Chapter 1

Look Inside

Introduction

1.1 Introduction

This is the specification for a USB note acceptor as defined by the GDS within the GSA. For the purposes of this document the terms “USB note acceptor”, “note acceptor”, “GDS device” and “device” may be used interchangeably. The aim of the GDS is to develop true plug ‘n’ play peripherals for the gaming environment which are already common place in the computer industry. Rather than use a derivative of RS232 it was decided to move the technology forward and adopt the USB standard.

This document specifies the complete set of functionality for the GDS Note Acceptor.

1.2 Note Acceptor Function Overview

Commands supported by the note acceptor are defined in Chapter 3. Supported events are defined in Chapter 4. Common commands and events are identified in Chapter 2.

The note acceptor on power-up must default to escrow mode, be disabled, run a diagnostic test, and report events as necessary.

The note acceptor stores accepted note data, even over power failure intervals, until the acknowledgment of the note data transaction is received from the host.

The interrupt service period, bInterval, is 100ms.

1.3 USB Compliance and Benefits

USB is currently available in three speeds:

- low speed: 1.5 Mbits/sec.
- full speed: 12Mbits/sec.
- high speed: 480Mbits/sec.

GDS peripherals MUST, at a minimum, support **full speed** and MUST comply with all published USB requirements.

Only one master exists on the USB bus and that is the host machine. All peripherals MUST be connected through hubs (up to 127 allowed but this is likely to be less than 10). For security reasons, it is undesirable to have a direct peripheral-to-peripheral communication path. Therefore, OTG (On- The-Go) extensions that allow peripherals to talk directly to one another MUST NOT be used; all communication MUST be via the host.

1.4 Note Acceptors as HID Class Devices

To remove the need for special manufacturer-specific device drivers, the note acceptor MUST be implemented as an HID—Human Interface Device—class device (see document reference DCDHID). This is the same class as mice, keyboards and joysticks but is a flexible enough to be used for general input/output data packets

where the data sizes are relatively small and the transfer rates low. HID implements usage pages and usage numbers to allow access to data in an abstracted fashion, regardless of the manufacturer or even the exact location within a report of where the data can be found.

This document assumes hardware and firmware are in place to support a HID class device within USB and so need only concern itself with HID commands and data formats for gaming. Product identification is also discussed.

1.5 Device Firmware Upgrade (DFU)

GDS devices MUST use the Device Firmware Upgrade (DFU) standard, version 1.1, as defined by USB-IF, http://www.usb.org/developers/docs/devclass_docs/DFU_1.1.pdf as the method of upgrading device firmware.

The use of the DFU class MUST NOT affect the behavior of the HID class.

Upon firmware upgrade the device MUST clear its internal memory including memory involving Transaction IDs.

The serial number exposed through the device descriptor in DFU and GDS modes MUST be identical.

Support for firmware download is REQUIRED; support for firmware upload is OPTIONAL.

Chapter 2

Look Inside

Command Support

2.1 Command Support

The following note acceptor commands MUST be supported and implemented as HID Feature reports.

Table 2.1 Supported Note Acceptor Commands

Report ID	Usage ID	Name	Data	
0x01	0x40	ACK	No	Page 7
0x02	0x41	Enable	No	Page 9
0x03	0x42	Disable	No	Page 9
0x04	0x43	Self Test	No	Page 9
0x05	0x44	Request GAT Report	No	Page 10
0x08	0x47	Calculate CRC	Yes	Page 10
0x80	0x0210	Number of Note Data Entries	No	Page 11
0x81	0x0211	Read Note Table	No	Page 11
0x82	0x0212	Extend Timeout	No	Page 12
0x83	0x0213	Accept Note/Ticket	No	Page 12
0x84	0x0214	Return Note/Ticket	No	Page 12
Extension in v1.2				
0x8A	0x021A	Read Note Acceptor Metrics	No	Page 13

NOTE:
The following reports are *diagnostic commands*: Report 0x04 Self Test, Report 0x05 Request GAT Report, Report 0x08 Calculate CRC, Report 0x80 Number of Note Data Entries, and Report 0x81 Read Note Table.

2.2 Command Execution When Enabled/Disabled

The following table shows the device state a given command can be performed under. Any command sent to the note acceptor while it is in the wrong state MUST be ignored. Unless specified differently elsewhere, any command sent to the note acceptor while it is in the correct state MUST NOT be ignored.

Table 2.2 Command Execution When Enabled/Disabled

Command	Operation When Device Enabled	Operation When Device Disabled
0x02 Enable	Yes	Yes
0x03 Disable	Yes	Yes
0x04 Self Test	No	Yes
0x05 Request GAT Report	No	Yes
0x08 Calculate CRC	No	Yes
0x80 Number of Note Data Entries	No	Yes
0x81 Read Note Table	No	Yes
0x82 Extend Timeout	Yes	No
0x83 Accept Note/Ticket	Yes	No
0x84 Return Note/Ticket	Yes	No
Extension in v1.2		
0x8A Read Note Acceptor Metrics	No	Yes

2.3 Report 0x01 ACK

This is a pseudo-command as it is sent in response to a Transaction ID event from the note acceptor as part of the handshake sequence. This command is used to confirm critical events such as Note Validated, Ticket Validated, Note/Ticket Status and Stacker Status events.

The Note Validated event, Ticket Validated event, Note/Ticket Status event and Stacker Status event share the same Transaction ID sequence, and are referred to as note acceptor Transaction ID, or TID, events.

A Transaction ID stamp is used to ensure that every note acceptor TID event is properly handled by the host and only acknowledged events are removed from the note acceptor queue. If the device does not receive an ACK from the host within one (1) second then the event MUST be resent to the host.

The Transaction ID protocols are only used with TIDs. If power is removed before all of the note acceptor TID events have been completely communicated with the host, they MUST be re-issued by the acceptor when power is restored and enabled. Therefore, any un-sent or pending note acceptor TIDs MUST be buffered in NVM.

2.3.1 Transaction ID Rules

1. The Transaction ID sequence is common to events that contain a Transaction ID. The Transaction ID MUST be incremented sequentially each time one of these events occurs and is acknowledged. For example, a 0x86 Note Validated event (see Page 30) with a Transaction ID of 100 would be followed by a 0x88 Note/Ticket Status event (see Page 32) with a Transaction ID of 101.
2. Transaction IDs MUST increment through the range 0 to 255.
3. Transaction IDs MUST NOT be cleared from the note acceptor stack except:
    a. After the host has acknowledged the event and the Transaction ID rules have been met whereby both the host and device Transaction IDs are in sync.
    b. Upon a firmware upgrade of the device (DFU). In this case all Transaction IDs MUST be cleared.
4. The current Transaction ID MUST be stored in the device's NVM so as to provide a reference number in the event of a power interruption.
5. If the device does not receive an acknowledge from the host within one (1) second after sending a critical event, the device MUST retry sending the event every one (1) second until the host properly acknowledges the event and the Transaction IDs are in sync.
6. While waiting for an ACK, a device MUST NOT NAK, at the USB level, a command from the host.
7. When the Resync bit is set in an ACK command, the device MUST reset the current Transaction ID to the value specified by the host in the ACK command. The pending event MUST then be retried using the new the Transaction ID. The Transaction ID MUST be incremented sequentially from that new Transaction ID. For example, if an event is sent with Transaction ID 10 and the host resyncs the Transaction ID to 20, the event will be retried with Transaction ID 20 and the next event will be sent with Transaction ID 21.

Table 2.3 0x01 ACK Structure

Bit	7	6	5	4	3	2	1	0
Byte 0	**0x01**							
Byte 1	-	-	-	-	-	-	-	Resync
Byte 2	Transaction ID							

Table 2.4 0x01 Field Descriptions

Name	Value	Description
Resync	0	The device MUST NOT re-sync its Transaction ID. This is an acknowledgement.
	1	The device MUST re-sync its Transaction ID and resend the pending report.
Transaction ID	0 to 0xFF	The confirmed or new Transaction ID. See Section 2.3.1.

2.4 Report 0x02 Enable

The device MUST activate any inputs and/or enable any outputs. If the device has a fault then it MUST stay disabled and report the fault to the host.

The host issues this command to allow notes to be drawn into the note acceptor for the purposes of validation.

Animation lamps are enabled.

If the device has no faults, after enabling itself, the device MUST send the 0x0A Device State event with the Enable bit set (see Page 26).

Table 2.5 0x02 Enable Structure

Bit	7	6	5	4	3	2	1	0
Byte 0	**0x02**							

2.5 Report 0x03 Disable

The device MUST deactivate any inputs and/or disable any outputs. The host issues this command when it does not want to allow notes/tickets to be drawn into the note acceptor. If a note/ticket is in escrow when this command is received, the device MUST return the note/ticket immediately after disabling itself; the device MUST NOT wait for the timeout period for the note/ticket to expire before returning the note/ticket.

The host MUST send a Disable command before issuing any diagnostic commands (defined on Page 61).

Animation lamps are disabled; however, any lamps that indicate that the device is out of service are not affected.

When the device is in the enabled state and receives a Disable command, the device MUST immediately disable itself and respond with the 0x0A Device State event (see Page 26).

Table 2.6 0x03 Disable Structure

Bit	7	6	5	4	3	2	1	0
Byte 0	**0x03**							

2.6 Report 0x04 Self Test

The device MUST initiate a self-test sequence when instructed by the host. When the host requests a self-test, the device MUST respond with a 0x85 Failure Status event (see Page 29). The device MUST complete all tests before sending the Failure Status event to the host. Multiple failures MUST be presented in the final single

Failure Status event, with the exception of multiple 'Other' failures, which MUST be reported in separate Failure Status events.

Table 2.7 0x04 Self Test Structure

Bit	7	6	5	4	3	2	1	0
Byte 0	**0x04**							
Byte 1	-	-	-	-	-	-	-	NVM

Table 2.8 0x04 Field Descriptions

Name	Value	Description
NVM	0	Perform self-test.
	1	Non-volatile Memory (NVM), previously queued Transaction ID events and Transaction ID sequence number, MUST be cleared before the self-test is performed.

2.7 Report 0x05 Request GAT Report

The host sends this command to request information from the note acceptor via a 0x07 GAT Data event (see Page 24). The device is not required to monitor itself for fault conditions during the performance of this command. The 0x07 GAT Data response MUST be in ASCII format. The following characters MUST NOT be used in the response data: '<' or '/>'. The note acceptor does not output in XML.

At this time, the serial and network-based GAT protocols, such as GAT v3.50, G2S v1.1, and S2S 1.5, do not require the use of this command. Authentication of peripheral devices is performed using the 32-bit CRC algorithm. This command is simply a placeholder. The GAT data in the response is manufacturer-specific in terms of length and content. The response MAY include no data bytes.

Table 2.9 0x05 Request GAT Report Structure

Bit	7	6	5	4	3	2	1	0
Byte 0	**0x05**							

2.8 Report 0x08 Calculate CRC

The host can request a 32-bit checksum from the code space within the note acceptor's ROM memory, excluding the USB Identification Strings (see Page 50). For additional accuracy and security, a 32 bit seed is sent to the device as a parameter of this command. The host may choose any seed value and compare the result with a look-up table or a similar calculation based on an exact reference copy of the code to be verified.

A checksum report is returned in the 0x09 CRC Data event (see Page 26) with the result.

While calculating the CRC the device is not required to receive any additional commands. The device is permitted to NAK_OUT all requests from the host while performing the CRC calculation. The device is also not required to monitor itself for fault conditions during the performance of this command.

See Page 63 for the algorithm used.

Table 2.10 0x08 Calculate CRC Structure

Bit	7	6	5	4	3	2	1	0
Byte 0	**0x08**							
Byte 1	Seed 0							
Byte 2	Seed 1							
Byte 3	Seed 2							
Byte 4	Seed 3							

Table 2.11 0x08 Field Descriptions

Name	Value	Description
Seed	0 to 255	The starting seed for the CRC-32 calculation. Seed 0 is the LSB.

2.9 Report 0x80 Number of Note Data Entries

The host issues this command to first determine how many note data entries are expected to be read or written when an upgrade or 0x81 Read Note Table command (see Page 11) is executed.

Table 2.12 0x80 Number of Note Data Entries Structure

Bit	7	6	5	4	3	2	1	0
Byte 0	**0x80**							

2.10 Report 0x81 Read Note Table

The host issues this command to get the list of notes supported by the note acceptor. If note data is upgraded, the host must re-issue this command as the number of entries and/or value associated with a given Note ID (see Page 28) may have changed.

Table 2.13 0x81 Read Note Table Structure

Bit	7	6	5	4	3	2	1	0
Byte 0	**0x81**							

2.11 Report 0x82 Extend Timeout

The host issues this command in response to either 0x86 Note Validated event (see Page 30) or 0x87 Ticket Validated event (see Page 31) when it needs to retain a note/ticket in escrow for a longer period. The Extend Timeout command resets the timeout period to five (5) seconds.

Table 2.14 0x82 Extend Timeout Structure

Bit	7	6	5	4	3	2	1	0
Byte 0	**0x82**							

2.12 Report 0x83 Accept Note/Ticket

The host issues this command in response to either 0x86 Note Validated event (see Page 30) or 0x87 Ticket Validated event (see Page 31) when it wants to accept and stack a note/ticket. The host must not credit the note/ticket value until a 0x88 Note/Ticket Status event (see Page 32) is received from the device with the Accepted bit set.

The device MUST automatically return the note/ticket if it does not receive a command to accept or return the note/ticket from the host within five (5) seconds from sending the 0x86 Note Validated or 0x87 Ticket Validated event and the timeout has not been extended. A 0x88 Note/Ticket Status event, (see Page 32) MUST be issued by the device if a note/ticket is returned due to lack of host response.

The host MAY extend the timeout period by issuing 0x82 Extend Timeout command (see Page 12) before the device's timeout period expires. The host MAY continue to extend the device's timeout period indefinitely.

Table 2.15 0x83 Accept Note/Ticket Structure

Bit	7	6	5	4	3	2	1	0
Byte 0	**0x83**							

2.13 Report 0x84 Return Note/Ticket

The host issues this command in response to either 0x86 Note Validated event (see Page 30) or 0x87 Ticket Validated event (see Page 31) when it wants to return the note/ticket. The device MUST automatically return the note/ticket if it does not receive a command to accept or return the note/ticket from the host within five (5) seconds from sending the 0x86 Note Validated or 0x87 Ticket Validated event and the timeout has not been extended. After returning the note/ticket, the device MUST generate the 0x88 Note/Ticket Status event (see Page 32) with the Returned bit set for confirmation. See Section 4.17.2.1, Note Acceptor Automatic Reject, for special requirements that apply if the host instructed the note acceptor to reject the note/ticket but the note/ticket was not returned before a power cycle or system reset occurred.

The host MAY extend the timeout period by issuing 0x82 Extend Timeout command (see Page 12) before the device's timeout period expires. The host MAY continue to extend the device's timeout period indefinitely.

Table 2.16 0x84 Return Note/Ticket Structure

Bit	7	6	5	4	3	2	1	0
Byte 0	**0x84**							

2.14 Report 0x8A Read Note Acceptor Metrics

Extension in v1.2

The host issues this command to request the note acceptor's capability metrics, such as types of barcodes supported and UTF-16 support. The note acceptor sends the 0x8A Read Note Acceptor Metrics event (Page 35) in response.

Table 2.17 0x8A Read Note Acceptor Metrics Structure

Bit	7	6	5	4	3	2	1	0
Byte 0	**0x8A**							