not define the requirements for the Baseband, LMP, L2CAP, or RFCOMM.

Bluetooth Hardcopy Cable Replacement Profile Specification

Application profile for connecting devices such as printers and scanners to devices such as laptops and desktop computers without a physical cable.

Defines the interoperability requirements for the applications within the Hardcopy Cable Replacement application profile.

Does not define the requirements for the Baseband, LMP, or L2CAP.

1.4 Symbols and Conventions

1.4.1 Requirement Status Symbols

In this document the following symbols are used:

"M" for mandatory to support (used for capabilities that shall be implemented in the profile)

"O" for optional to support (used for capabilities that may be implemented in the profile)

"C" for conditional support (used for capabilities that shall be implemented in case a certain other capability is supported)

"X" for excluded (used for capabilities that may be supported by the unit but shall never be used in the profile)

"N/A" for not applicable (in the given context it is impossible to use this capability)

Some excluded capabilities are capabilities that, according to the relevant Bluetooth specification, are mandatory. These are features that may degrade operation of devices following this profile. Therefore, these features shall never be activated while a unit is operating as a unit within this profile.

1.4.2 Word Usage

When the words "shall", "should", "may", and "can" are used in this profile, they have the meanings described in the IEEE Standards Style Manual [11].

2 Profile Overview

2.1 Protocol Stack

Figure 2 shows the protocols and entities used in this profile.

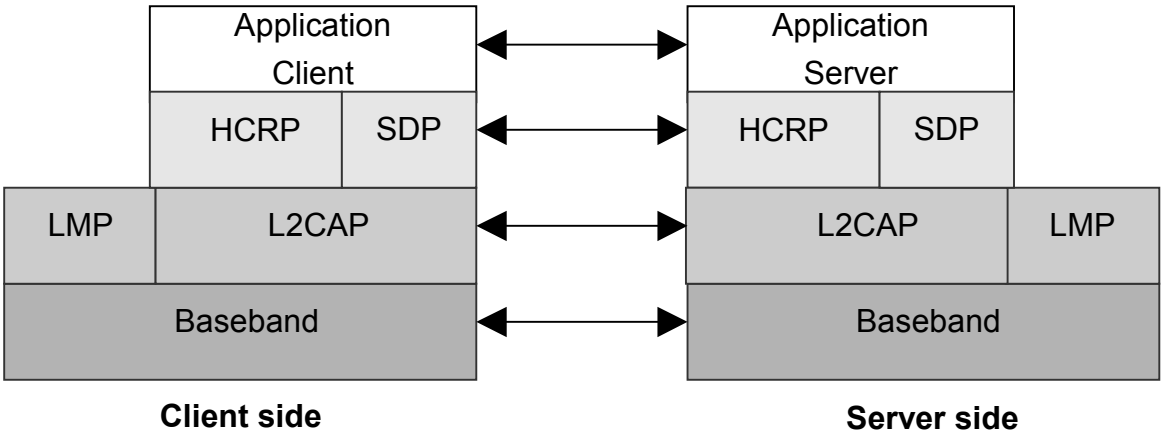


Figure 2: Protocol Model

The Baseband [1], LMP [2], and L2CAP [3] are the OSI layer 1 and 2 Bluetooth protocols. SDP is the Bluetooth Service Discovery Protocol [4]. HCRP is the Hardcopy Cable Replacement Profile (this profile).

Interoperability between devices using this profile depends on the presence of peer applications running on top of the HCRP layer in both the Client and the Server. On the client side this application takes the form of a driver; on the server device, it is usually an interpreter or generator of a page description format. This application layer is outside the scope of this profile. If the driver application is not available on the client, interoperability between the devices may fail.

2.2 Configurations and Roles

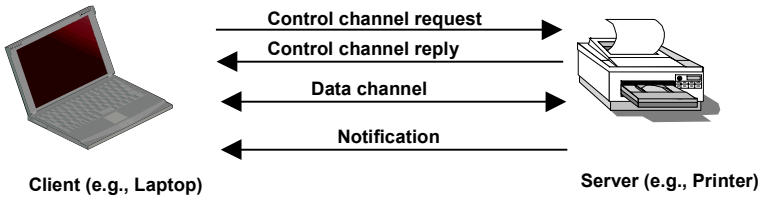


Figure 3: Example, Laptop Sending a Print Job to a Printer

The following roles are defined for this profile:

Server

This is the server device that offers the Hardcopy Cable Replacement Profile as a service. It can receive binary data from the client. Alternatively it may send binary data back to the client as it is requested. Connections are generally made by the client, but the server may be able to connect to a client to notify the client of changes in the server if the client has registered for notifications from the server. In addition to the interoperability requirements defined in this profile, the client and server shall comply with the requirements of the data protocol that they exchange – this is tightly bound between the client and the server. The server shall comply with the interoperability requirements for GAP if not defined to the contrary.

Client

This is the client device that connects to the server to use a particular function of the server. In addition to the interoperability requirements defined in this profile, the client shall comply with the requirements of the data protocol that the client says it understands. The client shall also comply with the interoperability requirements for GAP if not defined to the contrary.

Notifications

When a client connects to a server, it may register for notifications, giving the server sufficient information to allow it to connect back to the client. This is to enable the server to connect to the client in an asynchronous manner if the server needs to. The notification functionality specified by this profile provides the framework for a host application and printer to communicate event-driven information. The actual events communicated and the format or encoding of those events are outside the scope of this specification; they may be anything mutually-understood by the host application and printer.

2.3 User Requirements and Scenarios

The scenarios covered by this profile are:

- Printing any type of document. The document would be rendered to the relevant PDL on the server and sent in the appropriate format to the printer.
- Scanning documents through the use of a driver on the client device.

Servicing multiple clients simultaneously for printing and status is a potential user scenario but it is not a user requirement. The server device may support multiple connections, but this is not required. How the server services multiple jobs simultaneously is implementation-specific and beyond the scope of this profile. If a server cannot support any more clients connecting to it, it is required to either make itself unavailable for service discovery or refuse connections.

2.4 Profile Fundamentals

Link and channel establishments shall be done according to the procedures defined in L2CAP [3] unless otherwise stated in this specification.

Link-level authentication and encryption are mandatory to support and optional to use; see Section 8.1.

Bonding [5] is mandatory to support and optional to use; see Section 8.1.

There are no fixed master-slave roles for the execution of this profile. The master-slave switch is optional to support and optional to use.

This profile does not require any low power mode to be used.

The Bluetooth L2CAP connections are configured in such a way as to guarantee reliable delivery of data. Refer to Section 6.2, Flush Timeout Option in L2CAP [3] for further information.

2.5 Conformance

When conformance to this profile is claimed, all capabilities indicated as mandatory for this profile 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 certification program.

3 Device Modes

The server device can be in four different modes. The tables below describe these modes and the requirements put on the server device in each mode.

Device Mode	Requirement	Description
Bluetooth Offline mode *	Optional	Device is not able to receive data over Bluetooth. It is not possible to connect to the device. The device cannot be discovered using either of the two inquiry procedures described in [5].
Bonding mode	Mandatory	Device is ready to be bonded with client devices.
Private Online mode	Optional	Device can be connected to and used over Bluetooth only from devices that know the device's Bluetooth device address. The device cannot be discovered using either of the two inquiry procedures described in [5].
Public Online mode	Mandatory	Device can be connected to and used over Bluetooth from devices that know the Printer's Bluetooth device address. The device can be discovered using one or both of the two inquiry procedures described in [5].

Table 1: Device Modes and Requirements

* Bluetooth Offline mode refers to the printer’s Bluetooth link.

The table below describes the relationship between the device modes and the Bluetooth GAP modes (see [5]).

Device Mode	GAP Mode when Device is Idle	GAP Mode when Device is Busy
Bluetooth Offline mode *	Mandatory: Non-connectable mode Mandatory: Non-discoverable mode	Mandatory: Non-connectable mode Mandatory: Non-discoverable mode
Bonding mode	Mandatory: Connectable mode Mandatory: Either limited discoverable mode, general discoverable mode, or both Mandatory: Pairable mode	N/A
Private Online mode	Mandatory: Connectable mode Mandatory: Non-discoverable mode	Optional: Connectable mode Mandatory: Non-discoverable mode
Public Online mode	Mandatory: Connectable mode Mandatory: Either limited discoverable mode, general discoverable mode, or both	Optional: Connectable mode Optional: Either limited discoverable mode, general discoverable mode, or both

Table 2: GAP Modes Associated with Device Modes

* Bluetooth Offline mode refers to the printer’s Bluetooth link.

When entering Bonding mode, Private Online mode, and Public Online mode, devices shall register a service record in the SDDB (see Section 7).

4 User Interface Aspects

In the following sections, the presented scenarios work as examples and variations in the allowed implementations.

4.1 Print Job, Bluetooth Device Address of Printer Known

When a client wants to print a job, the following scenario may be followed. In this case the client already knows the Bluetooth device address of the printer.

If link-level authentication is used, the user might have to enter a Bluetooth PIN at some point.

Client	**Server (Printer)**
	The server is in (or is set by the user into) Private Online mode or Public Online mode.
The user of the client device selects the print function on the device.	
The client connects to the server with the known Bluetooth device address. If the client cannot connect to the server, the server may be out of reach, busy, or not in either of the two Online modes. It is recommended that this be communicated to the user. The server is connectable; the client device queries the server device to determine what driver needs to be loaded to talk to the server. A query to the server's IEEE 1284 ID string or any other method may be used for this purpose. If the driver is not locally available on the client or the server does not support a suitable method for determining which driver to load, the user is prompted for help to install the driver. The user may choose to install the driver or to cancel the process.	
The client renders the data using the driver and then sends it to the printer.	
	The printer receives the data and prints it. If the printer is busy or if an error occurs, the printer returns a status code to the client.
The client disconnects. It is recommended that the user be notified of the result of the operation.	

4.2 Print Job, Bluetooth Device Address of Printer Not Known

When a client wants to print a job on a printer, the following scenario may be followed. In this case the client does not know the Bluetooth device address of the printer.

If link-level authentication is used, the user might have to enter a Bluetooth PIN at some point.

Client	Server (Printer)
	The server is in (or is set by the user into) Public Online mode.
The user of the client device selects the print function on the client device and selects the object to print.	
A list of server devices that may support the Hardcopy Cable Replacement service with a print function is displayed to the user. The user selects a server device to send the rendered data to. If the desired printer is not in the list of server devices or if the client cannot connect to the printer, the printer may be out of reach, busy, or not in either of the two Online modes. It is recommended that this be communicated to the user. If the selected server device does not support the Hardcopy Cable Replacement service with the print function, the user is prompted to select another device. The client device queries the server device to determine what drivers need to be loaded to talk to the server. The IEEE 1284 ID string or any other method may be used for this purpose. If the driver is not locally available on the client device or the server does not support a suitable method for determining which driver to load, the user is prompted for help to install the driver. The user may choose to install the driver or to select to use another server device.	
The client renders the data using the driver and then sends it to the printer.	
	The printer receives the data and prints it. If the printer is busy or if an error occurs, the printer returns a status code to the client.
The client disconnects. It is recommended that the user be notified of the result of the operation.	

4.3 Rich Status, Receive Notifications from Server Device

If a client wants to receive rich status in the form of notifications from a server device, the client needs to register for notifications from the server device. This means that even if there is no current connection between the client and server, the server can connect to the client and pass rich status information asynchronously. A notification is an asynchronous event from the server to the client notifying the client of a change in state in the server.

An example of a printer notification is an end-of-job notification. A job may be spooled to a printer's hard drive prior to being processed. This would mean that the data being sent to the printer over the Bluetooth link would complete sending before the printer has completed printing. To allow the sender not to have to maintain a connection to receive an end-of-job notification from the printer, the notification channel may be used.

An example of a scanner notification is a scanner button press event. A user may walk up to a scanner and press the scan button on the front panel. The scanner would then send a notification of the scan button press to all registered clients. This allows clients to not have to have an open connection at all times to receive this type of event.

Authentication should never need to occur when the server connects back to the client. Authentication should occur when a client initially connects to a server, hence re-authenticating when a server connects back to a client should be unnecessary. If, for any reason, authentication requires user interaction during this process, the connection request may be cancelled by the server, as it may not have a user interface to allow a user to authenticate through.

Client	**Server (Printer)**
The client is in a connectable mode.	
	The server needs to send a notification to the client. A list of client devices that expect to receive notifications from the server is stored in the server. The server connects to each client device and sends the notification data that it needs to send down the notification channel.
The client receives the connection. The client reads the notification data from the notification channel. The client deals with the notification data as it wishes. The client closes the notification channel to notify the server that the notification has been dealt with.	
	The server may close the notification channel if the client has not closed it after the negotiated time out. (See Section 6.4.15.)
If the negotiated timeout is about to expire, the client may attempt to increase the timeout period by renewing it. (See Section 6.4.16.)	

Client	Server (Printer)
	The server may optionally accept a request for increasing the time period if the client requests a renewal or it may timeout as initially negotiated and close the notification channel. (See Section 6.4.16.)
Once the notification process has completed, if the client wishes to receive new notifications, it shall reregister for notifications. (See Section 6.4.15.)	

5 Application Layer

5.1 Hardcopy Cable Replacement Application

In this section the operational framework of the hardcopy cable replacement application is presented.

The hardcopy cable replacement application has the following features

- A “control” L2CAP channel is used to transmit out-of-band control requests to the server device.
- A “data” L2CAP channel is used to transmit and receive raw, device-specific data to and from a server device. The contents of this channel are not specified by this profile.
- The “control” channel may be used by the client to register for notifications. This allows the server device to initiate a connection to the client if it needs to notify the client that it has information for the client. (See Section 6.4.16.)
- The “notification” channel is used by the server to notify a registered client about asynchronous events. (See Section 6.6.)
- The “control” channel may be used to obtain the LPT status bits from the server device. The definition of the LPT status bits is given in Section 6.4.12.
- The “control” channel may be used to obtain the IEEE 1284 ID string from the server device. The IEEE 1284 ID string is defined in [8].
- The “control” channel may be used to “soft reset” the server device. “Soft reset” is defined in Section 6.4.14.
- The “control” channel may be used to “hard reset” the server device. “Hard reset” is defined in Section 6.4.15.
- The “control” channel is used to transmit flow control information for the data channel. The credit flow control mechanism is defined in Section 6.2.

The “control” channel protocol data unit format and defined values are described in Section 6.4.1. The PDU identifiers in the range 0x0000 to 0x7FFF are either defined in this profile or reserved for future use. The PDU identifiers in the range 0x8000 to 0xFFFF are left for vendor-specific control requests.

6 Protocol Layer

The Hardcopy Cable Replacement Profile defines a simple protocol for communicating with hardcopy devices as if they were connected over a locally attached cable.

The protocol uses two connection-oriented L2CAP channels. These are the data and control channels. The protocol uses an additional L2CAP channel for notifications. The data and control channels are exposed by the server, and always initiated by the client. The notification channel is optionally exposed by the client. Notifications are always initiated by the server, and only to clients that have registered for notifications with that server.

The “control” L2CAP channel is used to transmit out-of-band control requests to the server device. All communications on this channel will be initiated by the client device.

The “data” L2CAP channel is used to transmit and receive raw, device-specific data to and from a server device. The contents of this channel are not specified by this profile.

The data and control channels may be connected in any order. In some cases, only the control channel is required in order to perform the desired operation (e.g., notification registration, hard reset, etc.). In this case, the data channel may never be connected. Although credits may be exchanged via the control channel prior to creation of the data channel, no data shall be transmitted on the data channel before credit is made available for the appropriate transfer direction. Credits for client to server data are provided using the CR_DataChannelCreditRequest command and server to client data credits are exchanged using the CR_DataChannelCreditGrant command.

The optional “notification” L2CAP channel may be used by the server to notify the client of asynchronous events.

The Hardcopy Cable Replacement Protocol defines only the method of communication between client and server devices on the “control” channel and “notification” channels. The Hardcopy Cable Replacement Protocol does not define any data that runs on the data channel nor how it is transmitted over the data channel.

6.1 Transfer Byte Order

The Hardcopy Cable Replacement Protocol transfers bytes in the standard network order (big-endian), with more significant (high-order) bytes being transferred before less significant (low-order) bytes.

6.2 Flow Control

L2CAP in Bluetooth 1.1 lacks a flow control mechanism. However, printing and scanning typically require flow control due to the volume of data involved. Therefore the Hardcopy Cable Replacement Protocol provides its own flow control mechanism. The mechanism is simple, lightweight, and credit-based.

Should L2CAP include optional or mandatory flow control in the future, the Hardcopy Cable Replacement Profile would then not require flow control functionality. If such L2CAP flow control exists and can be detected to be in use by both sides of an L2CAP connection at run time, then the simple flow control defined here should be effectively switched off by both sides of the connection granting a large credit grant. And, even if an L2CAP-based flow control mechanism is active but is not detected by the HCRP layer, the additional overhead of the HCRP-based flow control mechanism is not believed to be significant.

6.2.1 Overview

The Hardcopy Cable Replacement Profile uses a simple credit-based flow control mechanism. The control channel is defined in such a way as not to require flow control and the data channel's flow control is administered by the control channel. This ensures that the flow control remains simple.

Before a device may transmit on the data channel, the device shall first receive "credit" from the other device. A credit grant transaction is a message from a client device to a server device granting permission to transmit a specific number of bytes. A credit request transaction is a message from a client device to a server device requesting that the server grant credit to the client. The server decides how many bytes of credit to grant the client device.

All credit grants or requests originate from the client device and are transacted on the control channel. When a client wishes to have credit to transmit on the data channel, it sends a credit request to the server. When the client wishes to grant credit to the server device, it sends a credit grant message. Client devices should periodically grant and request credit to guarantee a continuous flow of data. If a device is capable of accepting large amounts of data very quickly, it will grant larger or more frequent credit grants to its peer. If a device is not capable of processing data as quickly, it will grant smaller or less frequent credit grants.

Credit grants are cumulative. When a device receives credit, it increases the total credit available to the device. When a device transmits bytes over the connection, it deducts the number of bytes transmitted from its total available credit. When a device's available credit reaches zero, the device shall not transmit anything on the data channel until additional credit has been received.

A device shall never grant credit to a remote device that would cause the total credit to overflow a 4-byte value.

When the data and control connections are closed, all credit granted to either the server or client on those connections is zeroed. When initiating new connections, the client will need to grant credit to the server as well as request credit from the server to transmit data on the data channel. When either the data or control connection closes, the related control or data channel shall be closed as well.

Note that if at some stage there is flow control provided at a lower level, this flow control mechanism can be effectively switched off by a client or server by granting massive amounts of credit at once. However, the available credit will still need to be monitored and responded to by a client or server if the credit outstanding reached zero, as it is possible to send more data across the data channel than the initial massive credit grants.

The control channel uses a request and reply method to ensure flow control. Only one control PDU may be outstanding at a time. The client and server shall each provide at least a 128-byte buffer for the control channel, ensuring that the maximum-sized PDU can be received and processed intact.

6.2.2 Data and Control Channel MTUs

For simplicity, the client and the server device shall each provide the ability to buffer at least 128 bytes on the control channel. This implies that the L2CAP MTU negotiated by each side of the control channel connection shall be at least 128 bytes. The data channel L2CAP connection is negotiated separately, and may be any size allowed by L2CAP. It is recommended to have a larger control channel MTU than the minimum allowed value in order to efficiently service the CR_Get1284ID transaction.

The client or server will need to fragment the data stream that it is sending down the data channel. This data may be fragmented into chunks any size less than or equal to the negotiated MTU for the remote endpoint of the L2CAP data channel. The client shall never send more data on the data channel than it has credit, as defined in this section. The client or server shall be able to receive data in any size less than or equal to the negotiated MTU for the local endpoint of the L2CAP data channel. The client or server shall assume nothing about the received data beyond that it is a data stream of printer or scanner-specific data to be passed up to the consumer of that data. The control channel is completely unaffected by segmentation and reassembly because there is a guaranteed minimum MTU size for the control channel as defined in this section (128 bytes). This ensures that a complete request or reply can be transmitted as a single L2CAP packet.

6.2.3 Initialization

All traffic across the data channel consumes credit. When a client connects to a server, the initial credit granted on the data channel to both devices is zero. Because of this, before any data may be sent on the data channel by either device, credit transactions to grant and request credit are necessary.

The control channel shall be able to buffer at least 128 bytes of data. The data channel will negotiate the size of its MTU at connection time.

If the data channel closes, the control channel shall close as well, and all credit granted for the data channel is reset to zero. If the channel reconnects, it shall start with the initialization credit of zero.

6.2.4 Channel Utilization

Flow control is only provided for the data channel. All control messages, including credit grants, take place in the control channel. As a result, credit grants for the data channel travel in the control channel. This ensures that there can never be a credit deadlock.

6.3 L2CAP Transmission Errors

L2CAP in Bluetooth 1.1 provides a relatively reliable connection between the client and server (see Section 6.2, Flush Timeout Option in L2CAP [3]). All data is guaranteed to be delivered despite interference, up to the point where the entire connection is dropped. (However, in the presence of very high interference with respect to the 16-bit CRC code of L2CAP, there is a finite possibility of a corrupted packet falsely being declared to be valid. This is expected to be corrected in post-Bluetooth 1.1 L2CAP specifications.) If the connection is dropped or a specified failure timeout is reached, the job should be aborted. Therefore, the Hardcopy Cable Replacement Profile provides no transmission error recovery. A recommended failure timeout is 5 minutes.

6.4 Control Channel Protocol

The control channel defines a format for PDUs as well as several PDUs that are transferred on the control channel.

6.4.1 Protocol Data Unit Format

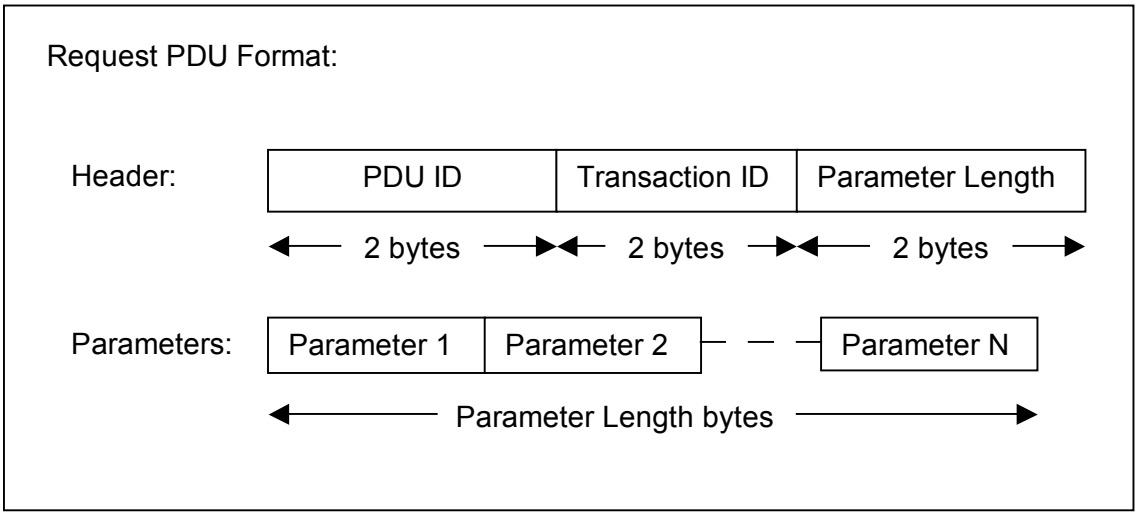
Each control channel transaction consists of a request from a client and a reply from the server. Each request takes the form of a request PDU, and every response takes the form of a reply PDU. Only one request may be outstanding at a time. The client may not initiate a new transaction until the previous transaction has been completed by the server responding with a reply PDU.

A client may close both the control and data connections if it does not receive a reply PDU within 5 seconds of transmitting a request PDU.

6.4.2 Request PDUs

Each request PDU contains a PDU ID, a transaction ID, a parameter length field, and a variable number of parameter values.

The PDU ID specifies the particular request being made.

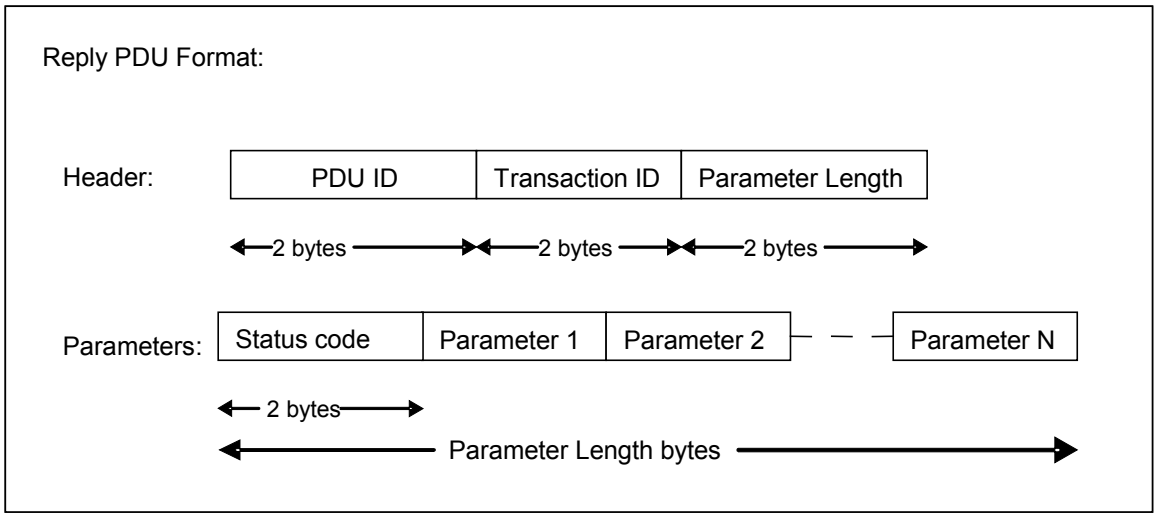


The transaction ID, which is incremented with each transaction, ensures that reply PDUs are correctly in sync with request PDUs.

The parameter length field specifies the remaining length of the PDU, not including the parameter length field itself. The parameter length field provides the only information regarding where one PDU ends and the next begins.

The parameter values provide additional request-specific information to the server.

6.4.3 Reply PDUs



Each reply PDU contains a PDU ID, a transaction ID, a parameter length field, a status code, and a variable number of parameter values. The transaction ID and PDU IDs are the same values as those of the received request PDU.

The status code specifies whether the request succeeded or failed. If the status code is 0x0000 (unsupported), then the server need not provide the return parameter values otherwise specified for a particular request. In all other cases, the reply PDU

shall maintain the parameter format expected by the client in response to the request.

The reply parameters of a PDU may be different from the request parameters.

The Parameter Length field includes the 2-byte size of the status code field in the reply PDU.

Note: The best way to uniquely match a reply PDU to a request PDU is via the transaction ID and PDU ID jointly.

6.4.4 PDU Identifiers

The credit grant and credit request PDUs are MANDATORY to implement on both clients and servers. All other PDU transactions are OPTIONAL for both client and server.

Value	Description	Client Status	Server Status
N	The PDU ID field identifies the type of PDU (i.e., its meaning and that of its parameters). *	Whether the PDU is MANDATORY or OPTIONAL to implement on the client	Whether the PDU is MANDATORY or OPTIONAL to implement on the server
0x0000	Reserved	N/A	N/A
0x0001	CR_DataChannelCreditGrant	MANDATORY	MANDATORY
0x0002	CR_DataChannelCreditRequest	MANDATORY	MANDATORY
0x0003	CR_DataChannelCreditReturn	OPTIONAL	OPTIONAL
0x0004	CR_DataChannelCreditQuery	OPTIONAL	OPTIONAL
0x0005	CR_GetLPTStatus	OPTIONAL	OPTIONAL
0x0006	CR_Get1284ID	OPTIONAL	OPTIONAL
0x0007	CR_SoftReset	OPTIONAL	OPTIONAL
0x0008	CR_HardReset	OPTIONAL	OPTIONAL
0x0009	CR_RegisterNotification	OPTIONAL	OPTIONAL
0x000A	CR_NotificationConnectionAlive	OPTIONAL	OPTIONAL
0x000B-0x7FFF	Reserved	N/A	N/A
0x8000-0xFFFF	Vendor-specific	OPTIONAL	OPTIONAL

Table 3: Control Channel PDU IDs

* The PDU ID of a reply PDU shall be the same as that of the request PDU.

6.4.5 PDU Header Fields

Transaction ID Values

Value	Description
N	The transaction ID uniquely identifies a request PDU and is used, along with the PDU ID, to match reply PDUs to request PDUs. The transaction ID shall increment for each transaction request. The transaction ID shall wrap to 0x0000 when incrementing past 0xFFFF. The transaction ID in the reply PDU shall be the same as the request transaction ID that it is responding to. If for any reason the client or server receives a transaction ID that is not the one they expect, the behavior is undefined and the client or server shall close both the control and data L2CAP channels. When an initial connection is made, the initial value for the transaction ID may be any value. Range: 0x0000-0xFFFF

Parameter Length Values

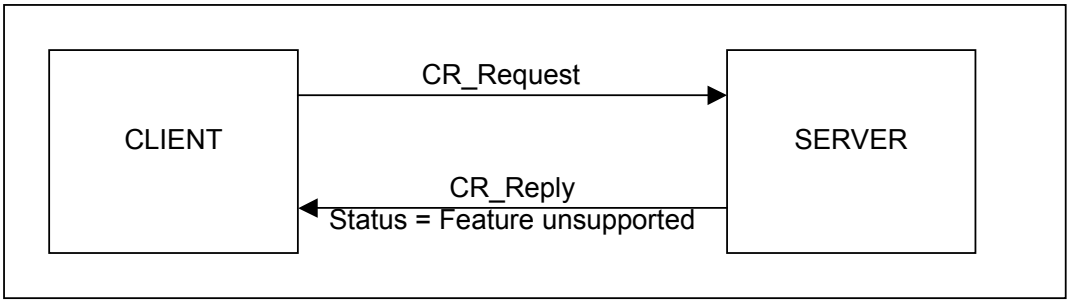
Value	Description
N	The parameter length specifies the size (in bytes) of all the parameters contained in the PDU. Request PDU Range: 0x0000-0xFFFF Reply PDU Range: 0x0002-0xFFFF (2 bytes due to the 2-byte status code)

Status Code Values for Reply PDUs

Value	Description
N	The status code identifies the result for a given PDU request.
0x0000	Feature unsupported
0x0001	Success
0xFFFF	Generic failure
0x0002	Credit synchronization error

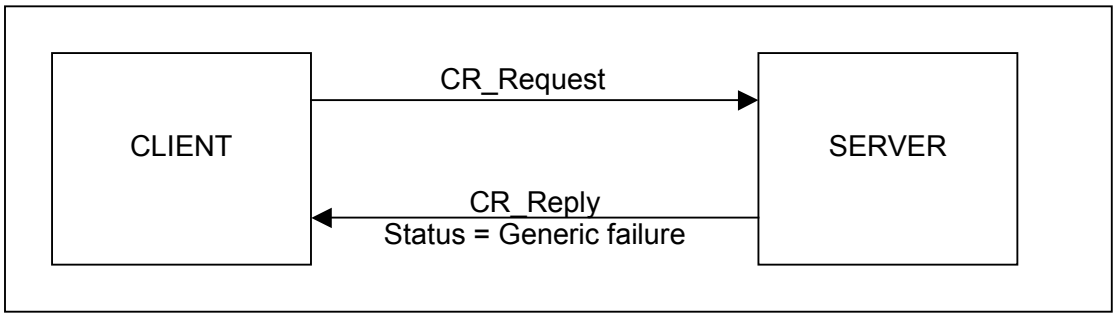
6.4.6 Unsupported Feature

Each transaction consists of a request and a reply PDU. If a server does not support a specific request PDU, it shall always respond with a reply PDU with the status code parameter containing the not supported error status (0x0000). This is the only time that a reply PDU may omit defined parameters expected for a reply PDU of that type.

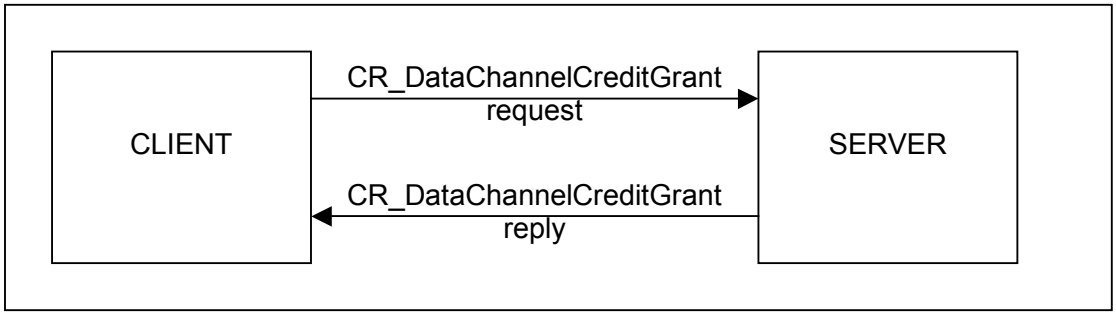


6.4.7 Error Handling

Every transaction consists of a request and a corresponding reply PDU. If a server determines that there is a problem with a specific request PDU for whatever reason, it will respond with a fully-formed reply PDU for that request with the status code parameter containing an error status code.



6.4.8 CR_DataChannelCreditGrant



Description

CR_DataChannelCreditGrant is used by the client to increase the number of bytes of credit the server device has available for transmission on the data channel for transmission from server to client. Clients should periodically grant the server credit to transmit on the data channel to ensure an uninterrupted flow of data. An extremely fast client device with very large buffers might grant several megabytes of credit at once. Obviously any credit grant should be based on the maximum throughput and buffering capability of the client.

A credit grant of zero bytes is a successful credit grant as long as nothing else goes wrong with the transaction.

A client shall never grant credit to a server that would cause the total credit to overflow a 4-byte value (2^{32} bytes).

CR_DataChannelCreditGrant PDU Request

PDU Type	PDU ID	Request PDU Parameters
CR_DataChannelCreditGrant	0x0001	CreditGranted

PDU Parameters

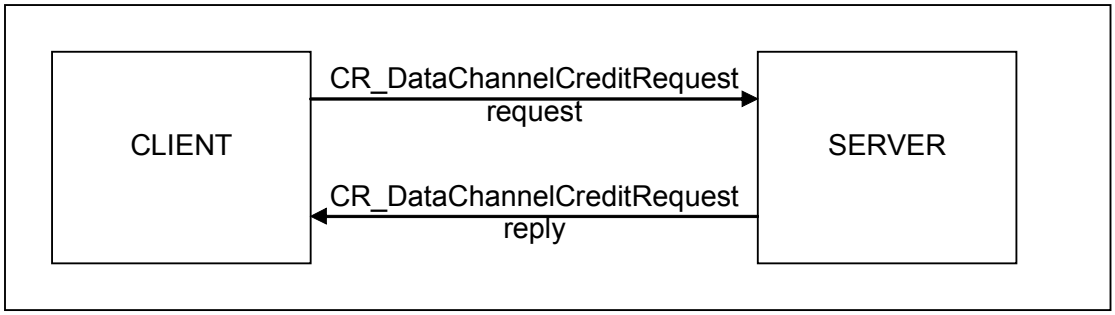
CreditGranted *Type: 32-bit unsigned integer* *Size: 4 Bytes*

Value	Description
0xXXXXXXXX	The amount of credit the client is granting to the server for the data channel.

CR_DataChannelCreditGrant PDU Reply

PDU Type	PDU ID	Reply PDU Parameters
CR_DataChannelCreditGrant	0x0001	None

6.4.9 CR_DataChannelCreditRequest



Description

CR_DataChannelCreditRequest is used by the client to request an increase in the number of bytes of credit it has for transmission on the data channel. The client requests credit from the server. The server returns an amount of credit that it wishes to grant the client. An extremely fast server device with very large buffers might grant several megabytes of credit at once. A server may deny a credit request by returning zero bytes of credit granted. Returning zero bytes of credit is a successful transaction as long as nothing else went wrong.

A server shall never grant credit to a client that would cause the total credit to overflow a 4-byte value (2^{32} bytes).

CR_DataChannelCreditRequest PDU Request

PDU Type	PDU ID	Request PDU Parameters
CR_DataChannelCreditRequest	0x0002	None

CR_DataChannelCreditRequest PDU Reply

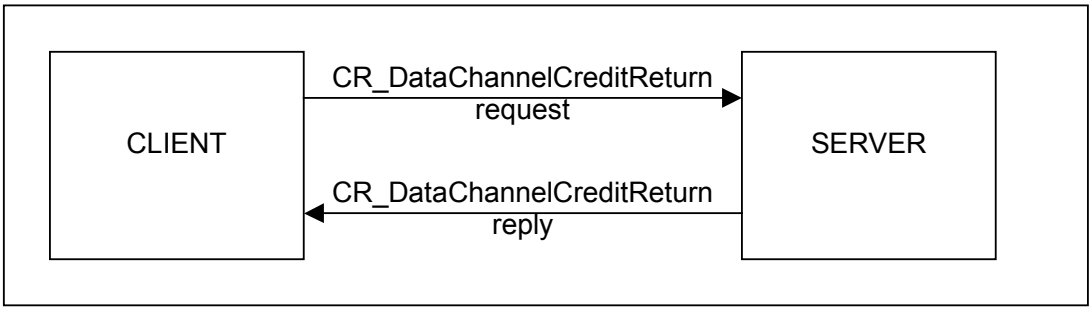
PDU Type	PDU ID	Reply PDU Parameters
CR_DataChannelCreditRequest	0x0002	CreditGranted

PDU Parameters

CreditGranted *Type: 32-bit unsigned integer* *Size: 4 Bytes*

Value	Description
0xXXXXXXXX	The number of bytes of credit the server is granting to the client for the data channel.

6.4.10 CR_DataChannelCreditReturn



Description

CR_DataChannelCreditReturn is used by the client to return granted credit and to see whether the server wishes to return any credit. Credit returns allow a device that granted credit to reclaim and reuse buffers that were allocated to that channel.

If the client or server intends to maintain the connection for any period of time after returning credit, it is recommended that the device does not return all outstanding granted credit in case it needs the credit for another transaction.

A device shall *never* return more credit than it has outstanding to use. If the ClientCreditReturn parameter of the CR_DataChannelCreditReturn request indicates more bytes are being returned than were provided as credit, the server device shall return the error code Credit synchronization error as the StatusCode in the corresponding reply. In this case, the client shall close the data and control L2CAP channels.

CR_DataChannelCreditReturn PDU Request

PDU Type	PDU ID	Request PDU Parameters
CR_DataChannelCreditReturn	0x0003	ClientCreditReturn

PDU Parameters

ClientCreditReturn *Type: 32-bit unsigned integer* *Size: 4 Bytes*

Value	Description
0xXXXXXXXX	The number of bytes (previously granted to and not yet used by the client) that the client wishes to return to the server as it will never use the bytes.

CR_DataChannelCreditReturn PDU Reply

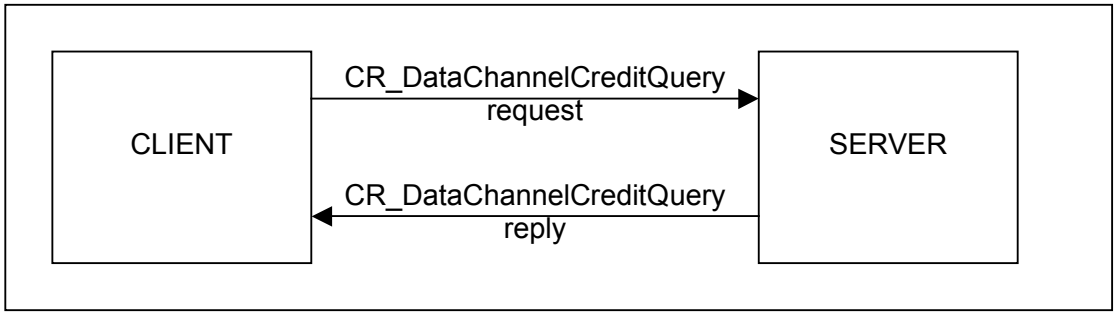
PDU Type	PDU ID	Reply PDU Parameters
CR_DataChannelCreditReturn	0x0003	ServerCreditReturn

PDU Parameters

ServerCreditReturn *Type: 32-bit unsigned integer* *Size: 4 Bytes*

Value	Description
0xXXXXXXXX	The number of bytes (previously granted to the server but not yet used) that the server wishes to return to the client as it will never use the bytes.

6.4.11 CR_DataChannelCreditQuery



Description

CR_DataChannelCreditQuery is initiated by the client. The client transmits how much outstanding credit that it currently has available on the data channel. The server in turn will return how much remaining credit it has to transmit on the data channel.

Note: This feature is intended primarily for debugging use. It is recommended that credit control schemes not be defined around this feature as it is optional to implement.

CR_DataChannelCreditQuery PDU Request

PDU Type	PDU ID	Request PDU Parameters
CR_DataChannelCreditQuery	0x0004	ClientCredit

PDU Parameters

ClientCredit *Type: 32-bit unsigned integer* *Size: 4 Bytes*

Value	Description
0xXXXXXXXX	The number of bytes previously granted to the client but not yet used.

CR_DataChannelCreditQuery PDU Reply

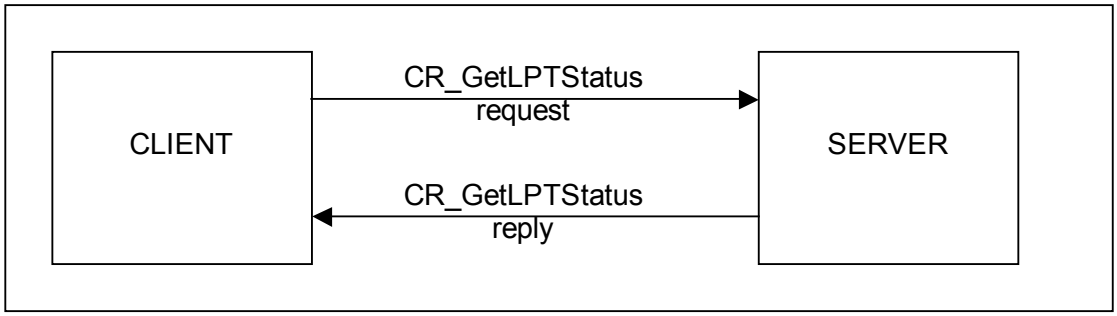
PDU Type	PDU ID	Reply PDU Parameters
CR_DataChannelCreditQuery	0x0004	ServerCredit

PDU Parameters

ServerCredit *Type: 32-bit unsigned integer* *Size: 4 Bytes*

Value	Description
0xXXXXXXXX	The number of bytes previously granted to the server but not yet used.

6.4.12 CR_GetLPTStatus Transaction



Description

The CR_GetLPTStatus is used for obtaining the IEEE 1284 job status bits as defined below. These bits are compatible with the status register of a standard PC parallel port. The reply parameter is a byte containing the status bits of the current job.

Printer Port Status Bits

7	6	5	4	3	2	1	0
Reserved		Paper Empty	Select	Not Error	Reserved		

Printer Port Status Bit Meanings

Bits	Field	Description
7..6	Reserved	Reserved for future use; device shall return these bits set to zero.
5	Paper Empty	1= Paper Empty, 0 = Paper Not Empty
4	Select	1 = Selected, 0 = Not Selected
3	Not Error	1 = No Error, 0 = Error
2..0	Reserved	Reserved for future use; device shall return these bits set to zero.

CR_GetLPTStatus Request

PDU Type	PDU ID	Request PDU Parameters
CR_GetLPTStatus	0x0005	None

CR_GetLPTStatus Reply

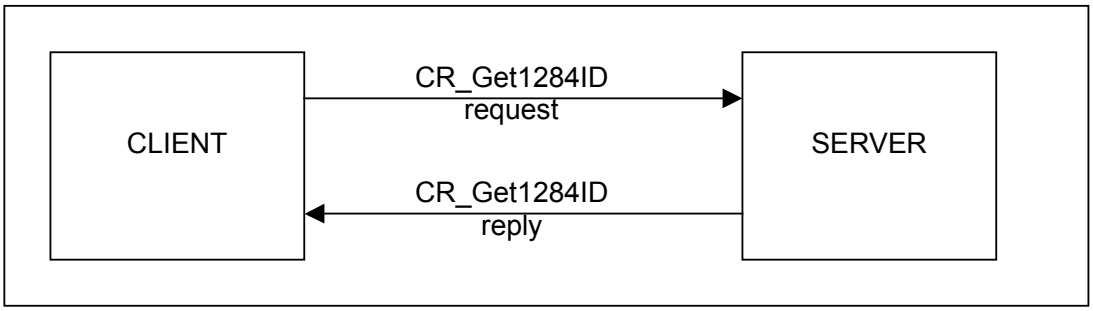
PDU Type	PDU ID	Reply PDU Parameters
CR_GetLPTStatus	0x0005	LPTStatus

PDU Parameters

LPTStatus *Type: Byte* *Size: 1 Byte*

Value	Description
0xXX	The IEEE 1284 bits set as defined above.

6.4.13 CR_Get1284ID Transaction



Description

The CR_Get1284ID is used for obtaining the IEEE 1284 ID string from the server device. The 1284 ID string is defined in [8]. There are two request parameters and there are two reply parameters. The second reply parameter is a string as defined in [8].

The two request parameters are the start value and number of bytes to return. This ensures that the IEEE 1284 ID string can be broken down into the maximum server control channel buffer size. The start value is zero-based. This means that a start value of zero is the first byte in the string.

The number of bytes requested may not exceed the control channel MTU negotiated. If this value does exceed the control channel MTU negotiated, the server shall return the number of bytes that can fit into the control channel MTU. It is recommended to have a larger control channel MTU than the minimum allowed value in order to efficiently service this request.

CR_Get1284ID PDU Request

PDU Type	PDU ID	Request PDU Parameters
CR_Get1284ID	0x0006	StartByte, NumberOfBytes

PDU Parameters

StartByte *Type: Unit16* *Size: 2 Bytes*

Value	Description
0xXXXX	The starting location in the IEEE 1284 ID string from which to start obtaining data. This is zero-based, so a value of zero implies the first byte in the string.

NumberOfBytes *Type: Unit16* *Size: 2 Bytes*

Value	Description
0xXXXX	The number of bytes to return. This may not exceed the control channel MTU negotiated. If this does exceed the control channel MTU negotiated, the server shall return the number of bytes that can fit into the control channel MTU.

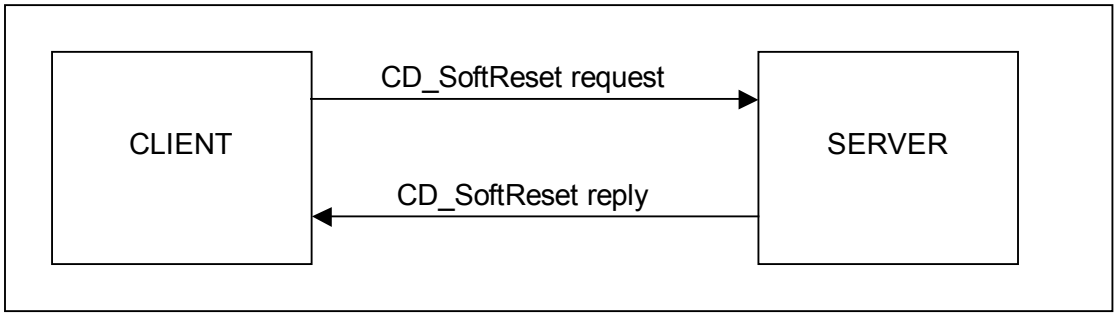
CR_Get1284ID PDU Reply

PDU Type	PDU ID	Reply PDU Parameters
CR_Get1284ID	0x0006	1284ID

1284ID *Type: String* *Size: Variable*

Value	Description
Variable length string.	1284 ID string as defined in [8] starting with the byte at the StartByte and length of NumberOfBytes bytes.

6.4.14 CR_SoftReset Transaction



Description

The soft reset is used to clear all currently spooling jobs submitted by the initiating client from the server device. All jobs that have completed spooling inside the device will be completely unaffected. A soft reset will result in closing the data and control connections from the initiating client to the server. If a job is simultaneously spooling and printing when the CR_SoftReset request is received, the initiating client's job shall be flushed from the server device, and the data and control connections established by the initiating client shall be closed by the server. The server device's buffers shall be flushed to get the device into a state where it is ready to act on a new job. For a printing device this includes flushing the currently printing page so that the next job received will start on a new clean page. For scanning-based server devices this would mean that any job still being spooled across the data connection to the client device would be lost.

Spooling is defined as actively downloading the print job to a printer or actively uploading the scan job from a scanner.

A soft reset will not clear the list of clients registered for notifications.

A soft reset will not affect any credit grant between a client and a server except for the initiating client, whose credit grants are nullified by the closing of the control and data connections.

A soft reset will result in closing only the initiating client's control and data connections.

A soft reset will only affect jobs being spooled from the client initiating the request. It is highly recommended that a client request a soft reset before attempting a hard reset.

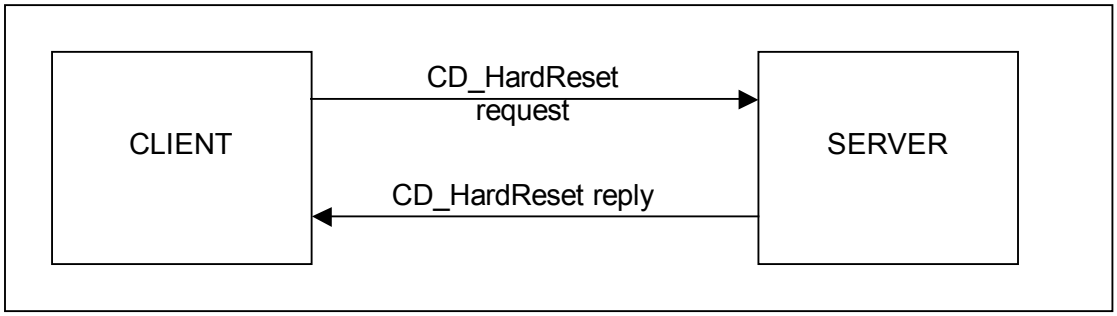
CR_SoftReset PDU Request

PDU Type	PDU ID	Request PDU Parameters
CR_SoftReset	0x0007	None

CR_SoftReset PDU Reply

PDU Type	PDU ID	Reply PDU Parameters
CR_SoftReset	0x0007	None

6.4.15 CR_HardReset Transaction



Description

The hard reset is used to return the server device to its original state after power-up. A hard reset will clear all clients' notification requests.

A hard reset will clear all outstanding credit grants to all clients. All connections will be closed. This also means that all outstanding granted credit is reset. Note that hard reset on a server which allows multiple clients to connect to it at once will result in all clients being affected by the hard reset, not just the client sending the hard reset request. Therefore, a hard reset is *not* recommended as a first recourse.

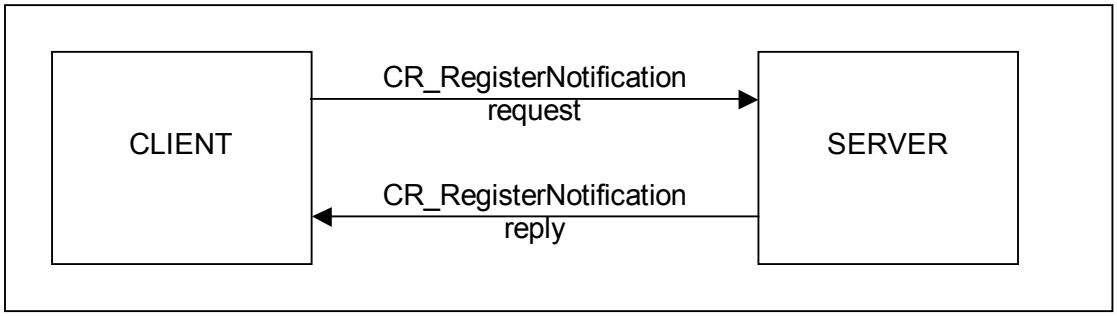
CR_HardReset PDU Request

PDU Type	PDU ID	Request PDU Parameters
CR_HardReset	0x0008	None

CR_HardReset PDU Reply

PDU Type	PDU ID	Reply PDU Parameters
CR_HardReset	0x0008	None

6.4.16 CR_RegisterNotification Transaction



Description

The register notification transaction is used to register the client device to receive notifications from the server device. The server device will be able to determine the address of the connection that it needs to connect back to. The client passes a unique context handle to the server that is used to identify the server when the server connects back to the client. The server also returns a timeout value after which the notification will expire. This ensures that the server will not attempt to connect to clients that register for notifications and are unavailable for connection, as the timeout will cause the notification request to expire.

When a server wishes to notify a client of an event, it connects back to the client on the reverse notification channel and passes the context handle to the client. This allows the client to uniquely identify the server context. When the client closes the connection, it implies that the client has processed the server's request.

The server shall keep the connection open for at least the length of the time negotiated between the client and server by the CallBackTimeOut value. During the negotiation, the client passes a requested value in the request PDU. The server returns the value that it shall honour in the reply PDU. The client's requested value is only a hint to the server. The client may attempt to keep the notification connection open longer by sending a CR_NotificationConnectionAlive PDU to the server. It is recommended that, if the server supports the CR_NotificationConnectionAlive PDU, the CallBackTimeOut value returned by the server exceeds the maximum time to make a Bluetooth connection.

Once a server has notified a client of an event, the client is unregistered for further events. The client should reregister after servicing an event.

A client may keep a current registration for events alive by reregistering for events prior to the TimeOut value expiring. A re-registration shall use the same CallbackContextID. The new TimeOut value returned represents the absolute TimeOut period before the registration expires.

A soft reset will not clear the clients' notification requests. A hard reset will clear the clients' notification requests. It is recommended that a server notify each client before completing the hard reset of the device to let the clients know that the notifications are going to be cancelled.

Refer to Section 6.6 for more information on notifications.

CR_RegisterNotification PDU Request

PDU Type	PDU ID	Request PDU Parameters
CR_RegisterNotification	0x0009	Register, CallbackContextID, CallBackTimeOut

PDU Parameters

Register *Type: Unit8* *Size: 1 Byte*

Value	Description
0x00	Remove this client from receiving notifications. The CallbackContextID parameter shall match the CallbackContextID that was used for the initial request to register for notifications.
0x01	Add this client to receive notifications.

CallbackContextID *Type: Unit32* *Size: 4 Bytes*

Value	Description
0xXXXXXXXX	A unique handle that identifies the context of this connection. This value is only relevant to the client that will be connected back to.

CallBackTimeOut *Type: Unit32* *Size: 4 Bytes*

Value	Description
0xXXXXXXXX	The time in milliseconds requested by the client that the server should keep the notification channel open to the client after connecting. The client's requested value is only a hint to the server. The client may attempt to keep the notification connection open longer by sending a CR_NotificationConnectionAlive PDU to the server.

CR_RegisterNotification PDU Reply

PDU Type	PDU ID	Reply PDU Parameters
CR_RegisterNotification	0x0009	TimeOut, CallBackTimeOut

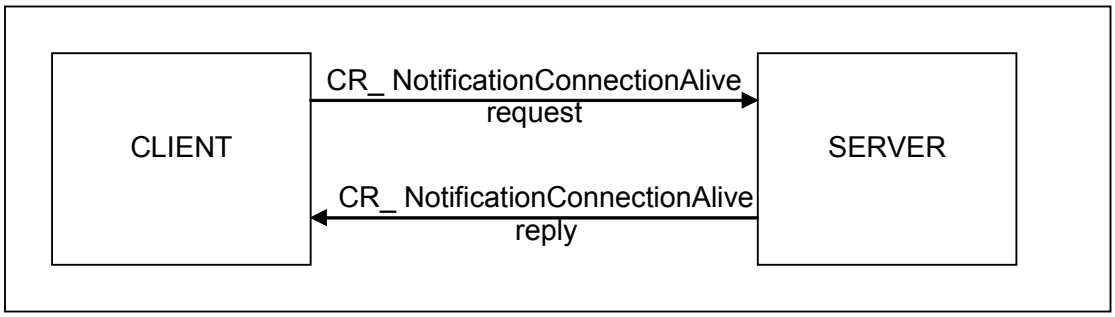
TimeOut *Type: Unit32* *Size: 4 Bytes*

Value	Description
0xXXXXXXXX	The time in milliseconds before the notification will expire. The server returns this value to the client so that it will know when to renew its subscription to events from the server.

CallBackTimeOut *Type: Unit32* *Size: 4 Bytes*

Value	**Description**
0xXXXXXXXX	The time in milliseconds that the server shall keep the notification channel open to the client after connecting. The client may attempt to keep the notification connection open longer by sending a CR_NotificationConnectionAlive PDU to the server.

6.4.17 CR_NotificationConnectionAlive Transaction



Description

The CR_NotificationConnectionAlive transaction is used to keep the notification connection open for longer than the negotiated CallBackTimeOut. The CallBackTimeOut that was negotiated during the CR_RegisterNotification transaction is extended by the returned value, TimeOutIncrement.

When either the client or server closes the notification channel it is assumed that the client has completed acting on the notification. This PDU is used by the client to notify the server that it has not finished processing the notification in the CallBackTimeOut allocated. The server is under no obligation to honor the request.

Refer to Section 6.6 for more information on notifications.

CR_NotificationConnectionAlive PDU Request

PDU Type	PDU ID	Request PDU Parameters
CR_NotificationConnectionAlive	0x000A	None

CR_NotificationConnectionAlive PDU Reply

PDU Type	PDU ID	Reply PDU Parameters
CR_NotificationConnectionAlive	0x000A	TimeOutIncrement

PDU Parameters

TimeOut *Type: Unit32* *Size: 4 Bytes*

Value	Description
0xXXXXXXXX	The time in milliseconds that the server has increased the timeout to close the notification connection.

6.5 Data Channel Protocol

This specification does not define the data that is transferred in the data channel. This is implementation-specific.

6.6 Notification Channel Protocol

The notification channel defines a format for PDUs as well as one PDU that may be transferred on the notification channel. All specific events sent on the notification channel and the specific client responses to them are out of scope of this profile.

A client may register to receive notifications from a server. This means that the server will asynchronously connect to the client to inform the client that the server has information for the client to receive. Notifications expire after the specified registration time (see Section 6.4.15). It is the client's responsibility to renew the notification registration. The client will not be notified in any way that the notification registration is expiring or has already expired.

Notifications are OPTIONAL to support for the client and server, but if a client makes a CR_RegisterNotification transaction it shall fully support notifications. That is, it shall have an SDP record to allow the server to connect back to the client as described in Section 7.2.

6.6.1 Notification Connections

When a server needs to notify a client (e.g., if the scan button is pressed on a multifunction printer), the server will connect to the client on the client's notification channel. This is equivalent to the client receiving an interrupt on a local bus. The client will have a single notification channel for each supported function for which it can receive notifications. The client's notification channel is discovered through the SDP. (See Section 7.2 for the notification service discovery record.) This is done to avoid having two different notification methods – one for currently connected clients and one for disconnected clients. When a server wishes to notify a client, it will always connect to the client on its notification connection. Once a server has notified a client, the client is de-registered automatically from receiving any further notifications from that server. The client may re-register if it wants to receive further notifications. This does have the potential of missing notifications; but in conjunction with a client's capability to query a server's current state, this can be solved easily.

The closing of the notification connection indicates that the notification has been processed. The client may close the connection at any time. The server may close the connection if the negotiated timeout value has expired as defined in Section 6.4.15 and Section 6.4.16.

Power cycling a server device will clear all notification subscriptions. For this reason it is highly recommended that servers not grant long subscription timeouts to minimize the time period during which a client could miss notifications.

6.6.2 Notification Failures

If a server fails in its first attempt to notify a client, the server shall retry the notification at least two times (for a total of at least three notification attempts). If multiple clients are simultaneously registered for notifications, the server shall attempt to notify all registered clients before retrying any client. There shall be at most a one-second interval between a failed notification attempt and the next notification attempt (which may or may not be an attempt to notify the same client).

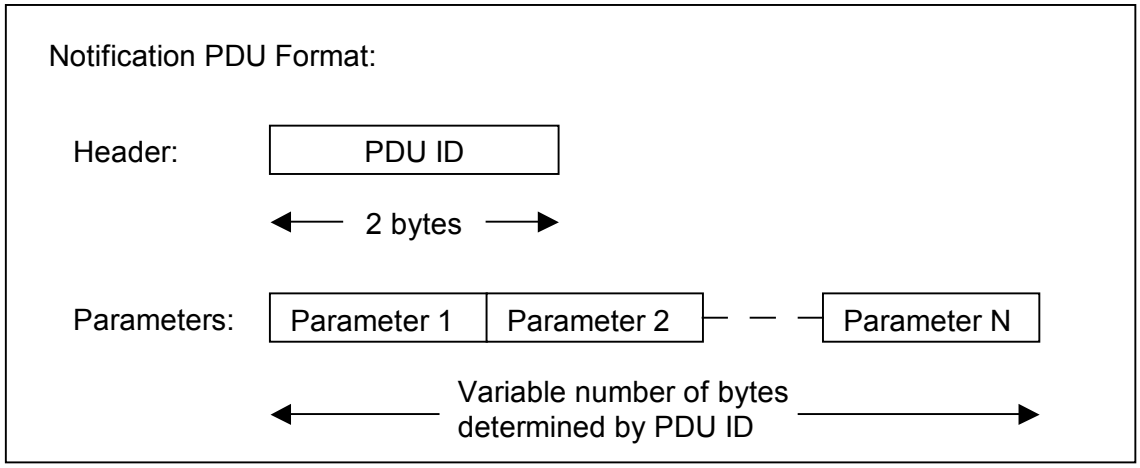
Note that a connection attempt may take as long as the timeout period of the connection request (which could be as long as fifteen seconds). Only after the connection request has failed will the one-second interval begin.

If the retry period exceeds the notification subscription period, the server shall retry up until the point that the subscription ends. A server may retry the notification connection more frequently or for a longer period of time than defined here.

If a server fails to notify a client after at least the minimum number of retries, the client misses the event on which it was to be notified. The server will, however, continue to attempt to connect to the client for any subsequent notifications until such time as the client's registration for notifications expires or the server successfully connects to the client and hence de-registers the client automatically.

6.6.3 Protocol Data Unit Format

Each notification consists of a PDU from the server to the client and an optional response from the client.

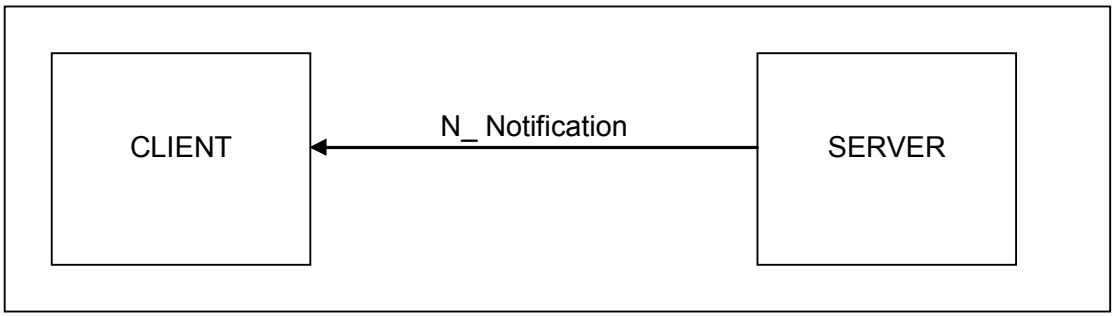


6.6.4 PDU Identifiers

Notification PDU ID Values

Value	Description	Client Status	Server Status
N	The PDU ID field identifies the type of PDU (i.e., its meaning and that of its parameters).	Whether the PDU is MANDATORY or OPTIONAL to implement on the client	Whether the PDU is MANDATORY or OPTIONAL to implement on the server
0x0000	Reserved	N/A	N/A
0x0001	N_Notification	OPTIONAL	OPTIONAL
0x0002-0x7FFF	Reserved	N/A	N/A
0x8000-0xFFFF	Vendor specific	OPTIONAL	OPTIONAL

6.6.5 N_Notification Transaction



The creation of the notification connection with an N_ Notification PDU by the server indicates to the client that the server has change of status information to deliver. The context ID that was passed in the register notification control request will be read out of the notification connection to uniquely identify the server context from which the client is receiving notification information. The context ID will be the first four bytes (in big-endian order) sent on the notification channel after the PDU ID.

The response to the N_Notification notification connection is implementation-specific and is defined by how the server expects the notification to be handled. The server may require the client to connect back to the server (if it is not already connected) so that the client may make vendor-specific requests down the control channel in order to respond to the notification.

There is no reply expected for this request; instead the channel will be closed by the client.

N_Notification PDU Request

PDU Type	PDU ID	Parameters
N_Notification	0x0001	CallbackContextID

Parameters

CallbackContextID *Type: Unit32* *Size: 4 Bytes*

Value	Description
0xXXXXXXXX	A unique handle that identifies the context that this connection from the server is connecting from. This was previously provided by the client in the CR_RegisterNotification control transaction (see Section 6.4.16).

7 Service Discovery

The server shall provide a Service Discovery Record describing its Control and Data Channels. The client shall provide a Service Discovery Record if it will register for notifications.

7.1 Control and Data Channel Service Discovery Record

Table 4 lists the entries in the SD record that are defined by this profile. The "Status" column indicates whether the presence of each field is mandatory or optional. This is used to make the control and data connections from the client to the server device.

Item	Definition	Type Size	Value**	AttrID	Status	Default Value
Service Class ID List				See [6]	M	
Service Class #0		UUID	HCR_Print OR HCR_Scan***	See [6]	M	
ServiceID****	See [4], 5.1.4	UUID	Varies	See [6]	O	
Protocol Descriptor List				See [6]	M	
Protocol ID #0		UUID	L2CAP		M	
Param #0	PSM	Uint16	Varies*****		M	
Protocol ID #1	Connection type	UUID	Hardcopy-ControlChannel	See [6]	M	
Service Name	Displayable text name	String	Configurable	See [6]	O	See 7.3.1
Bluetooth Profile Descriptor List				See [6]	O	
Profile ID #0	Supported profile	UUID	HardcopyCable Replacement			Hardcopy Cable-Replace-ment [6]
Version #0	Profile version	Uint16	Varies			0x0100
Additional Protocol Descriptor Lists				See [6]	M	
Protocol Descriptor List #0						
Protocol ID #0		UUID	L2CAP			
Param #0	PSM	Uint16	Varies*****			
Protocol ID #1	Connection type	UUID	HardcopyData-Channel	See [6]		
1284ID*	IEEE 1284 ID	String	Model-specific IEEE 1284 ID string; See [8]	0x0300	O	See 7.3.2
Device Name	Displayable text name	String	Model specific display name	0x0302	O	See 7.3.3
Friendly Name	Displayable text name	String	Device-specific friendly name	0x0304	O	See 7.3.4
Device Location	Physical Device Location	String	Varies	0x0306	O	See Section 7.3.5

Table 4: Control and Data Service Discovery Record (Server)

* The “1284ID” entry can be queried by the CR_Get1284ID transaction and also from the control and data SD records. This ensures that a client driver can be identified and loaded for the server device without the client actually having to connect to the server device. Furthermore, dongles that don’t necessarily know what they are attached to do not have to implement this entry and can thus force the client to connect and query the server device directly.

** All values that are of the type UUID are defined in the Assigned Numbers Specification [6].

*** The separate Service Class implies that there will be a separate control SD record for each function made available by the server. This means that there will be a unique control channel for each function exposed.

**** The ServiceID entry is used to uniquely identify a single control and data SD record if more than one control and data SD record are exposed by a single SDP server. An example of where this can be used is if a device exposes two or more different physical printers as HCRP printers from the same device.

***** Valid PSM values are defined in [3]. Each implementation is responsible for ensuring that the PSMs are unique within their system. The PSMs for HCRP are not defined specifically; they are implementation-specific. These are the only requirements for the PSM values used in the HCRP SDP record(s). In cases where there are multiple HCRP service records exposed, the PSM values shall be different for each record.

7.2 Notification Service Discovery Record

The Notification SD record shall be supported by the client device if the client registers for notifications through CR_RegisterNotification. This service discovery record allows the server device to find the connection on which to connect asynchronously back to the client.

If the client does not register for notifications, it does not have to provide this SD record.

Item	Definition	Type Size	Value*	AttrID	Status	Default Value
Service Class ID List				See [6]	M	
Service Class #0		UUID	HCR_Print OR HCR_Scan**	See [6]	M	
Protocol Descriptor List				See [6]	M	
Protocol ID #0		UUID	L2CAP		M	
Param #0	PSM	Uint16	Varies*****		M	
Protocol ID #1	Connection type	UUID	HardcopyNotifi-cationChannel	See [6]	M	
Service Name	Displayable text name	String	Varies	See [6]	O	See 7.3.1
Bluetooth Profile Descriptor List				See [6]	O	
Profile ID #0	Supported profile	UUID	HardcopyCable Replacement			Hardcopy-CableRe-placement [6]
Version #0	Profile version	Uint16	Varies			0x0100

Table 5: Notification Service Discovery Record (Client)

* Values that are of the type UUID are defined in the Assigned Numbers Specification [6].

** The separate Service Class implies that there will be a separate notification SD record for each function utilized by the server. This means that there will be a unique notification channel for each function for which the client will register for notification.

***** Valid PSM values are defined in [3]. Each implementation is responsible for ensuring that the PSMs are unique within their system. The PSMs for HCRP are not defined specifically; they are implementation-specific. These are the only requirements for the PSM values used in the HCRP SDP record(s). In cases where there are multiple HCRP service records exposed, the PSM values shall be different for each record.

7.3 Service Record Attribute Details

7.3.1 Service Name

The Service Name string provides a displayable text name that can be directly transmitted to the user. A suggested value for devices that include this optional string and choose not to localize is “Hardcopy Cable Replacement”.

Maximum length of the Service Name attribute is 248 bytes; strings should not be null terminated.

Note that the Service Name attribute as defined in [6] is a SDP attribute ID offset. Thus, if Service name is included in the SDP record then a LanguageBaseAttributeIDList attribute must also be present.

7.3.2 1284ID

The 1284ID shall be as defined in [8].

The string shall be an exact copy of the string defined in [8] including the initial 2-byte length field.

The length of the string can be up to 65535 bytes as defined in [8].

7.3.3 Device Name

The Device Name string provides a displayable text name that can be directly transmitted to the user.

Maximum length of the Device Name attribute is 248 bytes; strings should not be null-terminated.

The Device Name is UTF-8 encoded.

The default Device Name should be chosen by the manufacturer; it is not expected that the Device Name is configurable by the user.

7.3.4 Friendly Name

The Friendly Name string provides a displayable text name that can be directly transmitted to the user.

Maximum length of the Friendly Name attribute is 248 bytes; strings should not be null-terminated.

The Friendly Name is UTF-8 encoded.

The default Friendly Name should be set to the same as Device name; the Friendly Name may be configurable by the user.

7.3.5 Device Location

The Device Location string provides a displayable text name that can be directly transmitted to the user.

Maximum length of the Device Location attribute is 248 bytes; strings should not be null-terminated.

The Device Location is UTF-8 encoded.

The default Device Location should be set to empty string; it is intended that the Device Location may be user-configurable.

8 Link Manager

8.1 Authentication, Encryption, and Bonding

Link-level authentication and encryption are mandatory to support and optional to use.

If the server device initiates authentication, it may do so by using a fixed PIN. The PIN could be changed using some proprietary method in the server device. The use of a fixed PIN is explained in Section 14.2.1 of [1]. How the fixed PIN is communicated to the (user of the) client device is not specified in the Hardcopy Cable Replacement profile.

Neither the client nor the server is required to be able to initiate authentication, but all devices shall be able to respond to an authentication request.

It is recommended that client devices without user interfaces avoid initiating link-level authentication. If the server initiates link-level authentication, interoperability cannot be guaranteed with devices without a user interface. Therefore it is recommended that, if turned on, link-level authentication can be turned off in the server.

8.2 Session Disconnection

To avoid blocking subsequent connections to the server by other clients, it is recommended that the client fully disconnect from the server following completion of a print or scan session, or after cancelling a job. This disconnection should include the L2CAP channels and their supporting ACL channel.

How the disconnection is managed is beyond the scope of this specification and may be dependent on other applications running concurrently between the client and server.

9 Normative References

[1] Bluetooth Core Specification, Baseband Specification

[2] Bluetooth Core Specification, LMP Specification

[3] Bluetooth Core Specification, L2CAP Specification

[4] Bluetooth Core Specification, SDP Specification

[5] Bluetooth Profile Specification, Generic Access Profile Specification

[6] Bluetooth Profile Specification, Assigned Numbers Specification
http://www.bluetooth.org/assigned-numbers.htm

[7] Bluetooth Core Specification, Appendix IX

[8] IEEE Standard Signaling Method for a Bidirectional Parallel Peripheral Interface for Personal Computers IEEE Std. 1284-2000

[9] Bluetooth Basic Printing Profile (title and reference TBD)

[10] Bluetooth Basic Imaging Profile Specification

[11] IEEE Standards Style Manual,
http://standards.ieee.org/guides/style/2000Style.pdf

10 Acronyms and Abbreviations

Abbreviation or Acronym	Meaning
BB	Baseband
CoD	Class of Device/Service
GAP	Generic Access Profile
HCRP	Hardcopy Cable Replacement Profile
L2CAP	Logical Link and Control Adaptation Protocol
LC	Link Controller
LM	Link Manager
LMP	Link Manager Protocol
MTU	Maximum Transmission Unit
OSI	Open System Interface
PDA	Personal Digital Assistant
PDL	Page Description Language
PDU	Protocol Data Unit
PIN	Personal Identification Number
PSM	Protocol/Service Multiplexer
RFCOMM	Serial Cable Emulation Protocol
SD	Service Discovery
SDDB	Service Discovery Database
SDP	Service Discovery Protocol
UI	User Interface
UUID	Universal Unique Identifier

11 Appendix A

11.1 Bluetooth General and Device-Specific Inquiry

The Class of Device/Service (CoD) field facilitates discovery of relevant Bluetooth devices via Bluetooth Inquiry mechanisms described in [5].

A Server Device that supports the Hardcopy Cable Replacement Profile shall set the "Rendering" and/or “Capturing” Major Service Class bit, as appropriate. These bits are described in the Bluetooth Assigned Numbers Document [6]. Other Major Service Class bits may be set as appropriate.

A Server Device that supports the Hardcopy Cable Replacement Profile should set the Major Device Class to "Imaging" as specified in [6]. Note that a Server Device that supports HCRP in addition to other Bluetooth Profiles may advertize a different Major Device Class than "Imaging". For instance, a device with a print proxy function might present a Major Device Class of "Computer" or "LAN/Network access point". Such a device shall be detected by Client Devices that are looking for available HCRP Server Devices; i.e., a Client Device shall not filter only for devices that have "Imaging" as their Major Device Class.

If the "Imaging" Major Device Class is advertized, the "Printer" and/or “Scanner” bits shall be set in the Minor Device Class field. Other Minor Device Class bits may be set as appropriate.

HCRP does not require any particular bits to be set in the CoD field of a Client Device.

The following is an example of an algorithm that makes use of the CoD field to locate Server Devices supporting HCRP for Printing:

1 Is the "Rendering" bit set in the Major Service Class field? If not, ignore this Bluetooth device.

2 If the Major Device Class is "Imaging", is the "Printer" bit set in the Minor Device Class field? If not, ignore this Bluetooth device.

3 Request the SDP records from the device and see if the UUID "HCR_Print" is present in the Service Class ID List.

If so, the device supports Printing via HCRP.

Similar algorithms can be constructed to find Scanners or Printer/Scanner multi-function devices that support HCRP.