AUTHOR

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Department** |  | **Signature** |  | **Print Name** |  | **Date** |
| R&D |  |  |  |  |  |  |

SIGNATURES

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Department** |  | **Signature** |  | **Print Name** |  | **Date** |
| R&D |  |  |  |  |  |  |

Revision History

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Rev.** | | **Date** | **Initials** | **Description** |
| A | 18 June 2019 | | DAT | Initial Release |

Glossary

|  |  |
| --- | --- |
| ***BDM*** | *Bridge Device Manager, the software component that interfaces directly with the capital device. BDM is responsible for synchronization of all data and control between the Capital Device and the individual INTELLIO Connect Protocols.* |
| ***Capital Device*** | *Resection, Coblation, Fluid Management or other devices that require an INTELLIO Link (in the context of this document) to be part of INTELLIO Connect.* |
| ***INTELLIO Connect*** | *A system of connected nodes on a common WIFI network that conform to the INTELLIO Connect Protocols.* |
| **SCD** | Shaver Capital Device |
| **Handpiece** | An MDU, Drill or Saw that is plugged into Port A or Port B of the SCD. |
| **Timeout (TO)** | A condition where an expected result does not happen in a predetermined amount of time. |

References

15007715 ***INTELLIO Connect Discovery Protocol***

15006656 ***INTELLIO Connect GETSET Protocol***

15007717 ***INTELLIO Connect Setup Blob Protocol***

15007718 ***INTELLIO Connect Device Blob Protocol***

15007716 ***INTELLIO Connect OSD Manager Protocol***

15000695 ***DYONICS POWER II System Controller Software Requirements Specification***

15007726 ***DYONICS POWER II INTELLIO Link SRS***

Table of Contents

[1 Overview 4](#_Toc12535515)

[2 System Overview 4](#_Toc12535516)

[3 Goals and assumptions 5](#_Toc12535517)

[4 Network Message Definition 6](#_Toc12535518)

[4.1 Physical Network Message 6](#_Toc12535519)

[4.2 Message Reception 7](#_Toc12535520)

[4.2.1 Protocol ID 7](#_Toc12535521)

[4.2.2 Command 8](#_Toc12535522)

[4.2.3 Request Number 8](#_Toc12535523)

[4.2.4 Command Data Length 8](#_Toc12535524)

[4.2.5 Command Data 8](#_Toc12535525)

[4.2.6 Checksum 8](#_Toc12535526)

[4.2.7 Message End 9](#_Toc12535527)

[4.3 Running 2’s Complement Checksum Algorithm 9](#_Toc12535528)

[4.4 Message Retry Handling 9](#_Toc12535529)

[4.5 Physical connection 10](#_Toc12535530)

[5 Detailed NAK 10](#_Toc12535531)

[5.1 Detailed NAK DETAILED\_NAK (BDM->SCD) 10](#_Toc12535532)

[5.2 Detailed NAK DETAILED\_NAK (SCD->BDM) 10](#_Toc12535533)

[6 Capital Device Discovery 11](#_Toc12535534)

[6.1 Discovery Request DR\_MSG (BDM->SCD) 11](#_Toc12535535)

[6.2 Discovery Request Reply DR\_MSG\_RPLY (SCD->BDM) 12](#_Toc12535536)

[7 Heart Beat 12](#_Toc12535537)

[7.1 Heart Beat Status HB\_MSG (BDM->SCD) 12](#_Toc12535538)

[7.2 Heart Beat Status Reply HB\_MSG\_RPLY (SCD->BDM) 13](#_Toc12535539)

[8 Shaver Status Messages 13](#_Toc12535540)

[8.1 Port Status PORT\_STATUS\_MSG (SCD->BDM) 14](#_Toc12535541)

[8.1.1 SCD Errors and Warnings 16](#_Toc12535542)

[8.2 Port Status Reply PORT\_STATUS\_MSG\_RPLY (BDM->SCD) 17](#_Toc12535543)

[8.3 Get Port Status GET\_PORT\_STATUS\_MSG (BDM->SCD) 17](#_Toc12535544)

[9 Setting Device Specific Information 17](#_Toc12535545)

[9.1 Set Device Info SET\_DEVICE\_INFO\_MSG (BDM->SCD) 17](#_Toc12535546)

[9.2 Set Device Info Reply SET\_DEV\_INFO\_MSG\_RPLY (SCD->BDM) 18](#_Toc12535547)

[10 Lavage Toggle Event Function (SCD->BDM) 18](#_Toc12535548)

[10.1 Lavage Toggle Event LAVAGE\_TOGGLE\_EVENT\_MSG (SCD->BDM) 18](#_Toc12535549)

[10.2 Lavage Toggle Reply LAVAGE\_TOGGLE\_MSG\_RPLY (BDM->SCD) 18](#_Toc12535550)

[11 SCD Command Function 19](#_Toc12535551)

[11.1 SCD Command SCD\_CMD\_MSG (BDM->SCD) 19](#_Toc12535552)

[11.2 SCD Command Reply SCD\_CMD\_MSG\_RPLY (SCD->BDM) 19](#_Toc12535553)

[12 Getting Configuration Data from SCD to the BDM 19](#_Toc12535554)

[12.1 Configuration Get Packet CONFIGURATION\_GET\_MSG\_PKT (BDM->SCD) 21](#_Toc12535555)

[12.2 Configuration Get Packet Reply CONFIGURATION\_GET\_MSG\_PKT\_RPLY (SCD->BDM) 21](#_Toc12535556)

[13 Setting Configuration Data from BDM to SCD 22](#_Toc12535557)

[13.1 Configuration Set Packet CONFIGURATION\_SET\_MSG\_PKT (BDM->SCD) 24](#_Toc12535558)

[13.2 Configuration Set Packet Reply CONFIGURATION\_SET\_MSG\_PKT\_RPLY (SCD->BDM) 24](#_Toc12535559)

[14 Appendix A: SCD SETUP\_BLOB 24](#_Toc12535560)

[14.1 SETUP\_BLOB Header (23 bytes) 25](#_Toc12535561)

[15 Serial Number Messages 25](#_Toc12535562)

[15.1 Serial Number SN\_MSG (SCD->BDM) 26](#_Toc12535563)

[15.2 Serial Number Reply SN\_MSG\_RPLY (BDM->SCD) 26](#_Toc12535564)

# Overview

The purpose of this document is to specify DYONICS POWER II and Bridge Device Manager ***(BDM)*** control and data serial protocol (Figure 1). For the purposes of brevity, the communication protocol between the DYONICS POWER II capital device and the INTELLIO Link will be referred to as **PROT\_DII\_BRIDGE** for the remainder of the document.

# System Overview

Figure 1 demonstrates the relationship between the INTELLIO Link and the Capital Device and names the control/data protocol **PROT\_DII\_BRIDGE**. Figure 2 outlines the INTELLIO Connect configuration for this specification. It is important to note that this document describes the protocol between the DYONICS POWER II platform and INTELLIO Link depicted in the middle of the diagram.



Figure 1 Relationship between DYONICS POWER II Capital Device and the Bridge Device Manager

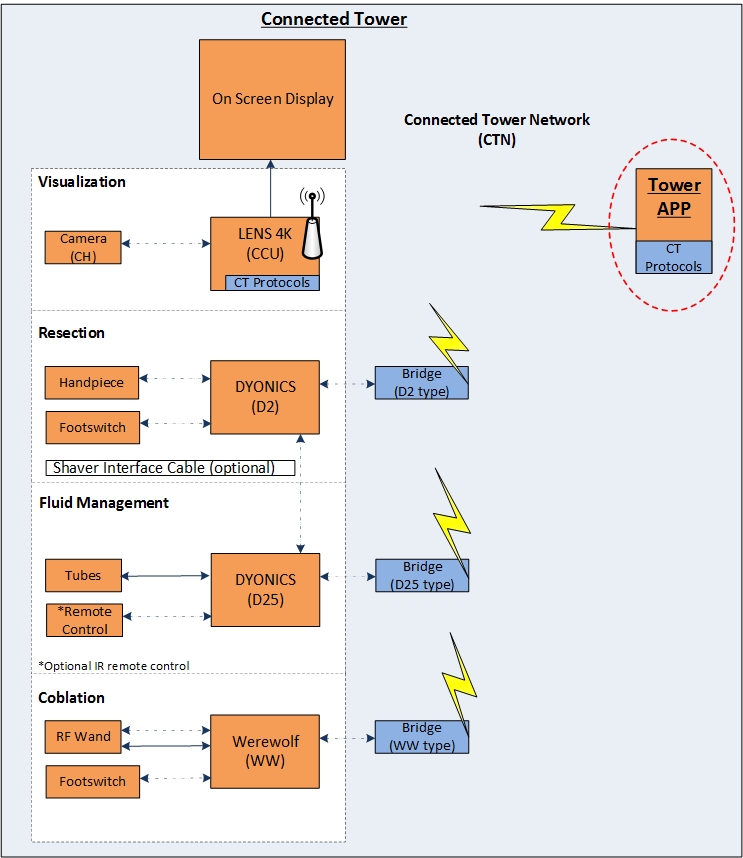


Figure 2. Example of DYONICS II device as part of INTELLIO Connect

# Goals and assumptions

The goal of **PROT\_DII\_BRIDGE** is to provide the control interfaces and data needed for the ***BDM*** to support the INTELLIO Connect Protocol functionality, currently defined as but not limited to Table 1:

Table 1 General Description of INTELLIO Connect Protocols

|  |  |
| --- | --- |
| **INTELLIO Connect Protocol** | **Description** |
| **Discovery Protocol** | Protocol that establishes and maintains connection and identification of the Capital Device on INTELLIO Connect***.*** |
| **GET/SET Protocol** | Protocol that exposes state of device critical parameters to INTELLIO Connect primarily for display (GET) and allows temporary changes of those critical parameters by devices on the INTELLIO Connect (SET). |
| **Setup Blob Protocol** | Protocol that allows INTELLIO Connect devices to:   * **Get** the current configuration of a capital device without knowing the detailed configuration makeup of the capital device, for the purposes of saving that state for future **sets.** * **Set** the default configuration state of a capital device without knowing the detailed configuration makeup of the capital device. This serves the purpose of allowing an INTELLIO Touch user to change the default state of all capital devices via one user interface action. |
| **Device Blob Protocol** | Protocol that allows INTELLIO Connect devices to request (GET) procedure/device specific information for the purposes of archiving to current patient procedure and archiving to the INTELLIO Cloud. |
| **OSD Manager Protocol** | Protocol that allows INTELLIO Connect devices to control the visualization On Screen Display (**OSD**) functionality for the purposes of reflecting capital device status information onto the visualization monitor. |
| **Trigger Protocol** | Protocol that allows predefined capital device functionality to be triggered by an INTELLIO Connect device. |

DYONICS POWER II System Controller Software Requirements Specification (SAP# 150006952) contains the requirements as they relate to each INTELLIO Connect Protocol for the DYONICS POWER II device that is part of INTELLIO Connect via the INTELLIO Link except for the OSD Manager Protocol which is detailed in the DYONICS POWER II INTELLIO Link SRS (SAP #15007726).

For the remainder of the document this INTELLIO Link enabled DYONICS POWER II will be referred to as the Shaver Capital Device (**SCD**).

# Network Message Definition

## Physical Network Message

All message lengths are variable, but it has a minimum length of 6 bytes (no command data)

* Protocol ID Byte <*used to define the protocol/message structure used*> (1 Byte). This byte also acts as a message start byte that is used to synchronize messages for this protocol.
* Command Byte (1 Byte)
  + A non-zero value that corresponds to a specific message function.
  + Values less than or equal to 0x32 are used for Request Messages that do not require Message Retry Handling as defined in Section 4.4 as they have their own special message handling. Capital Device Discovery Commands (Section 6) and Heart Beat Commands (Section 7) should always be defined in this range.
  + Values greater than 0x32 are used for Request Messages that require Message Retry Handling as defined in Section 4.4.
* Request Number Byte (1 Byte)
  + Used for a request/reply session. The requester increments the Request Number for every new request. The receiver shall re-use the Request Number for its reply to the request. The sender can verify the reply message is corresponding to the original request by checking this field.
  + The highest order bit (B7) is set (1) when the message is a request and clear (0) when the message is a reply. This allows for easier message processing.
  + The lower 7 bits (B6-B0) are the Request Number which is a value in the range of 0x01 to 0x7F (1-128). A Request Number of 0x00 is reserved for NAK of an invalid Command.
* Command Data Length word (1 Byte)
* Command Data <variable length (up to 255 Bytes)>
* Checksum Byte: Running 2’s Complement checksum (1 Byte)
* Message End (1 Byte = 0xFC)

**Protocol ID**

**Command**

**Command Data Length**

**Command Data (0…n) bytes**

**Checksum**

**Request**

**Number**

**Message End**

Figure 3 Network Message Format

The Message End is not displayed in the rest of the message examples since it is only needed for message framing.

## Message Reception

### Protocol ID

In order to receive a message from the incoming data stream, the message parser must first receive the Protocol ID. All other values will be discarded until a valid Protocol ID is received. Once a valid Protocol ID has been received, the message receiver will transition to receive the message Command.

### Command

After receiving the Protocol ID, the next value received from the data stream is presumed to be the message Command. The message receiver will determine if the value is a valid Command. If the Command is valid, the message receiver will transition to receive the Request Number.

If the Command is not valid, a NACK message will be returned, with the cause of Command ID not supported. The message receiver will revert to looking for Protocol ID.

### Request Number

After receiving the Command, the next value received from the data stream is presumed to be the message Request Number. For request messages, the Request Number should be a single increment from the preceding request message. For reply messages the Request Number should be the Request Number of the request message that the reply message is for. After receiving the Request Number, the message receiver will transition to receive the Command Data Length.

### Command Data Length

After receiving the Request Number, the next value received from the data stream is presumed to be the message Command Length. The message receiver will determine if the value is a valid Command Data Length based upon the Command being received. If the Command Data Length is valid, the message receiver will transition to receive Command Data for a Command Length greater than 0, and to receive Checksum for a Command Length of 0.

If the Command Data Length is not valid, a NACK message will be returned, with the cause of *Command Data Length not supported*. The message receiver will revert to looking for Protocol ID.

### Command Data

After receiving a non 0 Command Data Length, the message receiver will begin appending values to the Command Data. The message receiver will continue to append received values from data stream to the message Command Data until Command Data Length values have been appended and then transition to receiving the Checksum.

### Checksum

After receiving the Command Data, the next value received from the data stream is presumed to be the message Checksum. The message receiver will determine if a valid Command has been received by comparing the received Checksum against a checksum of the values from the Command ID through the last Command Data value. If the Checksum verification passes, the message receiver will transition to receive Message End.

If the Checksum verification fails, a NACK message will be returned, with the cause of *Checksum Error*. The message receiver will revert to looking for Protocol ID.

### Message End

After receiving a valid Checksum, the next value received from the data stream is presumed to be the Message End. The message receiver will determine if the value is a valid Message End. If the value is valid, the message receiver will process the Command Data. The message receiver will revert to looking for Protocol ID.

If the Message End is not valid, a NACK message will be returned, with the cause of *Framing Error*. The message receiver will revert to looking for Protocol ID.

## Running 2’s Complement Checksum Algorithm

The following is the fundamental algorithm used to calculate the 2’s Complement Checksum used for the **GC\_RUTP.**

* Add all values from array, from Protocol ID through Command Data, into an integer(a)
* Mask the lower 8 bits of (a) and save in 8 bit integer(b)
* Return an unsigned Integer cast of integer & two’s complement it (~b + 1)

An example of this in ***c-pseudo code***:

***u8 Checksum(u8 \*Buf, int Size)***

***{***

***u8 Sum = 0;***

***while (Size-- > 0)***

***Sum += \*Buf ++;***

***return ~Sum + 1;***

***}***

## Message Retry Handling

Message Retry Handling is performed on all Request Messages with Command values greater than 0x32. This excludes Capital Device Discovery (Section 6) and Heart Beat (Section 7) messages. Message Retry Handling is not performed on Reply Messages.

Messages are sent back and forth between the BDM and SCD in Request / Reply Pairs. Each side sends a Request Message and then expects back a Reply Message to that Request. In order to ensure a reliable exchange of data between the BDM and SCD the following rules apply to all Request / Reply Message Pairs:

* The first time there is no Reply Message to a Request Message after a minimum of 250ms (but no later than 325ms) or the reply to a Request Message is a NAK then the Request Message is resent.
* The second time there is no Reply Message to a Request Message after a minimum 250ms (but no later than 325ms) or the reply to a Request Message is a NAK then the Request Message is resent.
* The third time there is no Reply Message to a Request Message after a minimum 250ms (but no later than 325ms) or the reply to a Request Message is a NAK then the BDM or SCD will flush all queued Messages and revert to back to Capital Device Discovery.

The following table defines the Request Message and corresponding Reply Message for each Message Pair. Section numbers are provided for referencing detail on each of the Messages.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Request** | | | **Reply** | | |
| **Sent From** | **Section** | **Message** | **Sent From** | **Section** | **Message** |
| SCD | 8.1 | Port Status | BDM | 8.2 | Port Status Reply |
| BDM | 8.3 | Get Port Status | SCD | 8.1 | Port Status |
| BDM | 9.1 | Set Device Info | SCD | 9.2 | Set Device Info Reply |
| SCD | 10.1 | Lavage Toggle Event | BDM | 10.2 | Lavage Toggle Reply |
| BDM | 11.1 | SCD Command | SCD | 11.2 | SCD Command Reply |
| BDM | 12.1 | Configuration Get Packet | SCD | 12.2 | Configuration Get Packet Reply |
| BDM | 13.1 | Configuration Set Packet | SCD | 13.2 | Configuration Set Packet Reply |
| SCD | 15.1 | Serial Number | BDM | 15.2 | Serial Number Reply |

## Physical connection

The physical connection between the BDM and SCD is an RS232 serial connection configured as 115200 baud 8N1 (resulting in an 86uS inter-byte delay 10bit/115200bps). No HW flow control will be used, and thus software must provide mechanisms for message framing and frame error and overflow recovery mechanisms.

# Detailed NAK

The Detailed NAK is sent as a negative acknowledgement if there is a message error. This NAK is used for errors in the Message Reception or in reply to a Discovery Message.

## Detailed NAK DETAILED\_NAK (BDM->SCD)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **PROTOCOL HEADER** | | | | **COMMAND DATA** | **CHECKSUM** |
| **PID <>** | **CMD**  **<>** | **RN**  **<>** | **LEN**  **<>** | **Command Data** | **CHECKSUM**  **<>** |
| 0x35 | 0x30 | See  Section 4.1 | 0x03 | Refer to table below <3 BYTES> | Calculated |

## Detailed NAK DETAILED\_NAK (SCD->BDM)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **PROTOCOL HEADER** | | | | **COMMAND DATA** | **CHECKSUM** |
| **PID <>** | **CMD**  **<>** | **RN**  **<>** | **LEN**  **<>** | **Command Data** | **CHECKSUM**  **<>** |
| 0x53 | 0x30 | See  Section 4.1 | 0x03 | Refer to table below <3 BYTES> | Calculated |

Command data definition (3 Bytes)

|  |  |  |  |
| --- | --- | --- | --- |
| ENTRY | BYTES | Description | Detailed Description |
| **<NACK>** | 1 | **Negative Acknowledgment** | 1 – General error  2 – Version not supported  3 – Protocol ID not supported  4 – Command ID not supported  5 – Command data length not supported  6 – Checksum error  7 – Framing error |
| **<COMMAND>** | 1 | **Command Number** | The Command of the message that this reply is directed. |
| **<REQUEST\_NUMBER>** | 1 | **Request Number** | The Request Number of the message to which the reply is directed, 00 if the request number is not known. |

# Capital Device Discovery

After completion of self-initialization, the BDM will attempt to initiate communications with a capital device by sending a Discovery Request once every 100ms (nominally but no more than 125ms maximum) until receiving a Discovery Request Reply from the SCD.

After completion of self-initialization, the SCD will wait for a Discovery Request from the BDM. Upon receiving the request, the SCD will send the Discovery Request Reply. If the SCD is ready and the major protocol versions are compatible, the SCD will transition to waiting for a Heart Beat. At the release of this document revision, the protocol version number is 2.1 (0x21).

After successful completion of Device Discovery, the BDM will begin transmission of the Heart Beat message.

If an interruption of communications between the BDM and SCD occurs, the BDM will attempt to reestablish communications by resuming periodic transmission of the Discovery Request message, and the SCD will revert to waiting for the Discovery Message.

## Discovery Request DR\_MSG (BDM->SCD)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **PROTOCOL HEADER** | | | | **COMMAND DATA** | **CHECKSUM** |
| **PID <>** | **CMD**  **<>** | **RN**  **<>** | **LEN**  **<>** | **Command Data** | **CHECKSUM**  **<>** |
| 0x35 | 0x31 | See Section 4.1 | 0x01 | Refer to table below <1 BYTE> | Calculated |

Command data definition (1 Byte)

|  |  |  |  |
| --- | --- | --- | --- |
| ENTRY | BYTES | Description | Detailed Description |
| **<Version\_Number>** | 1 | Protocol Version Number | B0:B3 Minor Version Number (0-15)  B4:B7 Major Version Number (0-15) |

## Discovery Request Reply DR\_MSG\_RPLY (SCD->BDM)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **PROTOCOL HEADER** | | | | **COMMAND DATA** | **CHECKSUM** |
| **PID <>** | **CMD**  **<>** | **RN**  **<>** | **LEN**  **<>** | **Command Data** | **CHECKSUM**  **<>** |
| 0x53 | 0x31 | See  Section 4.1 | 0x04 | Refer to table below <4 BYTES> | Calculated |

Command data definition (4 Bytes)

|  |  |  |  |
| --- | --- | --- | --- |
| ENTRY | BYTES | Description | Detailed Description |
| **<Version\_Number>** | 1 | Protocol Version Number | B0:B3 Minor Version Number (0-15)  B4:B7 Major Version Number (0-15) |
| **<DEVRDY>** | 1 | Device ready state | B0 Device Ready 0 – NOT Ready,  1 – Ready  B1:B7 Reserved |
| **<DEV TYPE>** | 1 | Device Type | 0x01 – Shaver Device Type |
| **<DEV SUB TYPE>** | 1 | Device Subtype | 0x01 – DYONICS POWER II Sub Device Type |

Note: Device Type and Sub Device Type must match the definitions in the INTELLIO Connect Discovery Protocol.

# Heart Beat

## Heart Beat Status HB\_MSG (BDM->SCD)

After successful discovery, the BDM will send a Heart Beat Status request (HB\_MSG) at a regular period of 100ms. The SCD will respond with Heart Beat Status Reply (HB\_MSG\_RPLY) to the BDM. The Heart Beat serves to meet the following requirements:

* The BDM will unilaterally provide a Heart Beat at a period of 100ms (nominally but no more than 125ms maximum) regardless of whether the bridge is connected or not. This serves the purpose of removing any dependency on startup sequence for the capital device.
* The bridge will determine device ready from the reply to this message on the connection or from the communicated state in the Command data. The BDM will publish ready status to all protocols subscribing. The primary use is for the discovery protocol, as the bridge will not join the INTELLIO Connect until the BDM is ready and identified.
* The BDM will initiate this message whenever it has completed its initialization and is ready for INTELLIO Connect interfaces to be used.
* This message serves to provide constant update of status information such as the SCD is connected to the INTELLIO Connect (ala connect indicator on devices with a front panel display)
* The HB\_MSG send state information from other devices published into the device manager such as Pump and the Connected/Disconnected and Pump On/Off.

If one second goes by without the SCD receiving a Heart Beat Status message or one second goes by without the BDM receiving a Heart Beat Status Reply message then the SCD or BDM will flush all queued Messages and revert to back to Capital Device Discovery.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **PROTOCOL HEADER** | | | | **COMMAND DATA** | **CHECKSUM** |
| **PID <>** | **CMD**  **<>** | **RN**  **<>** | **LEN**  **<>** | **Command Data** | **CHECKSUM**  **<>** |
| 0x35 | 0x32 | See  Section 4.1 | 0x01 | Refer to table below <1 BYTE> | Calculated |

Command data definition (1 Byte)

|  |  |  |  |
| --- | --- | --- | --- |
| ENTRY | BYTES | Description | Detailed Description |
| **< INTELLIO\_CONNECT\_PUMP>** | 1 | INTELLIO Connect Pump Status | B0 **INTELLIO Connect Pump Connection** 0 – NOT Connected,  1 – Connected  B1 **INTELLIO Connect Pump Run State** 0 – Stopped,  1 – Running  B2:B7 Future Expansion |

## Heart Beat Status Reply HB\_MSG\_RPLY (SCD->BDM)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **PROTOCOL HEADER** | | | | **COMMAND DATA** | **CHECKSUM** |
| **PID <>** | **CMD**  **<>** | **RN**  **<>** | **LEN**  **<>** | **Command Data** | **CHECKSUM**  **<>** |
| 0x53 | 0x32 | See  Section 4.1 | 0x01 | Refer to table below <1 Byte> | Calculated |

Command data definition (1 byte)

| ENTRY | BYTES | Description | Detailed Description |
| --- | --- | --- | --- |
| **<DEV\_STATUS>** | 1 | **Device Status** | B0 **Capital** **Device Ready** 0 – NOT Ready,  1 – Ready  B1:B7 **Reserved** |

# Shaver Status Messages

After the completion of Discovery, the SCD will send the status message once to synchronize the BDM with the SCD status. After the initial update, the SCD will then send the individual status messages when data in a message changes so that the BDM reflects the current shaver status. Also the BDM may request a status message through use of the Get Port Status Message.

These messages serve to provide updates of critical parameters for the use by at least the GET part of the INTELLIO Connect GET/SET protocol but will serve to keep INTELLIO Connect Protocols updated on state changes if INTELLIO Connect protocols subscribe to any field.

## Port Status PORT\_STATUS\_MSG (SCD->BDM)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **PROTOCOL HEADER** | | | | **COMMAND DATA** | **CHECKSUM** |
| **PID <>** | **CMD**  **<>** | **RN**  **<>** | **LEN**  **<>** | **Command Data** | **CHECKSUM**  **<>** |
| 0x53 | 0x33 | See  Section 4.1 | 0x06 | Refer to table below <6 BYTES> | Calculated |

Command data definition (6 bytes)

| ENTRY | BYTES | Description | Detailed Description |
| --- | --- | --- | --- |
| **<PORT\_A\_DISPLAY>** | 1 | Port A Set Speed Units, Blade, Mode, Display Up Arrow and Display Down Arrow | B0:B1 **Port A** **Units**  0 – No Device, 1 – RPM, 2 – RATE, 3 – Percent  B2:B3 **Port A** **Blade** 0 – Low Speed, 1 – Medium Speed, 2 – High Speed, 3 – Other  B4:B5 **Port A Mode** 0 – Forward, 1 – Reverse, 2 – Oscillate Mode 1, 3 – Oscillate Mode 2  B6 **Port A** **Up Arrow** 0 – Disabled, 1 – Enabled  B7 **Port A Down Arrow** 0 – Disabled, 1 – Enabled |
| **<PORT\_A\_SPEED\_RUN>** | 1 | Port A Set Speed and Run State | B0:B6 **Port A Set Speed**  1 - 100 (In units of 100 RPM. Multiply by 100 to get the displayed range of 100 – 10,000 RPM)  1 - 9 (in units of RATE)  10 - 100 (in units of Percent)  B7 **Port A Run State** 0 – Stopped, 1 – Running |
| **<PORT\_B\_DISPLAY>** | 1 | Port B Set Speed Units, Blade, Mode, Up Arrow Display and Down Arrow Display | B0:B1 **Port B Units**  0 – No Device, 1 – RPM, 2 – RATE, 3 – Percent  B2:B3 **Port B** **Blade** 0 – Low Speed, 1 – Medium Speed, 2 – High Speed, 3 – Other  B4:B5 **Port B Mode** 0 – Forward, 1 – Reverse, 2 – Oscillate Mode 1, 3 – Oscillate Mode 2  B6 **Port B Up Arrow** 0 – Disabled, 1 – Enabled  B7 **Port B Down Arrow** 0 – Disabled, 1 – Enabled |
| **<PORT\_B\_SPEED\_RUN>** | 1 | Port B Set Speed and Run State | B0:B6 **Port B** **Set Speed**  1 - 100 (In units of 100 RPM. Multiply by 100 to get the displayed range of 100 – 10,000 RPM)  1 - 9 (in units of RATE)  10 - 100 (in units of Percent)  B7 **Port B** **Run State** 0 – Stopped, 1 – Running |
| **<PORT\_A\_B\_ERR\_WARN>** | 1 | Port A and Port B Errors and Warnings. | B0:B3 **Port A** **Err and Warn**  See Section 8.1.1  B4:B7 **Port B** **Err and Warn**  See Section 8.1.1 |
| **<SETTINGS\_POPUPS>** | 1 | Capital Device Popups, Handpiece Override and Blade Recall. | B0:B2 **Capital** **Device Popups**  0 – No Error 1 – Unable to Save Custom Settings 2 – Unable to Save Set Speeds 3 – Unable to Retrieve Custom Settings 4 – Unable to Retrieve Set Speeds  B3 **Handpiece Override** 0 – Disabled, 1 – Enabled  B4 **Blade Recall** 0 – Disabled, 1 – Enabled  B5 **Pump Port** 0 – Port A, 1 – Port B  B6 **Footswitch Port** 0 – Port A, 1 – Port B  B7 **Settings Screen** 0 – Out of the Settings Screen, 1 – In the Settings Screen |

### SCD Errors and Warnings

The SCD supports the reporting of the following Device Warnings in <PORT\_A\_B\_ERR\_WARN> for Port A (bits B0-B3) and Port B (bits B4-B7):

|  |  |  |  |
| --- | --- | --- | --- |
| Message Number (Hex) | Message Meaning | Message ID | Message Type |
| 0x00 | No Error | - | None |
| 0x01 | Temperature Failure | PW1 | Status |
| 0x02 | Unknown Blade | PW2 | Status |
| 0x03 | Unknown Handpiece | PW3 | Status |
| 0x04 | Handpiece Sensor Fault | PW4 | Status |
| 0x05 | Blade Stall | PW5 | Status |
| 0x06 | Handpiece Stall | PW6 | Status |
| 0x07 | Handpiece Motor Fault | PW7 | Status |
| 0x08 | Short Circuit Detected | PW8 | Status |
| 0x09 | Handpiece Overload | PW10 | Status |
| 0x0A | Handpiece Overload Timeout. | PW11 | Status |
| 0x0B | Handpiece Error | PW12 | Status |
| 0x0C | Unknown Footswitch | PW13 | Status |
| 0x0D | Footswitch Error | PW14 | Status |
| 0x0E | Footswitch Battery Low | PW15 | Status |
| 0x0F | Footswitch required | PW16 | Status |

## Port Status Reply PORT\_STATUS\_MSG\_RPLY (BDM->SCD)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **PROTOCOL HEADER** | | | | **COMMAND DATA** | **CHECKSUM** |
| **PID <>** | **CMD**  **<>** | **RN**  **<>** | **LEN**  **<>** | **ACK/NAK** | **CHECKSUM**  **<>** |
| 0x35 | 0x33 | See Section 4.1 | 0x01 | 0 = ACK  >0=NAK | Calculated |

The BDM sends a Port Status Reply Message in reply to a Port Status Message sent from the SCD as a request message.

## Get Port Status GET\_PORT\_STATUS\_MSG (BDM->SCD)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **PROTOCOL HEADER** | | | | **COMMAND DATA** | **CHECKSUM** |
| **PID <>** | **CMD**  **<>** | **RN**  **<>** | **LEN**  **<>** | **Command Data** | **CHECKSUM**  **<>** |
| 0x35 | 0x34 | See  Section 4.1 | 0x00 | None | Calculated |

When the BDM determines that a resynchronization to the SCD status is required, the BDM sends a Get Port Status Message request message to the SCD. The SCD replies with a PORT\_STATUS\_MSG message with the RN set to reply message. In this case the BDM does not send a Port Status Reply Message as the BDM initiated this request/reply session.

# Setting Device Specific Information

## Set Device Info SET\_DEVICE\_INFO\_MSG (BDM->SCD)

The Set Device Information Message is a single byte containing a series of binary bit pairs where values 01 or 10 indicate updates for the associated parameter and values 00 and 11 indicate no change. With this method, one or more Device Information Settings can be updated with a single message.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **PROTOCOL HEADER** | | | | **COMMAND DATA** | **CHECKSUM** |
| **PID**  **<>** | **CMD**  **<>** | **RN**  **<>** | **LEN**  **<>** | **Packet Data** | **CHECKSUM**  **<>** |
| 0x35 | 0x35 | See  Section 4.1 | 0x01 | Refer to table below <1 BYTE> | Calculated |

Command data definition (1 byte)

| **ENTRY** | **BYTES** | **Description** | **Detailed Description** |
| --- | --- | --- | --- |
| **<HAND\_BLADE\_PUMP\_FOOT>** | 1 |  | B0:B1 Hand Control Override 1 – Off, 2 – On  B2:B3 Blade Defaults Settings 1 – Manufacturer Defaults 2 – User Custom Defaults  B4:B5 Pump Port 1 – Port A 2 – Port B  B6:B7 Footswitch Port 1 – Port A 2 – Port B |

## Set Device Info Reply SET\_DEV\_INFO\_MSG\_RPLY (SCD->BDM)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **PROTOCOL HEADER** | | | | **COMMAND DATA** | **CHECKSUM** |
| **PID <>** | **CMD**  **<>** | **RN**  **<>** | **LEN**  **<>** | **ACK/NAK** | **CHECKSUM**  **<>** |
| 0x53 | 0x35 | See  Section 4.1 | 0x01 | 0 = ACK  >0=NAK | Calculated |

# Lavage Toggle Event Function (SCD->BDM)

In order to support remote toggling of the Pump Lavage Function the SCD will send a LAVAGE\_TOGGLE\_EVENT\_MSG message to the BDM. The BDM will forward the message to BDM of the Pump to turn Pump Lavage On if Pump Lavage is Off, otherwise turn Pump Lavage Off if Pump Lavage in On.

## Lavage Toggle Event LAVAGE\_TOGGLE\_EVENT\_MSG (SCD->BDM)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **PROTOCOL HEADER** | | | | **COMMAND DATA** | **CHECKSUM** |
| **PID <>** | **CMD<>** | **RN**  **<>** | **LEN**  **<>** | **Command Data** | **CHECKSUM**  **<>** |
| 0x53 | 0x36 | See  Section 4.1 | 0x00 | None | Calculated |

## Lavage Toggle Reply LAVAGE\_TOGGLE\_MSG\_RPLY (BDM->SCD)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **PROTOCOL HEADER** | | | | **COMMAND DATA** | **CHECKSUM** |
| **PID <>** | **CMD**  **<>** | **RN**  **<>** | **LEN**  **<>** | **ACK/NAK** | **CHECKSUM**  **<>** |
| 0x35 | 0x36 | See  Section 4.1 | 0x01 | 0 = ACK  >0=NAK | Calculated |

# SCD Command Function

Send a command to SCD. The SCD will send a PORT\_STATUS\_MSG to the BDM if the command caused any status information change.

## SCD Command SCD\_CMD\_MSG (BDM->SCD)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **PROTOCOL HEADER** | | | | **COMMAND DATA** | **CHECKSUM** |
| **PID <>** | **CMD<>** | **RN**  **<>** | **LEN**  **<>** | **Command Data** | **CHECKSUM**  **<>** |
| 0x35 | 0x37 | See  Section 4.1 | 0x01 | Refer to table below <1 BYTE> | Calculated |

Command data definition (1 byte)

| **ENTRY** | **BYTES** | **Description** | **Detailed Description** |
| --- | --- | --- | --- |
| **<SCD\_CMD>** | 1 |  | **SCD Command** 1 – Port A Up Arrow,  2 – Port A Down Arrow,  3 – Port A Delta Oscillate Mode, 4 – Port B Up Arrow,  5 – Port B Down Arrow,  6 – Port B Delta Oscillate Mode, 7 – OK, 8 – Exit Settings |

## SCD Command Reply SCD\_CMD\_MSG\_RPLY (SCD->BDM)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **PROTOCOL HEADER** | | | | **COMMAND DATA** | **CHECKSUM** |
| **PID <>** | **CMD**  **<>** | **RN**  **<>** | **LEN**  **<>** | **ACK/NAK** | **CHECKSUM**  **<>** |
| 0x53 | 0x37 | See  Section 4.1 | 0x01 | 0 = ACK  >0=NAK | Calculated |

# Getting Configuration Data from SCD to the BDM

The following commands will allow the INTELLIO Connect to take a snapshot of the device (***INTELLIO Connect*** *Setup Blob Protocol*) in its current state and save that state so that users can recall it at another time for ease of setup.

To avoid causing too much load on the SCD or BDM because of the size of the SCD Configuration Data, the transfer of the SCD Configuration Data is broken up into Packets with a maximum Packet Data size of 48 bytes each.

The first Packet contains just the SETUP\_BLOB Header. The remaining Packets contain the SETUP\_BLOB Data and SETUP\_BLOB Data Checksum byte.

Figure 4 provides a sequence diagram of the entire transfer.

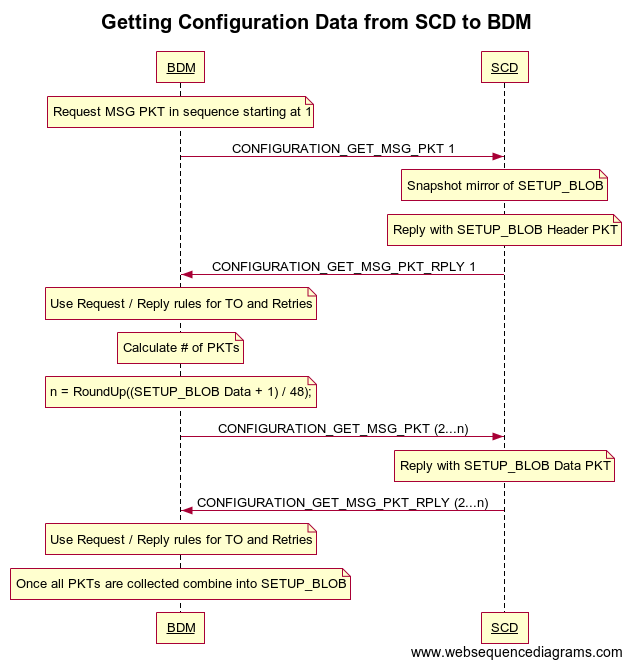


Figure 4 Getting Configuration Data from SCD to BDM

title Getting Configuration Data from SCD to BDM

note over BDM: Request MSG PKT in sequence starting at 1

BDM->SCD: CONFIGURATION\_GET\_MSG\_PKT 1

note over SCD: Snapshot mirror of SETUP\_BLOB

note over SCD: Reply with SETUP\_BLOB Header PKT

SCD->BDM: CONFIGURATION\_GET\_MSG\_PKT\_RPLY 1

note over BDM: Use Request / Reply rules for TO and Retries

note over BDM: Calculate # of PKTs

note over BDM: n = RoundUp((SETUP\_BLOB Data + 1) / 48);

BDM->SCD: CONFIGURATION\_GET\_MSG\_PKT (2...n)

note over SCD: Reply with SETUP\_BLOB Data PKT

SCD->BDM: CONFIGURATION\_GET\_MSG\_PKT\_RPLY (2...n)

note over BDM: Use Request / Reply rules for TO and Retries

note over BDM: Once all PKTs are collected combine into SETUP\_BLOB

## Configuration Get Packet CONFIGURATION\_GET\_MSG\_PKT (BDM->SCD)

The BDM requests the SCD **SETUP\_BLOB** (defined in Section 14) by sending the CONFIGURATION\_GET\_MSG\_PKT command with the Packet Count set to 1 indicating that this is the start of a **SETUP\_BLOB** Transmission.

The SCD responds to the Packet 1 request with the CONFIGURATION\_GET\_MSG\_PKT\_RPLY command that contains the information needed to calculate the total number of Packets required for transmitting the SCD **SETUP\_BLOB**.

After the BDM figures out the total number of Packets, it then requests each of the remaining Packets in order starting at Packet 2 until it reaches the total number of Packets that containing the data from the **SETUP\_BLOB Data** (Section 14.2) and the **Checksum byte of SETUP\_BLOB Data**.

The BDM then takes all the data from the Packets it requested and recombines them back into a single **SETUP\_BLOB**.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **PROTOCOL HEADER** | | | | **COMMAND DATA** | **CHECKSUM** |
| **PID**  **<>** | **CMD**  **<>** | **RN**  **<>** | **LEN**  **<>** | **Packet Count** | **CHECKSUM**  **<>** |
| 0x35 | 0x38 | See  Section 4.1 | 0x01 | <1...n> | Calculated |

## Configuration Get Packet Reply CONFIGURATION\_GET\_MSG\_PKT\_RPLY (SCD->BDM)

This is the reply message to each of the CONFIGURATION\_GET\_MSG\_PKT requests.

Whenever the SCD receives a request for Packet 1 it will first acquire an up to date snapshot of the **SETUP\_BLOB** including updating any checksums and then it will reply with 23 bytes of **SETUP\_BLOB Header** (defined in section 14.1 below).

The SCD response is to send the **SETUP\_BLOB** Header information (Section 14.1) in Packet 1.

The SCD calculates the Total Number of Packets that is required to send the **SETUP\_BLOB** using the following formula:

n = Total Number of Packets = 1 + Rounded up Value of ((Length of SETUP\_BLOB data + 1) / 48)

Replies to requests for Packets 2…n will contain the remaining data to be transferred (**SETUP\_BLOB Data** and the **Checksum byte of SETUP\_BLOB Data)** which gets broken up evenly into 48 byte chunks.

*It is important to note that the length of* ***SETUP\_BLOB Data*** *is always in (multiples of 48 bytes) - 1 for the checksum.*

This makes it easy to verify the CONFIGURATION\_GET\_MSG\_PKT\_RPLY command since it will only have two possible values and has the benefit of providing potential padding of the SETUP\_BLOB Data that could be used for future expansion without having to increase the total number of Packets.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **PROTOCOL HEADER** | | | | **COMMAND DATA** | | **CHECKSUM** |
| **PID**  **<>** | **CMD**  **<>** | **RN**  **<>** | **LEN**  **<>** | **Packet Count** | **Packet Data** | **CHECKSUM**  **<>** |
| 0x53 | 0x38 | See  Section 4.1 | 24  49 | 1  <2…n> | (SETUP\_BLOB Header)  (SETUP\_BLOB Data + Checksum) | Calculated |

# Setting Configuration Data from BDM to SCD

Similar to the Getting the Configuration Data from the SCD to the BDM, the following sequence diagram is provided to outline a multi transfer message sequence to allow the BDM to set the capital device configuration state. For context, this will be used to allow the user on the tablet to change the entire state for the devices with one user action.

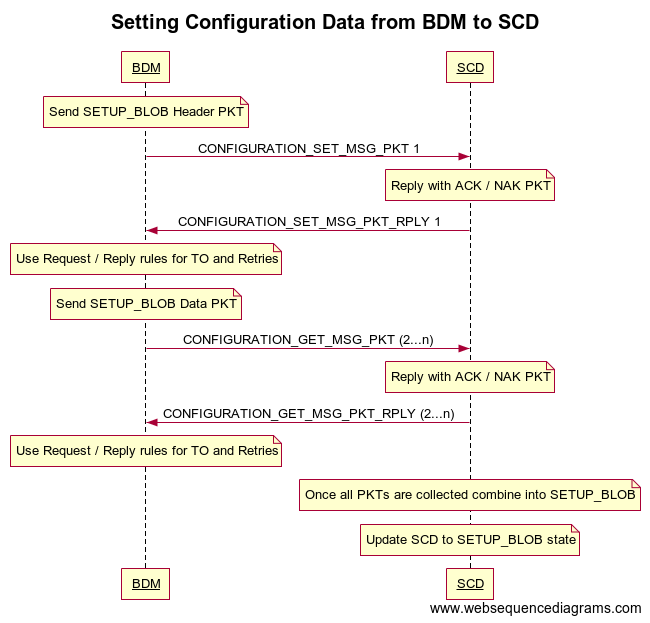


Figure 5 Setting Configuration Data from BDM to SCD

title Setting Configuration Data from BDM to SCD

note over BDM: Send SETUP\_BLOB Header PKT

BDM->SCD: CONFIGURATION\_SET\_MSG\_PKT 1

note over SCD: Reply with ACK / NAK PKT

SCD->BDM: CONFIGURATION\_SET\_MSG\_PKT\_RPLY 1

note over BDM: Use Request / Reply rules for TO and Retries

note over BDM: Send SETUP\_BLOB Data PKT

BDM->SCD: CONFIGURATION\_GET\_MSG\_PKT (2...n)

note over SCD: Reply with ACK / NAK PKT

SCD->BDM: CONFIGURATION\_GET\_MSG\_PKT\_RPLY (2...n)

note over BDM: Use Request / Reply rules for TO and Retries

note over SCD: Once all PKTs are collected combine into SETUP\_BLOB

note over SCD: Update SCD to SETUP\_BLOB state

## Configuration Set Packet CONFIGURATION\_SET\_MSG\_PKT (BDM->SCD)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **PROTOCOL HEADER** | | | | **COMMAND DATA** | | **CHECKSUM** |
| **PID <>** | **CMD**  **<>** | **RN**  **<>** | **LEN**  **<>** | **Packet Count** | **Packet Data** | **CHECKSUM**  **<>** |
| 0x35 | 0x39 | See  Section 4.1 | 24  49 | 1  <2…n> | (SETUP\_BLOB Header)  (SETUP\_BLOB Data + Checksum) | Calculated |

## Configuration Set Packet Reply CONFIGURATION\_SET\_MSG\_PKT\_RPLY (SCD->BDM)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **PROTOCOL HEADER** | | | | **COMMAND DATA** | | | **CHECKSUM** |
| **PID <>** | **CMD**  **<>** | **RN**  **<>** | **LEN**  **<>** | | **Packet Count** | **ACK/NAK** | **CHECKSUM**  **<>** |
| 0x53 | 0x39 | See  Section 4.1 | 0x02 | | <1…n> | 0 = ACK  >0=NAK | Calculated |

# Appendix A: SCD SETUP\_BLOB

The SETUP\_BLOB for the SCD consists of the SETUP\_BLOB Header followed by the SETUP\_BLOB Data and then the Checksum byte of the SETUP\_BLOB Data.

|  |  |  |
| --- | --- | --- |
| **SETUP\_BLOB Header** | **SETUP\_BLOB Data** | **Checksum byte of SETUP\_BLOB Data** |
| See section #[14.1](#SETUP_BLOB Header (25 bytes)) | See section #14.1 for size constraints | Calculated |

Note: The field data below is an excerpt from the Setup Blob Protocol, refer to that document for final implementation but it is provided here to provide context.

## SETUP\_BLOB Header (23 bytes)

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Size in Bytes** | **Value** | **Description** |
| Version of the SETUP\_BLOB | 1 | Varies – Based upon BLOB data | Provides backward compatibility for the capital device or a means to reject request wholesale. The byte is broken into 4 bits of Major (B4:B7) and 4 bits of Minor (B0:B3). The SCD uses the Version of the SETUP\_BLOB to determine if the SETUP\_BLOB is compatible. |
| Length of SETUP\_BLOB header | 1 | 23 | Size of Blob header |
| Length of SETUP\_BLOB data | 2 | Varies – Used to calculate number of packets | Length of the setup data (Big-Endian format). This value MUST be in multiples of 48 bytes minus one for the checksum (i.e. 47, 95, 143, …, or 12191) which will be sent in a Packet Count of 2, 3, 4, …, or 255 Packets. |
| INTELLIO Connect Device Type | 1 | 1 | Shaver Controller |
| INTELLIO Connect Sub Device Type | 1 | 1 | DYONICS POWER II |
| SETUP\_BLOB name | 16 | “DYONICS POWER II” | Could be used to make a standalone capital device with a webserver selection of setups, or display of setup on capital device interface |
| Checksum of header | 1 | Calculated | Used to validate header |

# Serial Number Messages

Send the serial number string of the SCD, Port A Device or Port B Device to the BDM.

Whenever a Device Serial Number is uninitialized, the Device does not support transmission of serial numbers or the Device is not present the Serial Number message string will be blank.

After the completion of Discovery the SCD will send the following Serial Number messages once to the BDM:

* The SCD Serial Number message.
* The Port A Handpiece Serial Number message.
* The Port B Handpiece Serial Number message.

After the initial connection, the SCD will then send Port A and Port B Serial Number messages whenever a Handpiece that supports transmission of a serial number is plugged into or removed from the SCD.

Whenever the BDM reverts to back to Capital Device Discovery it must blank all of the Serial Number strings it has stored.

## Serial Number SN\_MSG (SCD->BDM)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **PROTOCOL HEADER** | | | | **COMMAND DATA** | **CHECKSUM** |
| **PID <>** | **CMD**  **<>** | **RN**  **<>** | **LEN**  **<>** | **Command Data** | **CHECKSUM**  **<>** |
| 0x53 | 0x3A | See  Section 4.1 | 0x0C | Refer to table below <12 BYTES> | Calculated |

Command data definition (12 bytes)

| ENTRY | BYTES | Description | Detailed Description |
| --- | --- | --- | --- |
| **<SERIAL\_NUMBER>** | 11 | Serial Number | A string of up to 10 alphanumeric characters followed by a null character for a total of 11 bytes.  All bytes that are not part of the serial number string are set to null. |
| **<DEVICE>** | 1 | Which device this serial number represents | **Device** 0 – SCD,  1 – Port A,  2 – Port B |

## Serial Number Reply SN\_MSG\_RPLY (BDM->SCD)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **PROTOCOL HEADER** | | | | **COMMAND DATA** | **CHECKSUM** |
| **PID <>** | **CMD**  **<>** | **RN**  **<>** | **LEN**  **<>** | **ACK/NAK** | **CHECKSUM**  **<>** |
| 0x35 | 0x3A | See Section 4.1 | 0x01 | 0 = ACK  >0=NAK | Calculated |

The BDM sends a Serial Number Reply Message in reply to a Serial Number Message sent from the SCD as a request message.