DRAFT

BSR E1.37-7 - 202x

Additional Message Sets for ANSI E1.20 (RDM) - Gateway & Splitter Configuration Messages

Approved by the ANSI Board of Standards Review on _____

CP/2016-1026r4

NOTICE and DISCLAIMER

ESTA does not approve, inspect, or certify any installations, procedures, equipment or materials for compliance with codes, recommended practices or standards. Compliance with an ESTA standard or recommended practice is the sole and exclusive responsibility of the manufacturer or provider and is entirely within their control and discretion. Any markings, identification or other claims of compliance do not constitute certification or approval of any type or nature whatsoever by ESTA.

ESTA neither guarantees nor warrants the accuracy or completeness of any information published herein and disclaim liability for any personal injury, property or other damage or injury of any nature whatsoever, whether special, indirect, consequential or compensatory, directly or indirectly resulting from the publication, use of, or reliance on this document.

In issuing and distributing this document, ESTA does not either (a) undertake to render professional or other services for or on behalf of any person or entity, or (b) undertake any duty to any person or entity with respect to this document or its contents. Anyone using this document should rely on their own independent judgment or, as appropriate, seek the advice of a competent professional in determining the exercise of reasonable care in any given circumstance.

Published By:

Entertainment Services and Technology Association (ESTA)
271 Cadman Plaza PO Box 23200
Brooklyn, NY 11202-3200
USA
Phone: +1-212-244-1505
Email: standards@esta.org

ESTA's Technical Standards Program

ESTA's Technical Standards Program was created to serve the ESTA membership and the entertainment industry in technical standards-related matters. The goal of the program is to take a leading role regarding technology and safety within the entertainment industry by creating recommended practices and standards, monitoring standards issues around the world on benefit of our members, and improving communications and safety within the industry. In its technical standards development activities, ESTA works closely with other industry organizations, including ESA, CITT, USITT and VPLT, in addition to representing the interests of ESTA members to ANSI, UL, ASCE, ICC, and NFPA. ESTA is an ANSI Accredited Standards Developer.

The Technical Standards Council (TSC) established by ESTA's Board of Directors to oversee and coordinate the Technical Standards Program. Made up of individuals experienced in standards development work from throughout our industry, the Committee approves all projects undertaken and assigns them to the appropriate working group. The Technical Standards Council employs a Technical Standards Manager to coordinate the work of the Committee and its working groups as well as maintaining a "Standards Watch" on behalf of members. Working groups include: Control Protocols, Electrical Power, Event Safety, Floors, Fog and Smoke, Followspot Positions, Mental Health & Well-being Management, Photometrics, Rigging, Stage Machinery, and Prop Weapons Safety.

ESTA encourages active participation in the Technical Standards Program. There are several ways to become involved. The easiest way to actively participate is to respond to any of the public reviews advertised on ESTA's public review web page. The next level of participation requires completion of an application to become a working group member; applications are available from the TSP's procedural documents web page. Application as an Observer with non-voting status affords access to updates on standards development documents. Application as a voting participant affords full access to a consensus voice that helps shape the industry. Voting status carries responsibilities of responding to letter ballots and attending meetings, but membership in ESTA or any other organization is not a requirement for participation in the TSP. One can also become involved by requesting that the TSC develop a standard or a recommended practice in an area of concern to them.

The Control Protocols Working Group, which authored this standard, consists of a cross section of entertainment industry professionals representing a diversity of interests. ESTA is committed to developing consensus-based standards and recommended practices in an open setting.

Investors in Innovation

The Technical Standard Program (TSP) is financially supported by ESTA and by companies and individuals who make donations to the TSP. Contributing companies and individuals who have helped fund the TSP are recognized as “Investors in Innovation”. The Investors in Innovation when this standard was approved by ANSI's Board of Standards Review are gratefully acknowledged as follows:

[Insert the current Investors table here. Source from latest Swatch Edition]

Memorial donor: The Estate of Ken Vannice

All donations to the Technical Standards Program benefit the entire program, and are not directed to any specific use or project within the program. Please help support the Technical Standards Program by becoming an Investor in Innovation. Visit our website at http://tsp.esta.org/invest, or contact the ESTA office at 1-212-244-1505 and select "TSP" from the menu.

Contact Information

Technical Standards Manager

Richard J. Nix
ESTA
271 Cadman Plaza PO Box 23200
New York, NY 11202-3200
USA
+1-212-244-1505
richard.nix@esta.org

Technical Standards Council Chairperson

Alan Rowe
I.A.T.S.E Local 728
Phone: 1 310-702-2909
amrowe@iatse728.org

Dan Culhane
Wenger Corp
+1-612-868-4769
culhane.dan@gmail.com

Control Protocols Working Group Co-chairpersons

Javid Butler
Goddard Design, LLC.
+1-702-759-2427
javid@goddard.design

Milton Davis
Doug Fleenor Design, Inc.
+1-805-481-9599
milton@dfd.com

Maya Nigrosh
mnigrosh@alumni.cmu.edu

Acknowledgments

The Control Protocols Working Group members when this document was approved by the working group on [CPWG approval date], are gratefully acknowledged below.

Voting members:

Observer (non-voting) members:

Interest category codes:

CP = custom-market producer
DE = designer
DR = dealer rental company
G = general interest
MP = mass-market producer
U = user

Table of Contents

List of Tables

1 Introduction

1.1 Overview & Scope

This document provides additional Get/Set Parameter Messages for use with the ANSI E1.20 Remote Device Management protocol [RDM] and ANSI E1.33 RDMnet protocol [RDMnet].

This document contains messages relating to configuring managed splitters, proxy devices, and RDMnet Devices.

1.2 E1.20 (RDM) Basic Features

The ANSI E1.20 Remote Device Management protocol (RDM) [RDM] permits intelligent bi-directional communication between devices from multiple manufacturers using a modified DMX512 data link. RDM is an EF1.0 implementation of ANSI E1.11, see Annex B in [DMX].

RDM permits a console or other controlling device to discover and then configure, monitor, and manage intermediate and end-devices connected through a DMX512 network. RDM provides intelligent control of devices on a DMX512 network.

1.3 E1.33 (RDMnet) Basic Features

The ANSI E1.33 [RDMnet] protocol extends the capabilities of RDM to operate on RDMnet Devices natively on an IP network by encapsulating the E1.20 packet information into E1.33 packets. Significantly different from E1.20 is that E1.33 allows for multi-controller networks.

2 Normative References

[Art-Net] Art-Net 4. https://art-net.org.uk/downloads/art-net.pdf

This document is maintained by Singularity (UK) Ltd on behalf of Artistic Licence Engineering Ltd.

Singularity (UK) Ltd
The Potting Shed, Caseley Wood
Lustleigh, Devon TQ13 9TN, UK
office@singularity-uk.com

[DMX] ANSI E1.11 Entertainment Technology – USITT DMX512-A Asynchronous Serial Digital Data Transmission Standard for controlling lighting equipment and accessories.

This standard is maintained by ESTA.

ESTA
271 Cadman Plaza PO Box 23200
New York, NY 11202-3200
USA
+1-212-244-1505
http://tsp.esta.org/

ESTA is a standardization body accredited by ANSI to develop, maintain and withdraw American National Standards.

ANSI
25 West 43rd Street
4th floor
New York, NY 10036
+1-212-642-4900

http://www.ansi.org

[EIA-485] ANSI/TIA/EIA-485-A-1998 Electrical Characteristics of Generators & Receivers for Use in Balanced Digital Multipoint Systems

This standard is maintained by EIA/TIA.

Electronics Industries Alliance
2500 Wilson Boulevard
Arlington , VA 22201-3834 USA
+1-703-907-7500
http://www.eia.org/

Telecommunications Industry Association
2500 Wilson Blvd., Suite 300
Arlington, VA 22201 USA
+1-703- 907-7700 fax: +1-703-907-7727
http://www.tiaonline.org/

Note: EIA-485-A is compatible with: ISO/IEC 8482:1993 Information Technology - Telecommunications and information exchange between systems - Twisted pair multipoint interconnections.

[KiNET] KiNet v3. https://www.colorkinetics.com/global

This document is maintained by Color Kinetics Inc. A Signify company.

[RDM] ANSI E1.20 Entertainment Technology – Remote Device Management over DMX512 networks. This reference includes the current and previous versions of the standard.

This standard is maintained by ESTA.

[RDM-2025] ANSI E1.20 - 2025 Entertainment Technology – Remote Device Management over DMX512 networks.

This standard is maintained by ESTA.

[RDMnet] ANSI E1.33 Entertainment Technology – (RDMnet) Message Transport and Management for ANSI E1.20 (RDM) compatible and similar devices over IP Networks

This standard is maintained by ESTA.

[sACN] ANSI E1.31 Entertainment Technology - Lightweight streaming protocol for transport of DMX512 using ACN

This standard is maintained by ESTA.

[PIDS-1] ANSI E1.37-1 Entertainment Technology - Additional Message Sets for ANSI E1.20 (RDM) – Part 1, Dimmer Message Sets

This standard is maintained by ESTA.

3 General

3.1 Overview

These Parameter Messages are intended for managing and configuring a specific class of products that route DMX data through different network types or physical links. There are several common product types that fall into this class:

RDMnet Devices – RDMnet Devices respond to RDM GET_COMMAND and SET_COMMAND messages on IP networks. See [RDMnet] Section 7.1.3.

RDMnet Gateways – RDMnet Gateways are a class of RDMnet Devices that handle routing data between IP-based networks and DMX512 physical links. See [RDMnet] Section 7.3.5 for more information on the operation of RDMnet Gateways.

Splitters – Splitters are devices that route a DMX512 physical link to multiple physical links. See [RDM] Section 4 on “In-line Devices” for more information on the operation of Splitter devices.

Proxies – As defined in [RDM] Section 8, a proxy is any in-line device that acts as an agent or representative for one or more devices. Represented devices may be non-RDM devices and may be logical (non-physical) devices. Proxies are commonly associated with Wireless DMX512/RDM systems but can also apply to some managed Splitters as well. See [RDM] Section 8 for more information on the operation of Proxies.

3.2 Sub-Device Handling

Refer to Section 9 of [RDM] for information on Sub-Device usage. This document does not change or modify the requirements stated in ANSI E1.20. Requirements stated in this document are in addition to the stated ANSI E1.20 requirements.

3.3 Text Field Handling

Text fields shall conform to Section 10.1 in [RDM].

3.4 Byte Ordering

All multi-byte data shall be transmitted as specified in Section 6.1 of [RDM].

4 RDMnet Devices and Gateways

4.1 General

The messages in this document allow for the management of various parameters on the Endpoints (Section 4.2) for RDMnet Devices and Gateways, including mapping of E1.31 universes to DMX512 inputs and outputs.

4.2 RDMnet Endpoints

RDMnet Endpoints (including both Physical Endpoints and Virtual Endpoints) are defined in Section 7.1.6 of [RDMnet]. The details and requirements for their use can be found in Section 7.3.2 of [RDMnet].

5 Splitters & Proxies

5.1 General

RDM Splitters and Proxies are devices that exist solely on E1.20 networks and do not contain an E1.33 interface. See [RDM] Sections 4 and 8, respectively, for information on RDM Splitters and Proxies.

Managed Splitters and Proxies are devices that can be found via RDM Discovery (see [RDM], Section 7) and are represented with their own UID. The parameters listed in this document are designed to be sent to the UID of these devices.

Devices that contain Proxy functionality and also RDMnet Device or Gateway functionality will need to follow the requirements for their respective components. An example of this might be an RDMnet Gateway that also contains a wireless proxy.

5.2 RDM Endpoints

The parameter messages defined in this document use Endpoints to represent EIA-485 interfaces (referred to as "ports" in [RDM]) on splitters and proxies.

Each Endpoint is assigned a 16-bit unsigned integer identifier referred to as an Endpoint ID. Endpoint ID 0x0000 is reserved and shall not be used as an Endpoint ID. The valid range of Endpoint Identifiers is 0x0001 – 0xF9FF as specified in [RDMnet] Section 3.4.

Each Endpoint can operate in one of three modes; Disabled, Input or Output. The behavior of an Endpoint according to its mode is shown in Table A-5.

Endpoint Management Parameter Messages provide a mechanism to enumerate and control the Endpoints on a device. For example, the ENDPOINT_LIST PID (Section 6.1) can be used to fetch a list of Endpoint IDs (ports) that exist on a splitter.

There is no requirement that Endpoints be configurable.

5.3 Splitter / Proxy Discovery

Splitters / Proxies that follow the requirements for Proxies in Section 8 of [RDM], may perform Discovery ([RDM] Section 7.1). In these products, the DISCOVERY_STATE (Section 6.8) and BACKGROUND_DISCOVERY (Section 6.9) parameters should be used to report and configure this behavior.

If the splitter / proxy maintains a mapping of UIDs to Endpoints this information can be obtained with the ENDPOINT_RESPONDERS parameter.

5.4 Endpoint Routing

Some Splitter and Proxy devices may support the routing of multiple inputs to a specified output using Endpoints. The ENDPOINT_TO_UNIVERSE message (See Section 6.4) may be used to accomplish assigning the Inputs to Outputs.

Since these devices exist entirely on an E1.20 network the Universe Number has no relation or connection to the E1.31 Universe normally used in this message for RDMnet Devices and Gateways. In other words, the Universe Number in this context exists only inside this device for the purpose of assigning inputs and outputs.

To assign an Output Endpoint to an Input Endpoint, the Universe Number for the Output Endpoint shall be set to the same Universe Number as assigned to the desired Input Endpoint.

6 RDM Parameter Messages for Endpoint Management

The following RDM Parameter Messages are intended for managing endpoint properties.

Unless specified otherwise, the messages below are allowed to be used for both E1.33 and E1.20 networks.

6.1 Get Endpoint List (ENDPOINT_LIST)

This parameter is used to retrieve a packed list of all Endpoints that exist on a device. All devices shall support ENDPOINT_LIST.

Controller: (GET)

(Port ID) 0x01 – 0xFF	(Controller Flags) Bit Field	(Sub-Device) 0x0000 (Root)
(CC) GET_COMMAND	(PID) ENDPOINT_LIST	(PDL) 0x00
(PD) Not Present		

Response: (GET)

(Response Type) ACK / ACK_OVERFLOW	(Message Count) 0x01 – 0xFF	(Sub-Device) Copy of Controller SD
(CC) GET_COMMAND_ RESPONSE	(PID) ENDPOINT_LIST	(PDL) First: 0x07 – 0xE5 Subsequent: 0x03 – 0xE7
(PD) *--- First Response Only ---* List Change Number (32-bit) Packed list of 24-bit structures containing the following: Endpoint ID (16-bit) Endpoint Type		

Data Description:

List Change Number: The List Change Number is detailed in Section 6.2. This field is only included once.

The remaining contents contain a packed list of 24-bit structures containing the Endpoint IDs that exist on the device and their Endpoint Types.

In an ACK_OVERFLOW condition, the List Change Number is only sent in the first message. All subsequent ACK_OVERFLOW packets shall only contain the packed list of 24-bit structures.

Endpoint ID: See Section 5.2 for information on Endpoint IDs. In the case of an RDMnet Device, NULL_ENDPOINT (defined in [RDMnet] Appendix A) shall not be included in the list.

Endpoint Type: This field indicates if the Endpoint is a Physical or Virtual Endpoint as defined in [RDMnet] Section 7.1.6. A device may have a combination of both Physical and Virtual Endpoints. Endpoint Types are enumerated in Table A-6.

6.2 Get Endpoint List Change (ENDPOINT_LIST_CHANGE)

This parameter returns a List Change Number as a means for controllers to identify if the Endpoint List has changed.

RDMnet Devices shall send an unsolicited ENDPOINT_LIST_CHANGE response when the list of Endpoints present on the device changes.

RDM Devices shall queue a ENDPOINT_LIST_CHANGE response when the list of Endpoints present on the device changes.

A Controller shall query the ENDPOINT_LIST_CHANGE after a reconnection event (i.e. re-discovering a device that has been lost) as there is no guarantee that it will not have changed while the controller was disconnected.

Controller: (GET)

(Port ID) 0x01 – 0xFF	(Controller Flags) Bit Field	(Sub-Device) 0x0000 (Root)
(CC) GET_COMMAND	(PID) ENDPOINT_LIST_CHANGE	(PDL) 0x00
(PD) N/A		

Response: (GET)

(Response Type) ACK	(Message Count) 0x01 – 0xFF	(Sub-Device) Copy of Controller SD
(CC) GET_COMMAND_ RESPONSE	(PID) ENDPOINT_LIST_CHANGE	(PDL) 0x04
(PD) List Change Number (32-bit)		

Data Description:

List Change Number:
The Endpoint List Change Number is a monotonically increasing number used by controllers to track that the list of Endpoints has changed, or that an Endpoint Type has changed. This Change Number shall be incremented by one each time the set of Endpoints change. The Change Number is an unsigned 32-bit field which shall roll over from 0xFFFFFFFF to 0. Upon start-up (due to power-on reset, start of software, etc) this field shall be initialized to 0.

6.3 Get/Set Identify Endpoint (IDENTIFY_ENDPOINT)

This parameter is used for the user to identify an Endpoint on a device.
Endpoints on a device shall identify themselves if possible, using a visible and/or audible action. If identification is not possible for an Endpoint, the device shall respond with a NACK with a NACK Reason Code of NR_ACTION_NOT_SUPPORTED.

If the device supports IDENTIFY_MODE (Section 3.2 of [PIDs-1]), then the behavior of IDENTIFY_ENDPOINT may change depending on the setting for IDENTIFY_MODE

The current Endpoint identification state may be obtained using a GET: IDENTIFY_ENDPOINT message.

Controller: (GET)

(Port ID) 0x01 – 0xFF	(Controller Flags) Bit Field	(Sub-Device) 0x0000 (Root)
(CC) GET_COMMAND	(PID) IDENTIFY_ENDPOINT	(PDL) 0x02
(PD) Endpoint ID (16-bit)		

Response: (GET)

(Response Type) ACK	(Message Count) 0x01 – 0xFF	(Sub-Device) Copy of Controller SD
(CC) GET_COMMAND_RESPONSE	(PID) IDENTIFY_ENDPOINT	(PDL) 0x03
(PD) Endpoint ID (16-bit) Identify State Off/On (0/1)		

Controller: (SET)

(Port ID) 0x01 – 0xFF	(Controller Flags) Bit Field	(Sub-Device) 0x0000 (Root)
(CC) SET_COMMAND	(PID) IDENTIFY_ENDPOINT	(PDL) 0x03
(PD) Endpoint ID (16-bit) Identify State Off/On (0/1)		

Response: (SET)

(Response Type) ACK	(Message Count) 0x01 – 0xFF	(Sub-Device) Copy of Controller SD
(CC) SET_COMMAND_RESPONSE	(PID) IDENTIFY_ENDPOINT	(PDL) 0x02
(PD) Endpoint ID (16-bit)		

Data Description:

Endpoint ID: Valid Endpoint IDs for a device can be obtained using the ENDPOINT_LIST message. An Endpoint ID of BROADCAST_ENDPOINT ([RDMnet] Appendix A) shall only be used as an All Call in a SET_COMMAND message, setting all Endpoints to the same value.

If the Endpoint ID is not a valid Endpoint ID then the device shall respond with a NACK Reason Code of NR_ENDPOINT_NUMBER_INVALID.

In the case of an RDMnet Device, NULL_ENDPOINT (defined in [RDMnet] Appendix A) is not a valid Endpoint ID. A device receiving a command with the Endpoint ID field set to NULL_ENDPOINT shall respond with NR_ENDPOINT_NUMBER_INVALID. If an Identify message is desired for the device itself then the IDENTIFY_DEVICE message from [RDM] should be used.

Identify State: For a GET_COMMAND_RESPONSE message, a 0x00 shall indicate the Identify State for the specified Endpoint is “off.” A 0x01 shall indicate the Identify State for the specified Endpoint is “on.” For a SET_COMMAND message, 0x01 shall activate the Identify State for the specified Endpoint. A 0x00 shall deactivate the Identify State.

6.4 Get/Set Endpoint to Universe (ENDPOINT_TO_UNIVERSE)

This parameter is used to assign an Endpoint on a device to a specific E1.31 DMX512 Universe. It may also be used within a Splitter to assign inputs to outputs.

Controller: (GET)

(Port ID) 0x01 – 0xFF	(Controller Flags) Bit Field	(Sub-Device) 0x0000 (Root)
(CC) GET_COMMAND	(PID) ENDPOINT_TO_UNIVERSE	(PDL) 0x02
(PD) Endpoint ID (16-bit)		

Response: (GET)

(Response Type) ACK	(Message Count) 0x01 – 0xFF	(Sub-Device) Copy of Controller SD
(CC) GET_COMMAND_ RESPONSE	(PID) ENDPOINT_TO_UNIVERSE	(PDL) 0x04
(PD) Endpoint ID (16-bit) Universe Number (16-bit)		

Controller: (SET)

(Port ID) 0x01 – 0xFF	(Controller Flags) Bit Field	(Sub-Device) 0x0000 (Root)
(CC) SET_COMMAND	(PID) ENDPOINT_TO_UNIVERSE	(PDL) 0x04
(PD) Endpoint ID (16-bit) Universe Number (16-bit)		

Response: (SET)

(Response Type) ACK	(Message Count) 0x01 – 0xFF	(Sub-Device) Copy of Controller SD
(CC) SET_COMMAND_ RESPONSE	(PID) ENDPOINT_TO_UNIVERSE	(PDL) 0x02
(PD) Endpoint ID (16-bit)		

Data Description:

Endpoint ID: Valid Endpoint IDs for a device can be obtained using the ENDPOINT_LIST message. An Endpoint ID of BROADCAST_ENDPOINT shall only be used as an All Call in a SET_COMMAND message to set all Endpoints to the same value.

If the Endpoint ID is not a valid Endpoint ID then the device shall respond with a NACK Reason Code of NR_ENDPOINT_NUMBER_INVALID.

In the case of an RDMnet Device, NULL_ENDPOINT (defined in [RDMnet] Appendix A) is not a valid Endpoint ID. A device receiving a command with the Endpoint ID field set to NULL_ENDPOINT shall respond with NR_ENDPOINT_NUMBER_INVALID.

Universe Number: When configured as a Standard Universe, the Universe Number shall be in the range from 1-63999 (0x0001 – 0xF9FF).

When the universe is configured as a Composite Universe, the Universe field shall be 0xFFFF. Standard and Composite Universe types are described in Sections 7.3.2.1.2 and 7.3.2.1.3 of [RDMnet].

When the Endpoint is unpatched (i.e. when there is no connection for DMX512 type data), the Universe Number shall be 0x0000. An Endpoint configured as Universe Number 0x0000 shall not act on DMX512 data from any universe. RDM communication may still occur with RDM Devices connected to endpoints that are in an unpatched state. To unpatch a universe from an endpoint, a SET_COMMAND using a Universe Number of 0x0000 shall be sent.

Only Universe Numbers in the range 0 - 63999 are valid in SET_COMMAND messages. An Endpoint receiving a SET_COMMAND with the Universe Number set to 0xFFFF (Composite Universe) or any other Universe Numbers outside the range above shall respond with NACK reason NR_DATA_OUT_OF_RANGE.

Some Universe Numbers may not be valid, depending on the selected Network Stream as described in Section 6.16 Get/Set Endpoint Network Stream (ENDPOINT_NETWORK_STREAM). For example, when the Network Stream is set to NS_ART_NET the valid range of Universe Numbers is 0 – 32767.

6.5 Get/Set Endpoint Mode (ENDPOINT_MODE)

This parameter is used to specify whether an Endpoint operates in Input, Output, or Disabled mode.

Controller: (GET)

(Port ID) 0x01 – 0xFF	(Controller Flags) Bit Field	(Sub-Device) 0x0000 (Root)
(CC) GET_COMMAND	(PID) ENDPOINT_MODE	(PDL) 0x02
(PD) Endpoint ID (16-bit)		

Response: (GET)

(Response Type) ACK	(Message Count) 0x01 – 0xFF	(Sub-Device) Copy of Controller SD
(CC) GET_COMMAND_ RESPONSE	(PID) ENDPOINT_MODE	(PDL) 0x03
(PD) Endpoint ID (16-bit) Endpoint Mode		

Controller: (SET)

(Port ID) 0x01 – 0xFF	(Controller Flags) Bit Field	(Sub-Device) 0x0000 (Root)
(CC) SET_COMMAND	(PID) ENDPOINT_MODE	(PDL) 0x03

(PD)
Endpoint ID (16-bit)
Endpoint Mode

Response: (SET)

(Response Type) ACK	(Message Count) 0x01 – 0xFF	(Sub-Device) Copy of Controller SD
(CC) SET_COMMAND_ RESPONSE	(PID) ENDPOINT_MODE	(PDL) 0x02
(PD) Endpoint ID (16-bit)		

Data Description:

Endpoint ID: Valid Endpoint IDs for a device can be obtained using the ENDPOINT_LIST message. An Endpoint ID of BROADCAST_ENDPOINT shall only be used as an All Call in a SET_COMMAND message to set all Endpoints to the same value.

If the Endpoint ID is not a valid Endpoint ID then the device shall respond with a NACK Reason Code of NR_ENDPOINT_NUMBER_INVALID.

In the case of an RDMnet Device, NULL_ENDPOINT (defined in [RDMnet] Appendix A) is not a valid Endpoint ID. A device receiving a command with the Endpoint ID field set to NULL_ENDPOINT shall respond with NR_ENDPOINT_NUMBER_INVALID.

Endpoint Mode: Endpoint mode settings are enumerated in Table A-5. If the requested mode in a SET_COMMAND is not a valid action then it shall respond with a NACK Reason Code of NR_ACTION_NOT_SUPPORTED.

6.6 Get/Set Endpoint Label (ENDPOINT_LABEL)

This parameter provides a means of setting a descriptive label for each Endpoint on a device. This may be used to specify the purpose of that Endpoint or the origination source for the data.

Controller: (GET)

(Port ID) 0x01 – 0xFF	(Controller Flags) Bit Field	(Sub-Device) 0x0000 (Root)
(CC) GET_COMMAND	(PID) ENDPOINT_LABEL	(PDL) 0x02
(PD) Endpoint ID (16-bit)		

Response: (GET)

(Response Type) ACK	(Message Count) 0x01 – 0xFF	(Sub-Device) Copy of Controller SD
(CC) GET_COMMAND_ RESPONSE	(PID) ENDPOINT_LABEL	(PDL) 2-34 (2 + Number of bytes sent)
(PD) Endpoint ID (16-bit) Text Label. Up to 32 bytes.		

Controller: (SET)

(Port ID) 0x01 – 0xFF	(Controller Flags) Bit Field	(Sub-Device) 0x0000 (Root)
(CC) SET_COMMAND	(PID) ENDPOINT_LABEL	(PDL) 2-34 (2 + Number of bytes sent)
(PD) Endpoint ID (16-bit) Text Label. Up to 32 bytes.		

Response: (SET)

(Response Type) ACK	(Message Count) 0x01 – 0xFF	(Sub-Device) Copy of Controller SD
(CC) SET_COMMAND_ RESPONSE	(PID) ENDPOINT_LABEL	(PDL) 0x02
(PD) Endpoint ID (16-bit)		

Data Description:

Endpoint ID: Valid Endpoint IDs for a device can be obtained using the ENDPOINT_LIST message. An Endpoint ID of BROADCAST_ENDPOINT shall only be used as an All Call in a SET_COMMAND message to set all Endpoints to the same value.

If the Endpoint ID is not a valid Endpoint ID then the device shall respond with a NACK Reason Code of NR_ENDPOINT_NUMBER_INVALID.

In the case of an RDMnet Device, NULL_ENDPOINT (defined in [RDMnet] Appendix A) is not a valid Endpoint ID. A device receiving a command with the Endpoint ID field set to NULL_ENDPOINT shall respond with NR_ENDPOINT_NUMBER_INVALID.

Text Label:
The text field shall observe the requirements of [RDM] Section 10.1 for text fields. The field may be up to 32 bytes.

6.7 Get/Set Enable RDM Traffic on Endpoint (RDM_TRAFFIC_ENABLE)

This parameter is used to enable or disable the RDM Traffic on a specified Endpoint. This shall only affect traffic using the defined RDM Start Code and will not affect the traffic for Null Start Codes [DMX] or any other Alternate Start Codes. The RDM Start Code (SC_RDM) is defined in [RDM] Appendix A.

RDMnet Devices receiving SET_COMMAND messages containing the Endpoint ID of a Virtual Endpoint shall respond with a NACK with a NACK Reason Code of NR_ACTION_NOT_SUPPORTED.

Controller: (GET)

(Port ID) 0x01 – 0xFF	(Controller Flags) Bit Field	(Sub-Device) 0x0000 (Root)
(CC) GET_COMMAND	(PID) RDM_TRAFFIC_ENABLE	(PDL) 0x02
(PD) Endpoint ID (16-bit)		

Response:

(Response Type) ACK	(Message Count) 0x01 – 0xFF	(Sub-Device) Copy of Controller SD
(CC) GET_COMMAND_ RESPONSE	(PID) RDM_TRAFFIC_ENABLE	(PDL) 0x03
(PD) Endpoint ID (16-bit)		
RDM Enabled TRUE/FALSE (1/0)		

Controller: (SET)

(Port ID) 0x01 – 0xFF	(Controller Flags) Bit Field	(Sub-Device) 0x0000 (Root)
(CC) SET_COMMAND	(PID) RDM_TRAFFIC_ENABLE	(PDL) 0x03
(PD) Endpoint ID (16-bit)		
RDM Enabled TRUE/FALSE (1/0)		

Response:

(Response Type) ACK	(Message Count) 0x01 – 0xFF	(Sub-Device) Copy of Controller SD
(CC) SET_COMMAND_ RESPONSE	(PID) RDM_TRAFFIC_ENABLE	(PDL) 0x02
(PD) Endpoint ID (16-bit)		

Data Description:

Endpoint ID: Valid Endpoint IDs for a device can be obtained using the ENDPOINT_LIST message. An Endpoint ID of BROADCAST_ENDPOINT shall only be used as an All Call in a SET_COMMAND message to set all Endpoints to the same value.

If the Endpoint ID is not a valid Endpoint ID then the device shall respond with a NACK Reason Code of NR_ENDPOINT_NUMBER_INVALID.

In the case of an RDMnet Device, NULL_ENDPOINT (defined in [RDMnet] Appendix A) is not a valid Endpoint ID. A device receiving a command with the Endpoint ID field set to NULL_ENDPOINT shall respond with NR_ENDPOINT_NUMBER_INVALID.

RDM Enabled: For a GET_COMMAND, this indicates whether RDM traffic is currently enabled or disabled on the Endpoint. In the case of a SET_COMMAND, this will alert the RDMnet Device to set the state of RDM Enabled to a value of TRUE (0x01) or FALSE (0x00).

If a SET_COMMAND is sent to change the state of an Endpoint and the Endpoint is currently in a configuration where the requested state change would be invalid (e.g. where an Endpoint is currently configured to function as an Endpoint in input mode and the RDMnet Gateway does not allow RDM to be disabled for Endpoints configured in input mode) then the device should respond with a NACK Reason code of NR_ACTION_NOT_SUPPORTED.

If RDM Enabled is set to FALSE (0x00) then traffic for the RDM Start Code packets shall be filtered and not passed through the Endpoint. If it is set to TRUE (0x01) then it shall allow the RDM Start Code packets to pass through.

6.8 Get/Set Discovery State (DISCOVERY_STATE)

This parameter is used to initiate E1.20 RDM Discovery ([RDM] Section 7) of RDM Responders connected to the specified Endpoint or to get the status of the discovery process.

If an Endpoint is in Input Mode then it shall respond with a NACK Reason Code of NR_INVALID_ENDPOINT_MODE.

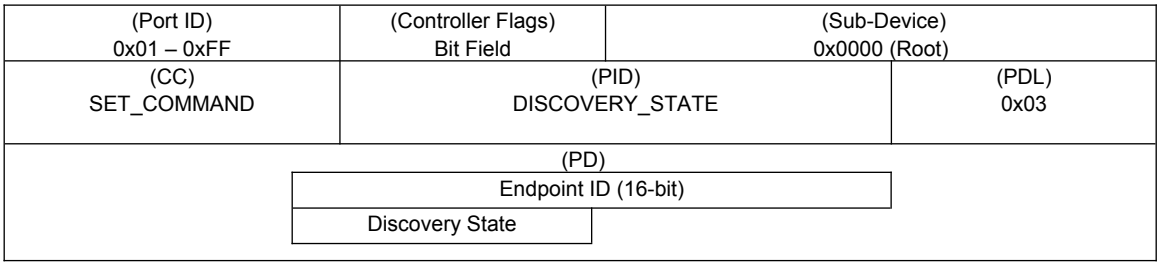
Controller: (GET)

(Port ID) 0x01 – 0xFF	(Controller Flags) Bit Field	(Sub-Device) 0x0000 (Root)
(CC) GET_COMMAND	(PID) DISCOVERY_STATE	(PDL) 0x02
(PD) Endpoint ID (16-bit)		

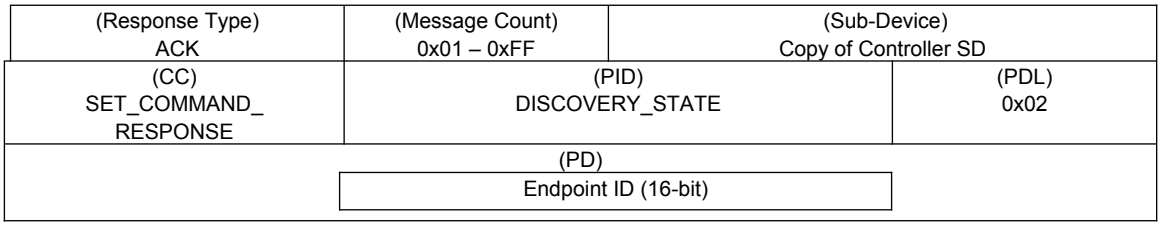
Response: (GET)

(Response Type) ACK	(Message Count) 0x01 – 0xFF	(Sub-Device) Copy of Controller SD
(CC) GET_COMMAND_ RESPONSE	(PID) DISCOVERY_STATE	(PDL) 0x05
(PD) Endpoint ID (16-bit) Device Count (16-bit) Discovery State		

Controller: (SET)



Response: (SET)



Data Description:

Endpoint ID: Valid Endpoint IDs for a device can be obtained using the ENDPOINT_LIST message. An Endpoint ID of BROADCAST_ENDPOINT shall only be used as an All Call in a SET_COMMAND message to set all Endpoints to the same value.

If the Endpoint ID is not a valid Endpoint ID then the device shall respond with a NACK Reason Code of NR_ENDPOINT_NUMBER_INVALID.

In the case of an RDMnet Device, NULL_ENDPOINT (defined in [RDMnet] Appendix A) is not a valid Endpoint ID. A device receiving a command with the Endpoint ID field set to NULL_ENDPOINT shall respond with NR_ENDPOINT_NUMBER_INVALID.

Discovery State: All Discovery States are enumerated in Table A-3 with detailed descriptions.

The GET_COMMAND_RESPONSE message shall only return the current status of discovery. If discovery has never been run before, the device shall return DISCOVERY_INCOMPLETE. Otherwise, if discovery is not currently in the process of executing, then it shall return DISCOVERY_NOT_ACTIVE.

A SET_COMMAND message with a status of DISCOVERY_NOT_ACTIVE shall terminate any discovery process that is currently executing.

If a device receives a SET_COMMAND with an invalid Discovery State such as DISCOVERY_INCOMPLETE, or an out of range value, it shall respond with a NACK Reason Code of NR_ACTION_NOT_SUPPORTED.

Device Count: When discovery is active, the return value for this field shall indicate the number of RDM responders discovered thus far. When discovery is not active, it shall return the number of RDM responders on the Endpoint during the most recent discovery. Devices without the capability to report this information at all shall return a value of DISCOVERY_COUNT_UNKNOWN.

If the discovery state is DISCOVERY_INCOMPLETE, then the Device Count shall be DISCOVERY_COUNT_INCOMPLETE.

The maximum number of responders that may be represented in the Device Count field is 63999. If more responders than this are present it shall report it as the limit of 63999.

6.9 Get/Set Background Discovery (BACKGROUND_DISCOVERY)

This parameter is used to enable/disable background E1.20 RDM discovery for an Endpoint. Background Discovery is an ongoing autonomous discovery routine running on a device that periodically discovers additional RDM Responders as they come online.

The background discovery process may also include the periodic unmuting of all RDM Responders and re-muting all known RDM Responders before performing an incremental discovery to ensure all connected RDM Responders are properly discovered regardless of their previous connection state. The frequency that Background Discovery operates at is up to the implementer and may vary based on the amount of other traffic.

If an Endpoint does not support Background Discovery then it shall respond with a NACK Reason Code of NR_ACTION_NOT_SUPPORTED.

If an Endpoint is in Input Mode then it shall respond with a NACK Reason Code of NR_INVALID_ENDPOINT_MODE.

Note that this parameter will likely have no effect on Virtual Endpoints. If it is not supported on a Virtual Endpoint then it shall respond with a NACK Reason Code of NR_ACTION_NOT_SUPPORTED.

Controller: (GET)

(Port ID) 0x01 – 0xFF	(Controller Flags) Bit Field	(Sub-Device) 0x0000 (Root)
(CC) GET_COMMAND	(PID) BACKGROUND_DISCOVERY	(PDL) 0x02
(PD) Endpoint ID (16-bit)		

Response: (GET)

(Response Type) ACK	(Message Count) 0x01 – 0xFF	(Sub-Device) Copy of Controller SD
(CC) GET_COMMAND_ RESPONSE	(PID) BACKGROUND_DISCOVERY	(PDL) 0x03
(PD) Endpoint ID (16-bit)		
Background Discovery Enabled TRUE/FALSE (1/0)		

Controller: (SET)

(Port ID) 0x01 – 0xFF	(Controller Flags) Bit Field	(Sub-Device) 0x0000 (Root)
(CC) SET_COMMAND	(PID) BACKGROUND_DISCOVERY	(PDL) 0x03
(PD) Endpoint ID (16-bit)		
Background Discovery Enabled TRUE/FALSE (1/0)		

Response: (SET)

(Response Type) ACK	(Message Count) 0x01 – 0xFF	(Sub-Device) Copy of Controller SD
(CC) SET_COMMAND_ RESPONSE	(PID) BACKGROUND_DISCOVERY	(PDL) 0x02
(PD) Endpoint ID (16-bit)		

Data Description:

Endpoint ID: Valid Endpoint IDs for a device can be obtained using the ENDPOINT_LIST message. An Endpoint ID of BROADCAST_ENDPOINT shall only be used as an All Call in a SET_COMMAND message to set all Endpoints to the same value.

If the Endpoint ID is not a valid Endpoint ID then the device shall respond with a NACK Reason Code of NR_ENDPOINT_NUMBER_INVALID.

In the case of an RDMnet Device, NULL_ENDPOINT (defined in [RDMnet] Appendix A) is not a valid Endpoint ID. A device receiving a command with the Endpoint ID field set to NULL_ENDPOINT shall respond with NR_ENDPOINT_NUMBER_INVALID.

Background Discovery: In a GET_COMMAND, this field indicates whether Background Discovery is currently enabled on the specified Endpoint. A value of 0x00 indicates that Background Discovery is disabled, whereas a value of 0x01 indicates that it is enabled. In a SET_COMMAND, this field shall be used to control the setting of E1.20 Background Discovery for the specified Endpoint ID.

6.10 Get/Set Endpoint Timing (ENDPOINT_TIMING)

This parameter is used to get and set the timing profile on Endpoints that support selecting different timing and refresh profiles.

This parameter message shall not be supported on RDMnet Devices that do not have Physical (E1.11 [DMX] or E1.20 [RDM]) Endpoints.

Controller: (GET)

(Port ID) 0x01 – 0xFF	(Controller Flags) Bit Field	(Sub-Device) 0x0000 (Root)
(CC) GET_COMMAND	(PID) ENDPOINT_TIMING	(PDL) 0x02
(PD) Endpoint ID (16-bit)		

Response: (GET)

(Response Type) ACK	(Message Count) 0x01 – 0xFF	(Sub-Device) Copy of Controller SD
(CC) GET_COMMAND_RESPONSE	(PID) ENDPOINT_TIMING	(PDL) 0x04
(PD) Endpoint ID (16-bit)		
Current setting	# of settings available	

Controller: (SET)

(Port ID) 0x01 – 0xFF	(Controller Flags) Bit Field	(Sub-Device) 0x0000 (Root)
(CC) SET_COMMAND	(PID) ENDPOINT_TIMING	(PDL) 0x03
(PD) Endpoint ID (16-bit) Endpoint Timing Setting		

Response (SET):

(Response Type) ACK	(Message Count) 0x01 – 0xFF	(Sub-Device) Copy of Controller SD
(CC) SET_COMMAND_RESPONSE	(PID) ENDPOINT_TIMING	(PDL) 0x02
(PD) Endpoint ID (16-bit)		

Data Description:

Endpoint ID: Valid Endpoint IDs for a device can be obtained using the ENDPOINT_LIST message. An Endpoint ID of BROADCAST_ENDPOINT shall only be used as an All Call in a SET_COMMAND message to set all Endpoints to the same value.

If the Endpoint ID is not a valid Endpoint ID then the device shall respond with a NACK Reason Code of NR_ENDPOINT_NUMBER_INVALID.

In the case of an RDMnet Device, NULL_ENDPOINT (defined in [RDMnet] Appendix A) is not a valid Endpoint ID. A device receiving a command with the Endpoint ID field set to NULL_ENDPOINT shall respond with NR_ENDPOINT_NUMBER_INVALID.

Current Setting / # Settings Available: The GET_COMMAND_RESPONSE message includes the current refresh frequency setting as well as the total number of settings available. These settings shall be consecutively numbered within the device, starting from the number 1.

Text descriptions for each setting can be retrieved using the ENDPOINT_TIMING_DESCRIPTION Parameter. Devices that support ENDPOINT_TIMING shall also support the ENDPOINT_TIMING_DESCRIPTION message.

6.11 Get Endpoint Timing Description (ENDPOINT_TIMING_DESCRIPTION)

This parameter is used to get a descriptive text label for a given Endpoint timing setting. The label may be up to 32 bytes.

This parameter message shall not be supported on RDMnet Devices that do not have Physical (E1.11 [DMX] or E1.20 [RDM]) Endpoints.

Controller: (GET)

(Port ID) 0x01 – 0xFF	(Controller Flags) Bit Field	(Sub-Device) 0x0000 (Root)
(CC) GET_COMMAND	(PID) ENDPOINT_TIMING_DESCRIPTION	(PDL) 0x01
(PD) Endpoint Timing Setting Requested		

Response: (GET)

(Response Type) ACK	(Message Count) 0x01 – 0xFF	(Sub-Device) Copy of Controller SD
(CC) GET_COMMAND_RESPONSE	(PID) ENDPOINT_TIMING_DESCRIPTION	(PDL) 1 – 33 (1 + Number of bytes sent)
(PD) Endpoint Timing Setting Requested Text Label. Up to 32 bytes.		

Data Description:

Endpoint Timing Setting Requested:
This field indicates the Endpoint Timing setting number requested from the GET_COMMAND message.

Text Label:
The text field shall observe the requirements of [RDM] Section 10.1 for text fields. The field may be up to 32 bytes.

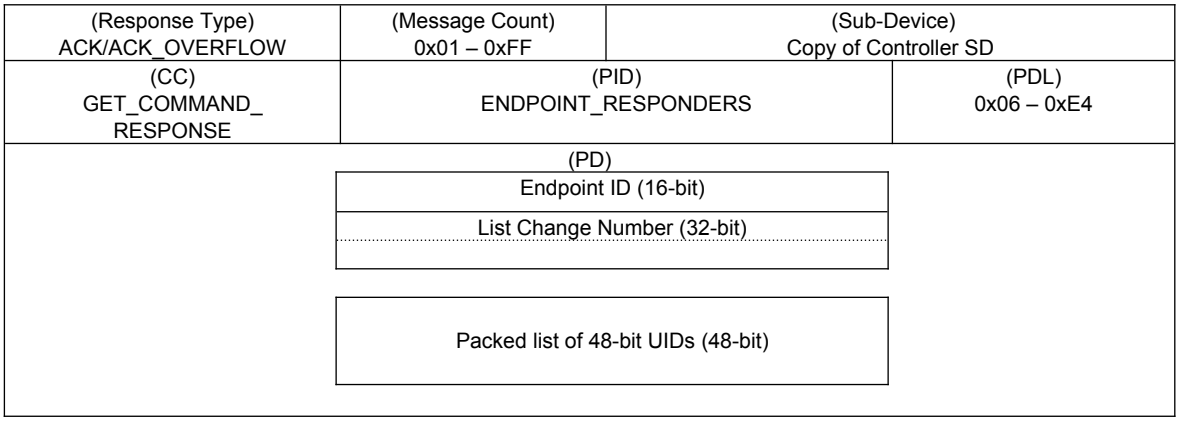
6.12 Get Endpoint Responders (ENDPOINT_RESPONDERS)

This parameter returns a packed list of responder UIDs associated with a given Endpoint ID number. It is used in conjunction with the ENDPOINT_RESPONDER_LIST_CHANGE message.

Controller: (GET)

(Port ID) 0x01 – 0xFF	(Controller Flags) Bit Field	(Sub-Device) 0x0000 (Root)
(CC) GET_COMMAND	(PID) ENDPOINT_RESPONDERS	(PDL) 0x02
(PD) Endpoint ID (16-bit)		

Response: (GET)



Data Description:
This message returns a packed list of 48-bit UIDs that are associated with the specified Endpoint ID and the List Change Number described in the ENDPOINT_RESPONDER_LIST_CHANGE message.

Endpoint ID: Valid Endpoint IDs for a device can be obtained using the ENDPOINT_LIST message.

If the Endpoint ID is not a valid Endpoint ID then the device shall respond with a NACK Reason Code of NR_ENDPOINT_NUMBER_INVALID.

In the case of an RDMnet Device, NULL_ENDPOINT (defined in [RDMnet] Appendix A) is not a valid Endpoint ID. A device receiving a command with the Endpoint ID field set to NULL_ENDPOINT shall respond with NR_ENDPOINT_NUMBER_INVALID.

List Change Number: The List Change Number is a number used for controllers to track if the list of responder UIDs for a given Endpoint ID has changed. This Change Number shall be incremented by one each time the set of responders for an endpoint changes. The Change Number is an unsigned 32-bit field. This field shall be initialized to 0x00000000 and roll over from 0xFFFFFFFF to 0x00000000.

Note that Background Discovery may add UIDs to the response causing the List Change Number to increment without the controller's knowledge and the subsequent GET: ENPOINT_RESPONDERS response for the new list change number may produce the same list of UIDs.

If there are no devices currently associated with the specified Endpoint, the Parameter Data Length field shall be returned as 0x06 with the Endpoint ID and List Change Number only.

In an ACK_OVERFLOW condition, the Endpoint ID and List Change Number are only sent in the first message. All subsequent ACK_OVERFLOW packets shall only contain the packed list of 48-bit UIDs.
Note: The preferred way to provide asynchronous notification of the responder list change is to queue an ENDPOINT_RESPONDER_LIST_CHANGE response message as opposed to queueing an ENDPOINT_RESPONDERS response message.

6.13 Get Endpoint Responder List Change (ENDPOINT_RESPONDER_LIST_CHANGE)

This parameter returns a unique List Change Number as a means for Controllers to identify if the Endpoint Responder List has changed.

RDMnet Devices shall send an unsolicited ENDPOINT_RESPONDER_LIST_CHANGE response whenever the list of responders for an Endpoint changes. RDM Devices shall queue an ENDPOINT_RESPONDER_LIST_CHANGE response when the list of Responders for an Endpoint changes.

Controller: (GET)

(Port ID) 0x01 – 0xFF	(Controller Flags) Bit Field	(Sub-Device) 0x0000 (Root)
(CC) GET_COMMAND	(PID) ENDPOINT_RESPONDER_LIST_CHANGE	(PDL) 0x02
(PD) Endpoint ID (16-bit)		

Response: (GET)

(Response Type) ACK	(Message Count) 0x01 – 0xFF	(Sub-Device) Copy of Controller SD
(CC) GET_COMMAND_ RESPONSE	(PID) ENDPOINT_RESPONDER_LIST_CHANGE	(PDL) 0x06
(PD) Endpoint ID (16-bit) List Change Number (32-bit)		

Data Description:

Endpoint ID: Valid Endpoint IDs for a device can be obtained using the ENDPOINT_LIST message.

If the Endpoint ID is not a valid Endpoint ID then the device shall respond with a NACK Reason Code of NR_ENDPOINT_NUMBER_INVALID.

In the case of an RDMnet Device, NULL_ENDPOINT (defined in [RDMnet] Appendix A) is not a valid Endpoint ID. A device receiving a command with the Endpoint ID field set to NULL_ENDPOINT shall respond with NR_ENDPOINT_NUMBER_INVALID.

List Change Number:
The Endpoint Device List Change Number is a number used for controllers to track if the list of responder UIDs for a given Endpoint ID has changed. This Change Number shall be incremented by one each time the set of responders for an endpoint changes. The Change Number is an unsigned 32-bit field. This field shall be initialized to 0x0000000 and roll over from 0xFFFFFFFF to 0x00000000.

6.14 Get Binding and Control Fields (BINDING_CONTROL_FIELDS)

This parameter allows a Controller to retrieve the Binding UID and Control Field information that is sent as part of the Discovery Mute (DISC_MUTE) message. See [RDM] Sections 7.6.1 and 7.6.2 for more details. This allows an RDMnet Device to retrieve these fields that are not otherwise available within RDMnet. The DISC_MUTE message is disallowed within RDMnet.

The request message includes the Endpoint ID that contains the RDM responder and the UID of the RDM responder.

If the Endpoint ID does not exist then the device shall respond with a NACK Reason Code of NR_ENDPOINT_NUMBER_INVALID.

In the case of an RDMnet Device, NULL_ENDPOINT (defined in [RDMnet] Appendix A) is not a valid Endpoint ID. A device receiving a command with the Endpoint ID field set to NULL_ENDPOINT shall respond with NR_ENDPOINT_NUMBER_INVALID.

Controller: (GET)

(Port ID) 0x01 – 0xFF	(Controller Flags) Bit Field	(Sub-Device) 0x0000 (Root)
(CC) GET_COMMAND	(PID) BINDING_CONTROL_FIELDS	(PDL) 0x08
(PD) Endpoint ID (16-bit) UID (48-bit)		

Response: (GET)

(Response Type) ACK	(Message Count) 0x01 – 0xFF	(Sub-Device) Copy of Controller SD
(CC) GET_COMMAND_ RESPONSE	(PID) BINDING_CONTROL_FIELDS	(PDL) 0x10
(PD) Endpoint ID (16-bit) UID (48-bit) Control Field (16-bit) Binding UID (48-bit)		

Data Description:

Endpoint ID: The Endpoint ID that the requested UID is associated with. If the Endpoint ID is not a valid Endpoint ID then the device shall respond with a NACK Reason Code of NR_ENDPOINT_NUMBER_INVALID.

UID: The UID for the target responder about which the Control Fields and Binding UID are being requested. If the requested UID does not exist for that Endpoint then the device shall respond with a NACK Reason Code of NR_UNKNOWN_UID.

Control Field: The Control Field for the UID being requested. This information would have been received by the device from the Discovery Mute (DISC_MUTE) message (See [RDM] Section 7.6.1).

Binding UID: The Binding UID Field for the UID being requested. This information would have been received by the device from the Discovery Mute (DISC_MUTE) message (See [RDM] Section 7.6.2). If there was no Binding UID information present from the DISC_MUTE response of the device then the Binding UID shall be set to 0x000000000000 in this message.

6.15 Get Manufacturer Endpoint Label (MANUFACTURER_ENDPOINT_LABEL)

This parameter provides a means of getting a descriptive manufacturer label for each Endpoint on a device.

Controller: (GET)

(Port ID) 0x01 – 0xFF	(Controller Flags) Bit Field	(Sub-Device) 0x0000 (Root)
(CC) GET_COMMAND	(PID) MANUFACTURER_ENDPOINT_LABEL	(PDL) 0x02
(PD) Endpoint ID (16-bit)		

Response: (GET)

(Response Type) ACK	(Message Count) 0x01 – 0xFF	(Sub-Device) Copy of Controller SD
(CC) GET_COMMAND_ RESPONSE	(PID) MANUFACTURER_ENDPOINT_LABEL	(PDL) 2-34 (2 + Number of bytes sent)
(PD) Endpoint ID (16-bit) Text Label. Up to 32 bytes.		

Data Description:

Endpoint ID: Valid Endpoint IDs for a device can be obtained using the ENDPOINT_LIST message. An Endpoint ID of BROADCAST_ENDPOINT shall only be used as an All Call in a SET_COMMAND message to set all Endpoints to the same value.

If the Endpoint ID is not a valid Endpoint ID then the device shall respond with a NACK Reason Code of NR_ENDPOINT_NUMBER_INVALID.

In the case of an RDMnet Device, NULL_ENDPOINT (defined in [RDMnet] Appendix A) is not a valid Endpoint ID. A device receiving a command with the Endpoint ID field set to NULL_ENDPOINT shall respond with NR_ENDPOINT_NUMBER_INVALID.

Text Label: The text label shall observe the requirements of [RDM] Section 10.1 for text fields. The field may be up to 32 bytes.

6.16 Get/Set Endpoint Network Stream (ENDPOINT_NETWORK_STREAM)

This parameter is used to assign an Endpoint on a device to operate with a specific Network Stream Type, as defined in Table A-8. When the Endpoint is selected as an input, the Network Stream Type defines the network data that will be generated in response to received DMX512.
When the Endpoint is selected as an output, the Network Stream Type defines the network data that will be used to generate the DMX512 output.

It should be noted that changing the Network Stream Type may change the valid range of Universe Numbers.

Controller: (GET)

(Port ID) 0x01 – 0xFF	(Controller Flags) Bit Field	(Sub-Device) 0x0000 (Root)
(CC) GET_COMMAND	(PID) ENDPOINT_NETWORK_STREAM	(PDL) 0x02
(PD) Endpoint ID (16-bit)		

Response: (GET)

(Response Type) ACK	(Message Count) 0x01 – 0xFF	(Sub-Device) Copy of Controller SD
(CC) GET_COMMAND_ RESPONSE	(PID) ENDPOINT_NETWORK_STREAM	(PDL) 0x04
(PD) Endpoint ID (16-bit) Network Stream Type (16-bit)		

Controller: (SET)

(Port ID) 0x01 – 0xFF	(Controller Flags) Bit Field	(Sub-Device) 0x0000 (Root)
(CC) SET_COMMAND	(PID) ENDPOINT_NETWORK_STREAM	(PDL) 0x04
(PD) Endpoint ID (16-bit) Network Stream Type (16-bit)		

Response: (SET)

(Response Type) ACK	(Message Count) 0x01 – 0xFF	(Sub-Device) Copy of Controller SD
(CC) SET_COMMAND_ RESPONSE	(PID) ENDPOINT_NETWORK_STREAM	(PDL) 0x02
(PD) Endpoint ID (16-bit)		

Data Description:

Endpoint ID: Valid Endpoint IDs for a device can be obtained using the ENDPOINT_LIST message. An Endpoint ID of BROADCAST_ENDPOINT shall only be used as an All Call in a SET_COMMAND message to set all Endpoints to the same value.

If the Endpoint ID is not a valid Endpoint ID then the device shall respond with a NACK Reason Code of NR_ENDPOINT_NUMBER_INVALID.

In the case of an RDMnet Device, NULL_ENDPOINT (defined in [RDMnet] Appendix A) is not a valid Endpoint ID. A device receiving a command with the Endpoint ID field set to NULL_ENDPOINT shall respond with NR_ENDPOINT_NUMBER_INVALID.

Network Stream Type: This field indicates the Network Stream Type as defined in Table A-8.
If the Network Stream Type is not supported the device shall respond with a NACK Reason Code of NR_INVALID_NETWORK_STREAM_TYPE.

6.17 Get Endpoint Network Stream Description (ENDPOINT_NETWORK_STREAM_DESCRIPTION)

This parameter is used to get a descriptive text label for a given manufacturer specific Network Stream Type (as defined in Table A-8 : Network Stream Codes

The label may be up to 32 bytes.

Controller: (GET)

(Port ID) 0x01 – 0xFF	(Controller Flags) Bit Field	(Sub-Device) 0x0000 (Root)
(CC) GET_COMMAND	(PID) ENDPOINT_NETWORK_STREAM_DESCRIPTION	(PDL) 0x02
(PD) Network Stream Type (16-bit)		

Response: (GET)

(Response Type) ACK	(Message Count) 0x01 – 0xFF	(Sub-Device) Copy of Controller SD
(CC) GET_COMMAND_RESPONSE	(PID) ENDPOINT_NETWORK_STREAM_DESCRIPTION	(PDL) 2 – 34 (2 + Number of bytes sent)
(PD) Network Stream Type (16-bit) Text Label. Up to 32 bytes.		

Data Description:

Network Stream Type Requested:
This field indicates the Network Stream Type requested from the GET_COMMAND message. Values in the range 0x8000 – 0xfffe are valid. If the Network Stream Type is not valid then the device shall respond with a NACK Reason Code of NR_INVALID_NETWORK_STREAM_TYPE.

Text Label:
The text field shall observe the requirements of [RDM] Section 10.1 for text fields. The field may be up to 32 bytes.

6.18 Get Supported Network Stream Types (SUPPORTED_NETWORK_STREAMS)

This parameter returns a packed list to indicate which Network Stream Types that are supported by a given Endpoint ID number.

Controller: (GET)

(Port ID) 0x01 – 0xFF	(Controller Flags) Bit Field	(Sub-Device) 0x0000 (Root)
(CC) GET_COMMAND	(PID) SUPPORTED_NETWORK_STREAMS	(PDL) 0x02
(PD) Endpoint ID (16-bit)		

Response: (GET)

(Response Type) ACK/ACK_OVERFLOW	(Message Count) 0x01 – 0xFF	(Sub-Device) Copy of Controller SD
(CC) GET_COMMAND_ RESPONSE	(PID) SUPPORTED_NETWORK_STREAMS	(PDL) First: 0x05 – 0xE6 Subsequent: 0x03 – 0xE7
(PD) *--- First Response Only ---* Endpoint ID (16-bit) Packed list of 24-bit structures containing the following: Flags (8-bit) Packed list of Network Stream Types (16-bit)		

Data Description:
This message returns a packed list of 16-bit Network Stream Types that are supported by the specified Endpoint ID.

Endpoint ID: Valid Endpoint IDs for a device can be obtained using the ENDPOINT_LIST message.

If the Endpoint ID is not a valid Endpoint ID then the device shall respond with a NACK Reason Code of NR_ENDPOINT_NUMBER_INVALID.

In the case of an RDMnet Device, NULL_ENDPOINT (defined in [RDMnet] Appendix A) is not a valid Endpoint ID. A device receiving a command with the Endpoint ID field set to NULL_ENDPOINT shall respond with NR_ENDPOINT_NUMBER_INVALID.

Flags: This field indicates the Endpoint Modes for which the Network Stream Type is supported, as defined in Table 6-1.

Table 6-1: Flags

Controller Flag	Name	Capabilities
Bit 0 (0x01)	Input	The Network Stream Type is supported for ENDPOINT_MODE_INPUT.
Bit 1 (0x02)	Output	The Network Stream Type is supported for ENDPOINT_MODE_OUTPUT.
Bit 2 (0x04)	RESERVED	Not used, set to zero.
Bit 3 (0x08)	RESERVED	Not used, set to zero.
Bit 4 (0x10)	RESERVED	Not used, set to zero.
Bit 5 (0x20)	RESERVED	Not used, set to zero.
Bit 6 (0x40)	RESERVED	Not used, set to zero.
Bit 7 (0x80)	RESERVED	Not used, set to zero.

Network Stream Type: This field indicates the Network Stream Type as defined in Table A-8.

7 RDM Parameter Messages for Distributed Queued/Status Management

The following RDM Parameter Messages are intended for managing Queued and Status messages on [RDM] devices.

7.1 Get/Set Background Queued/Status Message Collection Policy (BACKGROUND_QUEUED_STATUS_POLICY)

This parameter is used to set a background collection policy for managing collection of Queued messages from an RDM device on an E1.20 network.

Background collection policies are a flexible framework for Splitters, Proxies, and RDMnet Gateways to implement different configuration policies for collection of Queued and Status Messages from RDM Devices. Possible collection strategies might include more frequent collection of higher priority messages, or overall higher or lower collection frequencies.

See [RDM] Section 10.3 for more information on Queued and Status messages.

Controller: (GET)

(Port ID) 0x01 – 0xFF	(Controller Flags) Bit Field	(Sub-Device) 0x0000 (Root) or 0x0001-0x0200
(CC) GET_COMMAND	(PID) BACKGROUND_QUEUED_STATUS_POLICY	(PDL) 0x00
(PD)		

Response: (GET)

(Response Type) ACK	(Message Count) 0x01 – 0xFF	(Sub-Device) Copy of Controller SD
(CC) GET_COMMAND_RESPONSE	(PID) BACKGROUND_QUEUED_STATUS_POLICY	(PDL) 0x02
(PD) Current Policy Setting | # of Policy Settings		

Controller: (SET)

(Port ID) 0x01 – 0xFF	(Controller Flags) Bit Field	(Sub-Device) 0x0000 (Root) or 0x0001-0x0200 or 0xFFFF
(CC) SET_COMMAND	(PID) BACKGROUND_QUEUED_STATUS_POLICY	(PDL) 0x01
(PD) Policy Setting		

Response (SET):

(Response Type) ACK	(Message Count) 0x01 – 0xFF	(Sub-Device) Copy of Controller SD
(CC) SET_COMMAND_RESPONSE	(PID) BACKGROUND_QUEUED_STATUS_POLICY	(PDL) 0x00

(PD) Not Present

Data Description:

Policy Setting:

The GET_COMMAND_RESPONSE message includes the current policy setting as well as the total number of policy settings available. These policy settings shall be consecutively numbered within the responder starting from 0.

Text descriptions can be retrieved using the BACKGROUND_QUEUED_STATUS_POLICY_DESCRIPTION parameter. Responders that support BACKGROUND_QUEUED_STATUS_POLICY shall also support the BACKGROUND_QUEUED_STATUS_POLICY_DESCRIPTION message.

The first four Policy Settings available shall be as detailed in Table 6-1. This table shall be considered informative in nature since these states are defined in [RDM]. These policy types shall follow the behavior as defined in [RDM] Section 10.3.2.2. Manufacturers may create additional collection policies beyond those listed in in Table 6-1 but must support this minimum set in order to preserve consistent interoperability between products. The collection intervals for these specified policy settings may follow the manufacturer's default collection interval timing.

Table 6-1: Policy Setting Types (Informative)

Policy Type	Policy Setting #
STATUS_NONE	0x00
STATUS_ADVISORY	0x02
STATUS_WARNING	0x03
STATUS_ERROR	0x04

Note that STATUS_NONE was not allowed prior to [RDM-2025] and so some devices may NACK a QUEUED_MESSAGE request using type STATUS_NONE as an argument.

7.2 Get/Set Endpoint Background Queued/Status Message Collection Policy (ENDPOINT_BACKGROUND_QUEUED_STATUS_POLICY)

This parameter is used to set a background collection policy for managing collection of Queued Messages from an RDM device on an E1.20 network for a specific Endpoint ID.

Background collection policies are a flexible framework for Splitters, Proxies, and RDMnet Gateways to implement different configuration policies for collection of Queued and Status Messages from RDM Devices. Possible collection strategies might include more frequent collection of higher priority messages, or overall higher or lower collection frequencies.

See [RDM] Section 10.3 for more information on Queued and Status messages.

Controller: (GET)

(Port ID) 0x01 – 0xFF	(Controller Flags) Bit Field	(Sub-Device) 0x0000 (Root) or 0x0001-0x0200
(CC) GET_COMMAND	(PID) ENDPOINT_BACKGROUND_QUEUED_STATUS_POLICY	(PDL) 0x02
(PD) Endpoint ID (16-bit)		

Response: (GET)

(Response Type) ACK	(Message Count) 0x01 – 0xFF	(Sub-Device) Copy of Controller SD
(CC) GET_COMMAND_RESPONSE	(PID) ENDPOINT_BACKGROUND_QUEUED_STATUS_POLICY	(PDL) 0x04
(PD) Endpoint ID (16-bit) Current Policy Setting | # of Policy Settings		

Controller: (SET)

(Port ID) 0x01 – 0xFF	(Controller Flags) Bit Field	(Sub-Device) 0x0000 (Root) or 0x0001-0x0200 or 0xFFFF
(CC) SET_COMMAND	(PID) ENDPOINT_BACKGROUND_QUEUED_STATUS_POLICY	(PDL) 0x03
(PD) Endpoint ID (16-bit) Policy Setting		

Response (SET):

(Response Type) ACK	(Message Count) 0x01 – 0xFF	(Sub-Device) Copy of Controller SD
(CC) SET_COMMAND_RESPONSE	(PID) ENDPOINT_BACKGROUND_QUEUED_STATUS_POLICY	(PDL) 0x00
(PD) Not Present		

Data Description:

Endpoint ID: The Endpoint ID that the requested UID is associated with. If the Endpoint ID is not a valid Endpoint ID then the device shall respond with a NACK Reason Code of NR_ENDPOINT_NUMBER_INVALID.

Policy Setting:

The GET_COMMAND_RESPONSE message includes the current policy setting as well as the total number of policy settings available. These policy settings shall be consecutively numbered within the responder starting from 0.

Text descriptions can be retrieved using the BACKGROUND_QUEUED_STATUS_POLICY_DESCRIPTION Parameter. Responders that support ENDPOINT_BACKGROUND_QUEUED_STATUS_POLICY shall also support the BACKGROUND_QUEUED_STATUS_POLICY_DESCRIPTION message.

The first four Policy Settings available shall be as detailed in Table 6-1. This table shall be considered informative in nature since these states are defined in [RDM]. These policy types shall follow the behavior as defined in [RDM] Section 10.3.2.2. Manufacturers may create additional collection policies beyond those listed in in Table 6-1 but must support this minimum set in order to preserve consistent interoperability between products. The collection intervals for these specified policy settings may follow the manufacturer's default collection interval timing.

Note that STATUS_NONE is not allowed in prior versions of [RDM] so some devices may NACK a QUEUED_MESSAGE request using type STATUS_NONE as an argument. This is expected to be allowed in

future revisions of [RDM]. It is included here as a reference for future compatibility at such time it becomes allowed in [RDM].

7.3 Get Background Queued/Status Message Policy Description (BACKGROUND_QUEUED_STATUS_POLICY_DESCRIPTION)

This parameter is used to get a descriptive text label for a given Background Collection Policy Number. The label may be up to 32 bytes.

Controller: (GET)

(Port ID) 0x01 – 0xFF	(Controller Flags) Bit Field	(Sub-Device) 0x0000 (Root) or 0x0001-0x0200
(CC) GET_COMMAND	(PID) BACKGROUND_QUEUED_STATUS_POLICY_DESCRIPTION	(PDL) 0x01
(PD) Policy # Requested		

Response: (GET)

(Response Type) ACK	(Message Count) 0x01 – 0xFF	(Sub-Device) Copy of Controller SD
(CC) GET_COMMAND_RESPONSE	(PID) BACKGROUND_QUEUED_STATUS_POLICY_DESCRIPTION	(PDL) 1 – 33 (1 + Number of bytes sent)
(PD) Policy # Requested Text Label. Up to 32 bytes.		

Data Description:
The Response Data contains the Policy Number Requested, along with a text label.

The text label returned for the first five policies shall follow Table 6-2.

Table 6-2: Policy Description

Policy Setting #	Policy Text Description
0x00	Status Type None
0x01	Status Type Policy Undefined
0x02	Status Type Advisory and higher
0x03	Status Type Warning and higher
0x04	Status Type Error

Text Label:
The text field shall observe the requirements of [RDM] Section 10.1 for text fields. The field may be up to 32 bytes.

Appendix A: Defined Parameters (Normative)

Table A-1: RDM Parameter ID Defines

GET Allowed	SET Allowed	RDM Parameter IDs (Slot 21-22)	Value	Required By**:	Comment
		Category – RDMnet and RDM Splitter Management			
✓		ENDPOINT_LIST	0x0900	D/G/S/P	
✓		ENDPOINT_LIST_CHANGE	0x0901	D*/G*/S*/P*	*Support required if the list of endpoints can change.
✓	✓	IDENTIFY_ENDPOINT	0x0902		
✓	✓	ENDPOINT_TO_UNIVERSE	0x0903	D*/G*	* Support required if the device supports patching Endpoints to E1.31 universes.
✓	✓	ENDPOINT_MODE	0x0904		
✓	✓	ENDPOINT_LABEL	0x0905		
✓	✓	RDM_TRAFFIC_ENABLE	0x0906		
✓	✓	DISCOVERY_STATE	0x0907	G*/P*	* Support required only if the device supports E1.20.
✓	✓	BACKGROUND_DISCOVERY	0x0908		
✓	✓	ENDPOINT_TIMING	0x0909		
✓		ENDPOINT_TIMING_DESCRIPTION	0x090A	G*/S*/P*	* Support required only if ENDPOINT_TIMING is supported.
✓		ENDPOINT_RESPONDERS	0x090B	D/G/S/P	
✓		ENDPOINT_RESPONDER_LIST_ CHANGE	0x090C	D*/G*/S*/P*	* Support required if the responder list could change.
✓		BINDING_CONTROL_FIELDS	0x090D	G/S/P	
✓	✓	BACKGROUND_QUEUED_STATUS_ POLICY	0x090E		
✓		BACKGROUND_QUEUED_STATUS_ POLICY_DESCRIPTION	0x090F	G*/P*	* Support required only if BACKGROUND_ QUEUED_STATUS_ POLICY or ENDPOINT_ BACKGROUND_ QUEUED_STATUS_ POLICY is supported.
✓		MANUFACTURER_ENDPOINT_LABEL	0x????		
✓	✓	ENDPOINT_BACKGROUND_QUEUED _STATUS_ POLICY	0x????		
✓	✓	ENDPOINT_NETWORK_STREAM	0x????		
✓		ENDPOINT_NETWORK_STREAM_DE SCRIPTION	0x????		
✓		SUPPORTED_NETWORK_STREAMS	0x????		

**Required By column denotes support required by the following device classifications: ‘D’ RDMnet Device, ‘G’ RDMnet Gateway, ‘S’ Splitter, and ‘P’ Proxy. If a device has functionality of multiple classifications then ‘required support’ applies to each classification supported.

Table A-3: Discovery State Defines

Discovery State Defines	Value	Valid in GET_COMMAND_ RESPONSE	Valid in SET_COMMAND
DISCOVERY_INCOMPLETE	0x00	Yes, indicates Discovery has never been run or the previous discovery did not run to completion.	No
DISCOVERY_INCREMENTAL	0x01	Yes, device is currently running incremental discovery	Yes, device shall only discover current unmuted devices by running a discovery without any mute or unmute commands being sent beforehand.
DISCOVERY_FULL	0x02	Yes, device is currently running full discovery.	Yes, device shall clear the Endpoint Device List, unmute all and perform discovery.
DISCOVERY_NOT_ACTIVE	0x04	Yes, discovery is not currently running but has been successfully completed in the past.	Yes, device shall terminate any discovery routine currently running. The Endpoint Device List shall not be cleared.
Manufacturer-Specific States	0x80 – 0xDF		

Table A-4: Discovery Status Defines

Discovery Status Defines	Value	Comment
DISCOVERY_COUNT_INCOMPLETE	0x0000	See Section 6.8
DISCOVERY_COUNT_UNKNOWN	0xFFFF	See Section 6.8

Table A-5: Endpoint Mode Defines

Endpoint Mode Defines	Value	Comment
ENDPOINT_MODE_DISABLED	0x00	Endpoint is disabled for all traffic.
ENDPOINT_MODE_INPUT	0x01	Endpoint is configured as an Input, accepting DMX512 communication.
ENDPOINT_MODE_OUTPUT	0x02	Endpoint is configured as an Output generating DMX512 communication.

Table A-6: Endpoint Types

Endpoint Type Defines	Value	Comment
ENDPOINT_TYPE_VIRTUAL	0x00	See [RDMnet] Section 7.1.6.2
ENDPOINT_TYPE_PHYSICAL	0x01	See [RDMnet] Section 7.1.6.1

Table A-7: Additional Response NACK Reason Codes

Additional Response NACK Reason Codes*	Value	Comment
NR_ACTION_NOT_SUPPORTED	0x000B	The specified action is not supported.
NR_ENDPOINT_NUMBER_INVALID	0x000C	The specified endpoint is invalid.
NR_INVALID_ENDPOINT_MODE	0x000D	The specified endpoint is in an invalid Endpoint Mode for the requested action.
NR_UNKNOWN_UID	0x000E	The specified UID is not recognized.
NR_INVALID_NETWORK_STREAM_MODE		The specified endpoint does not support the requested Network Stream Mode in its current configuration.
		*These are in addition to NACK Reason Codes defined in Table A-17 of E1.20.

Table A-8 : Network Stream Codes

Network Stream Code	Value	Comment
NS_NOT_STREAMING	0x0000	This endpoint is not sending or receiving a network DMX512 stream.
NS_SACN	0x0001	This endpoint is sending or receiving a network DMX512 stream via sACN.
NS_RESERVED	0x0002-0x3fff	Reserved for future use by this standard.
NS_ART_NET	0x4000	This endpoint is sending or receiving a network DMX512 stream via [Art-Net].
NS_KINET	0x4001	This endpoint is sending or receiving a network DMX512 stream via [KiNET].
NS_RESERVED	0x4002-0x7fff	Reserved for future use by this standard.
NS_MANUFACTURER_SPECIFIC	0x8000-0xfffe	Manufacturer specific network streams.
NS_OTHER_MEANS	0xffff	The network stream is unavailable or set by other means.