DRAFT

BSR E1.37-8 – 202x

Entertainment Technology – IPv4 and IPv6 Network Configuration Messages for E1.20 RDM

Approved as an American National Standard by the ANSI Board of Standards Review on

_____

CP/2024-1027r0

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, recommended practice or an American National Standard developed by ESTA 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 his or her 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
+1-212-244-1505
standards@esta.org
http://www.esta.org/

The ESTA Technical Standards Program

The ESTA 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 within the entertainment industry by creating recommended practices and standards, monitoring standards issues around the world on behalf of our members, and improving communications and safety within the industry. ESTA works closely with the technical standards efforts of other organizations within our industry, including USITT and VPLT, as well as representing the interests of ESTA members to ANSI, UL, and the NFPA. The Technical Standards Program is accredited by the American National Standards Institute.

The Technical Standards Council (TSC) was established to oversee and coordinate the Technical Standards Program. Made up of individuals experienced in standards-making work from throughout our industry, the Council 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 Council and its working groups as well as maintain a “Standards Watch” on behalf of members. Working groups include: Control Protocols, Electrical Power, Event Safety, Floors, Fog and Smoke, Followspot Position, Photometrics, Rigging, Stage Machinery, and Weapons Safety.

ESTA encourages active participation in the Technical Standards Program. There are several ways to become involved. If you would like to become a member of an existing working group, as have over four hundred people, you must complete an application which is available from the ESTA office. Your application is subject to approval by the working group and you will be required to actively participate in the work of the group. This includes responding to letter ballots and attending meetings. Membership in ESTA is not a requirement. You can also become involved by requesting that the TSC develop a standard or a recommended practice in an area of concern to you.

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.

Contact information

Technical Standards Manager
Richard J. Nix
ESTA
271 Cadman Plaza; PO Box 23200
Brooklyn, NY 11202-3200
USA
+1-212-244-1505
richard.nix@esta.org

Senior Technical Standards Manager
Karl G. Ruling
ESTA
271 Cadman Plaza; PO Box 23200
Brooklyn, NY 11202-3200
USA
+1-212-244-1505
karl.ruling@esta.org

Technical Standards Council Chairpersons

Mike Garl
Mike Garl Consulting LLC
+1-865-389-4371
mike@mikegarlconsulting.com

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

Control Protocols Working Group Chairpersons

Michael Lay
Candela Controls, Inc
+1-407-654-2420
mlay@candelacontrols.com

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

Table of Contents

List of Tables

List of Figures

1 Introduction

1.1 E1.20 Basic Features

The 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 as detailed in Annex B [DMX512].

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.

1.2 Overview & Scope

This document provides additional Get/Set parameter messages for use with the E1.20 Remote Device Management protocol. The areas covered are:

- Network Interface Configuration Messages (Section 6)
- IPv4 Network Configuration Messages (Section 7)
- IPv6 Network Configuration Messages (Section 8)
- DNS Configuration Messages (Section 9)
- VLAN Configuration Messages (Section 10)

To ensure continued communication between a Controller and a Responder while making changes a session mechanism is defined in Section 5.

1.3 ANSI E1.37-2 Deprecation

This document supersedes ANSI E1.37-2 – 2015 (R2021) Entertainment Technology - Additional Message Sets for E1.20 (RDM) – Part 2, IPv4 & DNS Configuration Messages. ANSI E1.37-2 should not be considered for new designs. Controllers may wish to continue to support ANSI E1.37-2 to support legacy device implementations.

2 Normative References

[802] 802-2014 Local and Metropolitan Area Networks: Overview and Architecture

This standard Is maintained by the IEEE.

IEEE
3 Park Avenue, 17th Floor
New York, NY 10016-5997
USA
Phone: +1-212-419-7900
https://ieee.org

[802.3] 802.3-2022 IEEE Standard for Ethernet

This standard Is maintained by the IEEE.

[802c] 802c-2017 Local and Metropolitan Area Networks: Overview and Architecture – Amendment 2: Local Medium Access Control (MAC) Address Usage

This standard Is maintained by the IEEE.

[802f] 802f-2023 Local and Metropolitan Area Networks: Overview and Architecture – Amendment 3: YANG Data Model for EtherTypes

This standard Is maintained by the IEEE.

[DHCP] RFC 2131 Dynamic Host Configuration Protocol

This standard is maintained by the IETF.

IETF Secretariat
c/o Association Management Solutions, LLC (AMS)
5177 Brandin Court
Fremont, California 94538
USA
Phone: +1-510-492-4080
http://www.ietf.org

[DMX] ANSI E1.11-2008 (R2018) Entertainment Technology – USITT DMX512-A, Asynchronous Serial Digital Data Transmission Standard for Controlling Lighting Equipment and Accessories

This standard is maintained by Entertainment Services and Technology Association.

ESTA
271 Cadman Plaza
PO Box 23200
Brooklyn, NY 11202-3200
USA
Phone: 1-212-244-1505
standards@esta.org
http://www.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
USA
+1-212-642-4900
http://www.ansi.org

[DNS] RFC 1034 Domain Names – Concepts and Facilities

This standard is maintained by the IETF.

[DNSImplementation] RFC 1035 Domain Names – Implementation and Specification

This standard is maintained by the IETF.

[Interface] RFC 3493 Basic Socket Interface Extensions for IPv6

This standard is maintained by the IETF.

[IPv4] RFC 791 Internet Protocol

This standard is commonly referred to as Internet Protocol Version 4.

This standard is maintained by the IETF.

[IPv6-Addressing] RFC 4291 IP Version 6 Addressing Architecture

This standard is maintained by the IETF.

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

This standard is maintained by ESTA.

[RDM] E1.20-2010* Entertainment Technology – Remote Device Management over DMX512 networks

This standard is maintained by ESTA.

* to be updated with approved revised version

[sACN] ANSI E1.31-2018 Entertainment Technology - Lightweight streaming protocol for transport of DMX512 using ACN

This standard is maintained by ESTA.

[URI] RFC 3986 Uniform Resource Identifier (URI): Generic Syntax

This standard is maintained by the IETF.

[VLAN] 802.1Q-2016/Cor 1-2017 Bridges and bridged networks, Corrigendum 1: Technical and editorial corrections

This standard is maintained by the IEEE.

3 General

3.1 General

These parameter messages are for general IP networking configuration and may be used across any type of device and not limited to any specific class of products that contain this functionality.

3.2 Sub-Device Handling

Refer to E1.20 Section 9 for information on Sub-Device usage. This document does not change or modify the requirements stated in E1.20. Requirements stated in this document are in addition to the stated 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]. This includes IPv4 Addresses.

4 Interface Hierarchy Model

4.1 Overview

The interface hierarchy model is designed to support a wide range of network configurations and enables parity with the ways in which interfaces are modelled in most popular operating systems. The model supports complex hierarchies of physical and virtual interfaces to any depth, and the IPv4 and IPv6 addresses assigned to them.

4.2 Interface Model Properties

Every physical or virtual interface present on a Responder is identified using a unique identifier as defined in Section 6.1. By retrieving the identifier of an interface's parent, Controllers can establish the hierarchy of interfaces on a Responder.

Several properties for interfaces can be retrieved or set including:

- Enable/Disable
- Hardware Address
- Label (Fixed and User Assignable)
- Type (Physical or Virtual)
- Line Speed and Utilization
- Duplex Mode
- Energy Efficient Ethernet

The model provides mechanisms to get and set an arbitrary number of IPv4 and IPv6 addresses on both physical and virtual interfaces, including:

- Address
- Link-Local Address (IPv6 only)
- Gateway
- DNS Address
- Address Assignment Mode and Fallback
- DHCP Server Address

DNS information for an interface can be provided such as:

- Hostname (Label)
- Domain
- Search Domain

The model also supports discovering the VLAN capabilities of interfaces, adding and removing virtual interfaces (VLANs) to/from these interfaces, and labelling any VLANs present.

4.3 Sample Interface Hierarchy [Informative]

Due to the diverse hardware and operating systems utilized by Responders, this standard is designed to support a broad spectrum of interface hierarchies.

The example below illustrates a Responder with three physical interfaces (two wired and one wireless). The two wired interfaces have assigned IPv4 and IPv6 addresses, as well as child interfaces representing VLAN assignments, which also have their own IP addresses:

- Physical Interface 1 (Wired)
    - IPv4 Address 1
    - …
    - IPv4 Address n
    - IPv6 Link Local Address
    - IPv6 Address 1
    - …
    - IPv6 Address n
    - VLAN Interface 10
        - IPv4 Address 1
        - …
        - IPv4 Address n
        - IPv6 Link Local Address
        - IPv6 Address 1
        - …
        - IPv6 Address n
- Physical Interface 2 (Wired)
    - IPv4 Address 1
    - …
    - IPv4 Address n
    - IPv6 Link Local Address
    - IPv6 Address 1
    - VLAN Interface 1
        - IPv4 Address 1
        - …
        - IPv4 Address n
        - IPv6 Link Local Address
        - IPv6 Address 1
        - …
        - IPv6 Address n
    - VLAN Interface 11
        - IPv4 Address 1
        - …
        - IPv4 Address n
- Physical Interface 3 (Wireless)
    - No IP Assigned

5 Session Messages

To ensure continued communication between a Controller and a Responder, a session is used to make and commit changes.

All SET_COMMAND parameters defined in this document except SET:IFC_SESSION_CONTROL shall be applied atomically using a session. This section describes how a session is established and used.

5.1 Get/Set Session Control (IFC_SESSION_CONTROL)

This parameter is used to manage a session to make configuration changes to one or more interfaces on a Responder.

All SET_COMMAND parameters defined in this document except SET:IFC_SESSION_CONTROL shall be set together to ensure continued communication between a Controller and a Responder. A session is used to make and commit changes.

Process

When a Controller is ready to transmit changes to a Responder using any of the SET_COMMAND parameters in this document it shall do the following:

1. Request a Session ID from the Responder using GET:IFC_SESSION_CONTROL.
2. Make each change required, including the received Session ID with each message.
3. Either commit or discard changes.
4. Confirm applied changes within the time specified (not required for discard).

Responders shall only maintain one active session, i.e. they shall only issue a Session ID if the previous session has ended. Session IDs shall be valid for SESSION_WINDOW from the time they are issued.

If the request for a Session ID returns a NACK with a NACK Reason Code of NR_INTERFACE_INVALID_SESSION, this indicates another session is active, and the Controller should try again later.

When a Responder with no active session receives a request for a Session ID using GET_COMMAND it shall do the following:

1. Issue a GET_COMMAND_RESPONSE with a Session ID and the maximum time it will take to apply any changes.
2. Receive and cache any messages using the Session ID provided.
3. If so commanded, either commit the changes in the order they were received or discard the changes if so commanded.
4. If a SET_COMMAND with a Session Operation of CONFIRM is not received within the time specified with the commit, roll back to previous configuration.

Note: It is expected that Controllers wishing to make changes requiring a Session ID should stage such changes in a user interface and offer users a method to apply these changes, at which point the process above is initiated.

More detailed requirements and information can be found under Data Description.

An example of a successful session process is shown in Figure 5-1.

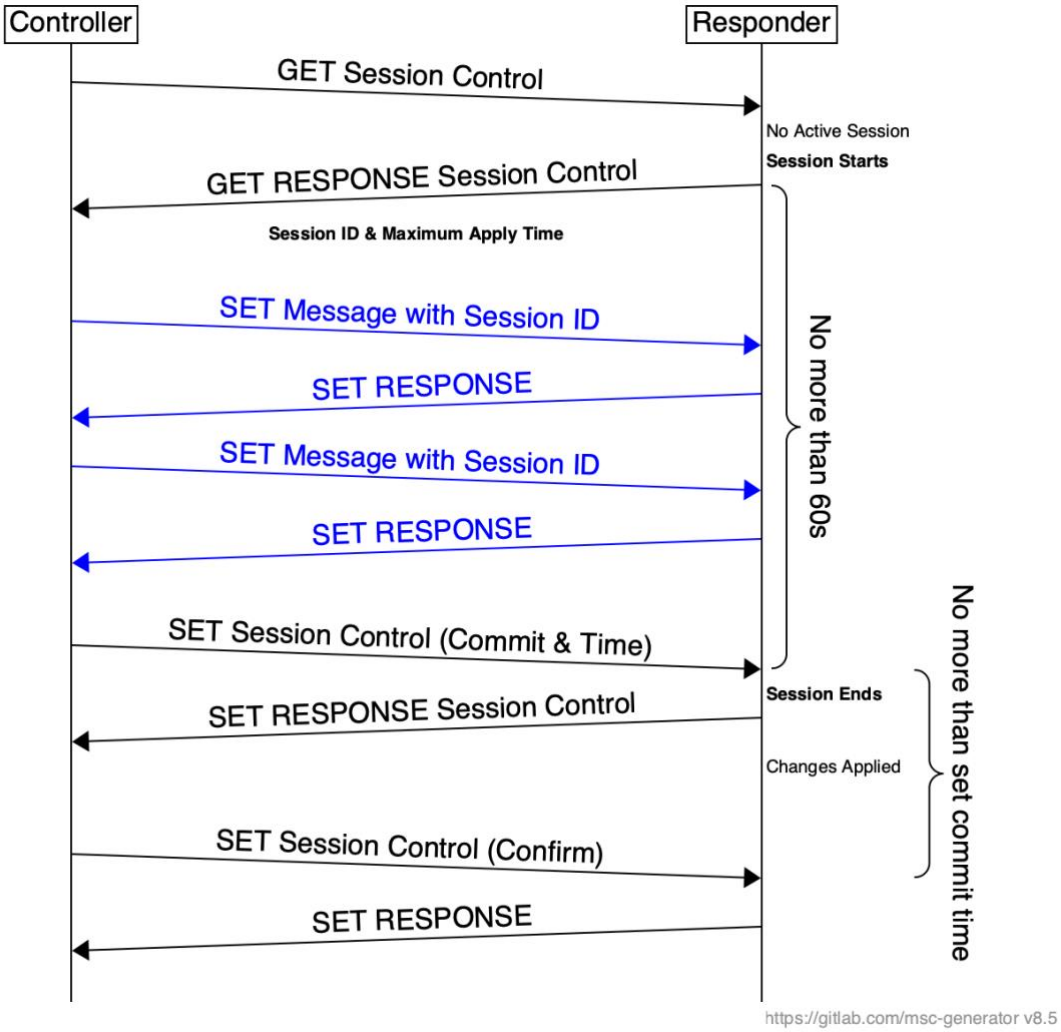


Figure 5-1: Successful Session Process

An example of a session process resulting in a roll back is shown in Figure 5-2.

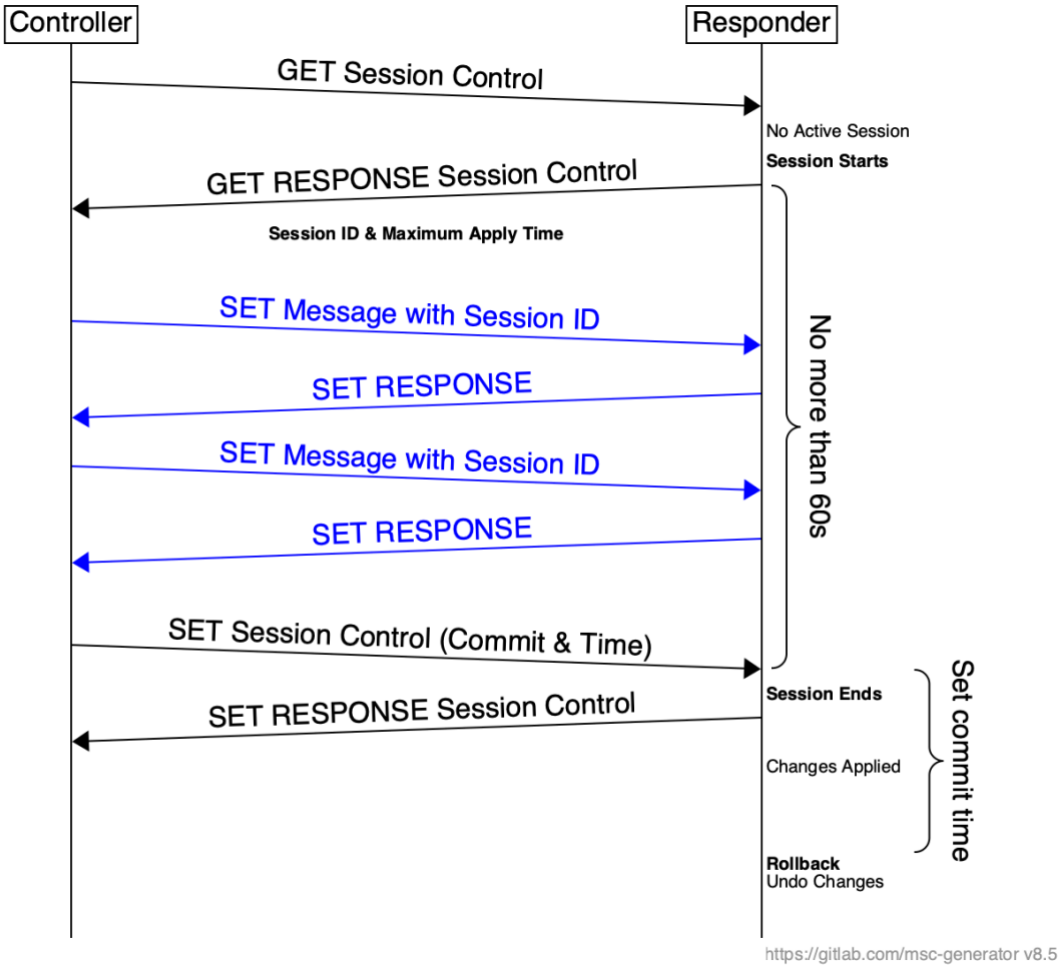


Figure 5-2: Rolled Back Session Process

Controller: (GET)

(Port ID) 0x01 – 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001-0xFFF0
(CC) GET_COMMAND	(PID) IFC_SESSION_CONTROL	(PDL) 0x00
(PD) Not Present		

Response:

(Response Type) ACK	(Message Count) 0x00 – 0xFF	(Sub-Device) Copy of Controller SD
(CC) GET_COMMAND_ RESPONSE	(PID) IFC_SESSION_CONTROL	(PDL) 0x04
(PD) Session ID (16-bit) Time (16-bit)		

Controller: (SET)

(Port ID) 0x01 - 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001-0xFFF0
(CC) SET_COMMAND	(PID) IFC_SESSION_CONTROL	(PDL) 0x05
(PD) Session ID (16-bit) Session Operations (8-bit) Time (16-bit)		

Response:

(Response Type) ACK	(Message Count) 0x00 - 0xFF	(Sub-Device) Copy of Controller SD
(CC) SET_COMMAND_ RESPONSE	(PID) IFC_SESSION_CONTROL	(PDL) 0x00
(PD) Not Present		

Data Description:

Session ID:
A 16-bit session identifier which is used to identify messages that belong to an atomic session.

For GET_COMMAND_RESPONSE, a session identifier which is valid for SESSION_WINDOW and may be used by a Controller to make and apply changes. If another session identifier has been issued and the session has not ended, the Responder shall NACK with a NACK Reason Code of NR_INTERFACE_INVALID_SESSION.

A session is ended when SESSION_WINDOW elapses, or a SET_COMMAND is received with a Session Operation of DISCARD, COMMIT_RUNNING, or COMMIT_RUNNING_PERSISTENT.

If a session ends and there are any requests for changes using the session identifier which have not yet been committed, the Responder shall discard these changes.

A Responder shall not re-issue the most recent session identifier upon the next GET:IFC_SESSION_CONTROL after a session has ended, but may otherwise reuse identifiers as desired.

Time:
For GET_COMMAND_RESPONSE, the maximum time in seconds after the receipt of a SET_COMMAND with a Session Operation of COMMIT_RUNNING or COMMIT_RUNNING_PERSISTENT that the Responder will take to apply changes and be ready to receive more messages.

Controllers shall set this field to 0x0000, and Responders shall ignore for a Session Operation of DISCARD or CONFIRM.

For SET_COMMAND, the time in seconds after which the Responder should roll back to its previous configuration if it does not first receive a SET_COMMAND with a Session Operation of CONFIRM. This should be greater than the time received in a GET_COMMAND_RESPONSE.

Session Operations:
Session Operations is a bit field indicating whether to commit or discard any pending changes or confirm previously applied changes.

The Session Operations bit field is enumerated in Table A-3.

DISCARD:
The Responder shall immediately discard any pending changes received in other messages using this Session ID which were not previously confirmed.

The session is ended, and the Responder may issue a new Session ID to subsequent GET:IFC_SESSION_CONTROL requests.

COMMIT_RUNNING:
The Responder shall commit to its running configuration any pending changes received in other messages using this Session ID.

The Responder shall roll back to the previous configuration if it does not subsequently receive a SET_COMMAND with a Session Operation of CONFIRM before the number of seconds provided by Time has elapsed. Rolling back ends the session, and the Responder may issue a new Session ID to subsequent GET:IFC_SESSION_CONTROL messages. Responders may choose to verify the Session ID provided in this message, even though the session has ended.

COMMIT_RUNNING_PERSISTENT:
The Responder shall commit to its running and persistent configuration any pending changes received in other messages using this Session ID.

The Responder shall roll back to the previous configuration if it does not subsequently receive a SET_COMMAND with a Session Operation of CONFIRM before the number of seconds provided by Time has elapsed. Rolling back ends the session, and the Responder may issue a new Session ID to subsequent GET:IFC_SESSION_CONTROL messages. Responders may choose to verify the Session ID provided in this message, even though the session has ended.

CONFIRM:
The Responder shall immediately end any pending operation to roll back to previous configuration.

The session is ended, and the Responder may issue a new Session ID to subsequent GET:IFC_SESSION_CONTROL messages.

Responders shall ignore this message unless a previous session has committed changes which have not yet been confirmed.

6 Network Interface Configuration Messages

The parameters in this section provide methods for retrieving and setting the configuration of network interfaces on a Responder.

6.1 Get Interface List (IFC_INTERFACE_ID_LIST)

This parameter is used to retrieve a packed list of all interfaces which exist on the Responder, regardless of their hierarchy.

Controller: (GET)

(Port ID) 0x01 – 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001-0xFFF0
(CC) GET_COMMAND	(PID) IFC_INTERFACE_ID_LIST	(PDL) 0x00
(PD) Not Present		

Response:

(Response Type) ACK / ACK_OVERFLOW	(Message Count) 0x00 – 0xFF	(Sub-Device) Copy of Controller SD
(CC) GET_COMMAND_ RESPONSE	(PID) IFC_INTERFACE_ID_LIST	(PDL) Variable (0x00-0xE4) Multiple of 0x04
(PD) Packed List of Interface IDs Interface ID (32-bit)		

Data Description:

A packed list of unique 32-bit identifiers for all interfaces (whether physical or virtual) which exist on the Responder. Interface identifiers shall be presented in numerical order ignoring any possible hierarchy. Interface identifier 0 (0x0000000) is reserved.

For example, a Responder with two physical interfaces (1 and 2), each of which has a VLAN assignment (10 and 100 respectively) may return:

0x00 00 00 01 00 00 00 02 00 00 00 0A 00 00 00 64 (1, 2, 10, 100)

The interface identifiers returned are known as Interface IDs and shall be used by other parameters to uniquely identify an interface until an IFC_INTERFACE_TOPOLOGY queued message is received.

A Responder shall queue an IFC_INTERFACE_TOPOLOGY message at initial boot, and whenever the physical or virtual interface previously addressable by an Interface ID is no longer addressable by that Interface ID, or the list of returned Interface IDs changes.

Interface IDs are compatible with those defined in [Interface] Section 4. It is up to the implementor whether Interface IDs map directly to the identifiers defined in [Interface] Section 4.

6.2 Get Response Interface Topology (IFC_INTERFACE_TOPOLOGY)

This parameter is used to alert the controller of changes to the interface topology of a Responder.

Response:

(Response Type) ACK	(Message Count) 0x00 – 0xFF	(Sub-Device) Copy of Controller SD
(CC) GET_COMMAND_ RESPONSE	(PID) IFC_INTERFACE_TOPOLOGY	(PDL) 0x01
(PD) Topology Flags (8-bit)		

This parameter is intended for use with queued messages. Controllers shall not issue a GET_COMMAND for this parameter. Responders receiving a GET_COMMAND for this parameter shall NACK with a NACK Reason Code of NR_GET_QUEUED_ONLY.

Data Description:

Topology Flags:
Topology Flags is a bit field containing information on changes to the interface topology of a Responder.

The Topology Flags bit field is enumerated in Table A-4.

Topology Changed (Bit 0):
A topology change is defined as any time the physical or virtual interface previously addressable by an Interface ID is no longer addressable by that Interface ID, or if the list of Interface IDs that would be returned from GET:IFC_INTERFACE_ID_LIST changes.

Responders receiving a GET_COMMAND shall respond with a GET_COMMAND_RESPONSE with the value of Topology Changed (Bit 0) set to 0.

6.3 Get Interface Parent (IFC_INTERFACE_PARENT)

This parameter is used to retrieve the identifier of the parent interface for the requested interface. This may be used to establish the hierarchy of interfaces present on the Responder.

If the parent interface is on the root of the Responder's interface tree, the response shall use 0 (0x00000000) in the parent field, indicating the interface resides at the root.

Responders shall only return parent identifiers for interfaces which are included in the response to IFC_INTERFACE_ID_LIST.

Controller: (GET)

(Port ID) 0x01 – 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001-0xFFF0
(CC) GET_COMMAND	(PID) IFC_INTERFACE_PARENT	(PDL) 0x04
(PD) Interface ID (32-bit)		

Response:

(Response Type) ACK	(Message Count) 0x00 – 0xFF	(Sub-Device) Copy of Controller SD
(CC) GET_COMMAND_ RESPONSE	(PID) IFC_INTERFACE_PARENT	(PDL) 0x08
(PD) Interface ID (32-bit) Parent ID (32-bit)		

Data Description:

Interface ID:
The identifier of the interface.

Parent ID:
The identifier of the parent interface of the interface, or 0 (0x00000000), indicating the interface is on the root.

6.4 Get/Set Interface Enable (IFC_INTERFACE_ENABLE)

This parameter is used to retrieve or set the enabled state of an interface.

An option is provided to “bounce” the interface, i.e., disable it then automatically re-enable it using a specified time.

Note that disabling the interface on which the Controller is communicating with the Responder will cause a loss of all communication.

Controller: (GET)

(Port Id) 0x01 – 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001-0xFFF0
(CC) GET_COMMAND	(PID) IFC_INTERFACE_ENABLE	(PDL) 0x04
(PD) Interface ID (32-bit)		

Response:

(Response Type) ACK	(Message Count) 0x00 – 0xFF	(Sub-Device) Copy of Controller SD
(CC) GET_COMMAND_ RESPONSE	(PID) IFC_INTERFACE_ENABLE	(PDL) 0x05
(PD) Interface ID (32-bit) Interface State (8-bit)		

Controller: (SET)

(Port Id) 0x01 – 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001-0xFFF0	
(CC) SET_COMMAND	(PID) IFC_INTERFACE_ENABLE		(PDL) 0x06
(PD) Interface ID (32-bit) Interface State (8-bit) | Time (8-bit)			

Response:

(Response Type) ACK	(Message Count) 0x00 – 0xFF	(Sub-Device) Copy of Controller SD	
(CC) SET_COMMAND_ RESPONSE	(PID) IFC_INTERFACE_ENABLE		(PDL) 0x04
(PD) Interface ID (32-bit)			

Data Description:

Interface ID:
The identifier of the interface.

Interface State:
For GET_COMMAND_RESPONSE, returns whether the interface is enabled, disabled or not present. BOUNCE shall not be returned, unless a previous SET_COMMAND using BOUNCE is still in progress.

Some Responders may choose to include Interface IDs for interfaces which are not present in the list of Interface IDs returned from GET:IFC_INTERFACE_ID_LIST. For example: ports which receive SFP (small form-factor pluggable) modules on a network switch. In such cases, a Responder shall return NOT_PRESENT in GET_COMMAND_RESPONSE.

For SET_COMMAND, whether the interface should be enabled, disabled, or whether it should bounce. Bounce means the interface shall be disabled, then automatically re-enabled after the specified time. During this period, the device shall NACK any additional SET:IFC_INTERFACE_ENABLE commands with a NACK Reason Code of NR_BUSY.

Interface States are enumerated in Table A-5.

Time:
The time in seconds for which the interface shall be disabled prior to re-enabling when Interface State is set to BOUNCE. Responders shall ignore Time if Interface State is not set to BOUNCE.

6.5 Get Interface Hardware Address (IFC_INTERFACE_HARDWARE_ADDRESS)

This parameter is used to retrieve the hardware address of the interface. This interface may either be a physical or virtual interface.

Controller: (GET)

(Port Id) 0x01 – 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001-0xFFF0
(CC) GET_COMMAND	(PID) IFC_INTERFACE_HARDWARE_ADDRESS	(PDL) 0x04
(PD) Interface ID (32-bit)		

Response:

(Response Type) ACK	(Message Count) 0x00 – 0xFF	(Sub-Device) Copy of Controller SD
(CC) GET_COMMAND_ RESPONSE	(PID) IFC_INTERFACE_HARDWARE_ADDRESS	(PDL) Variable (0x0B-0x0D)
(PD) Interface ID (32-bit)		
Hardware Address Type (8-bit)		
Hardware Address (Variable)		

Data Description:

Interface ID:
The identifier of the interface.

Hardware Address Type:
The expected type of the value of Hardware Address.

Hardware Address Types are enumerated in Table A-6.

EUI-48 and EUI-64 are defined in [802], [802c] and [802f] Clause 8.

Hardware Address:
The hardware address of the interface, as a raw byte sequence.

For example, an EUI-48 address represented in text as “6C-81-CE-13-BF-3E” or “6C:81:CE:13:BF:3E” would be:

0x6c81ce13bf3e

The length of this field is determined by the value of Hardware Address Type. See Table A-6.

6.6 Get Interface Fixed Label (IFC_INTERFACE_FIXED_LABEL)

This parameter is used to retrieve a manufacturer-defined descriptive label for an interface. For example, this may be used to indicate the label of a physical port on a device such as “Port 1”.

This parameter is read-only. For information on a method to provide a read/write user-definable descriptive label, see IFC_INTERFACE_USER_LABEL in Section 6.7.

Controller: (GET)

(Port ID) 0x01 – 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001-0xFFF0
(CC) GET_COMMAND	(PID) IFC_INTERFACE_FIXED_LABEL	(PDL) 0x04
(PD) Interface ID (32-bit)		

Response:

(Response Type) ACK	(Message Count) 0x00 – 0xFF	(Sub-Device) Copy of Controller SD
(CC) GET_COMMAND_ RESPONSE	(PID) IFC_INTERFACE_ FIXED_LABEL	(PDL) Variable (0x04 – 0xE7)
(PD) Interface ID (32-bit) Fixed Label (Variable Length)		

Data Description:

Interface ID:
The identifier of the interface.

Fixed Label:
The Fixed Label is a text field as defined in Section 3.3.

6.7 Get/Set Interface User Assignable Label (IFC_INTERFACE_USER_LABEL)

This parameter is used to retrieve or set a user-definable descriptive label for an interface. For example: “Local Area Connection 2” or “Lighting Network”.

This parameter is intended for user-definable labels. For information on a method to provide a read-only manufacturer-defined name for an interface, see IFC_INTERFACE_FIXED_LABEL in Section 6.6.

Controller: (GET)

(Port ID) 0x01 – 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001-0xFFF0
(CC) GET_COMMAND	(PID) IFC_INTERFACE_USER_LABEL	(PDL) 0x04
(PD) Interface ID (32-bit)		

Response:

(Response Type) ACK	(Message Count) 0x00 – 0xFF	(Sub-Device) Copy of Controller SD
(CC) GET_COMMAND_ RESPONSE	(PID) IFC_INTERFACE_USER_LABEL	(PDL) Variable (0x04 – 0xE7)
(PD) Interface ID (32-bit) User Label (Variable)		

Controller: (SET)

(Port ID) 0x01 - 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001-0xFFF0
(CC) SET_COMMAND	(PID) IFC_INTERFACE_USER_LABEL	(PDL) Variable (0x04 – 0xE7)
(PD) Interface ID (32-bit) User Label (Variable)		

Response:

(Response Type) ACK	(Message Count) 0x00 - 0xFF	(Sub-Device) Copy of Controller SD
(CC) SET_COMMAND_ RESPONSE	(PID) IFC_INTERFACE_USER_LABEL	(PDL) 0x04
(PD) Interface ID (32-bit) 		

Data Description:

Interface ID:
The identifier of the interface.

User Label:
The User Label is a text field as defined in Section 3.3.

6.8 Get Interface Type (IFC_INTERFACE_TYPE)

This parameter is used to retrieve the type of an interface, such as physical fiber or RJ45, or virtual.

Controller: (GET)

(Port ID) 0x01 – 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001-0xFFF0
(CC) GET_COMMAND	(PID) IFC_INTERFACE_TYPE	(PDL) 0x04
(PD) Interface ID (32-bit)		

Response:

(Response Type) ACK	(Message Count) 0x00 – 0xFF	(Sub-Device) Copy of Controller SD
(CC) GET_COMMAND_ RESPONSE	(PID) IFC_INTERFACE_TYPE	(PDL) 0x05
(PD) Interface ID (32-bit) Interface Type (8-bit)		

Data Description:

Interface ID:
The identifier of the interface.

Interface Type:
Identifies the broad category of the interface, such as if it is wired, wireless, optical, or even virtual.

Interface Types are enumerated in Table A-7.

6.9 Get Interface Line Speed (IFC_INTERFACE_LINE_SPEED)

This parameter is used to retrieve the line speed of an interface, which may also be known as bandwidth, or data transfer rate. The line speed is provided in bits per second.

If the interface is disabled or not present, the Responder shall NACK with a NACK Reason Code of NR_INTERFACE_UNAVAILABLE.

Controller: (GET)

(Port ID) 0x01 – 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001-0xFFF0
(CC) GET_COMMAND	(PID) IFC_INTERFACE_LINE_SPEED	(PDL) 0x04
(PD) Interface ID (32-bit)		

Response:

(Response Type) ACK	(Message Count) 0x00 – 0xFF	(Sub-Device) Copy of Controller SD
(CC) GET_COMMAND_ RESPONSE	(PID) IFC_INTERFACE_LINE_SPEED	(PDL) 0x0C
(PD) Interface ID (32-bit) Line Speed (64-bit)		

Data Description:

Interface ID:
The identifier of the interface.

Line Speed:
The number of bits per second that the interface provides.

Note: User interfaces are encouraged to present this in familiar understandable terms such as: 100 Mb, 1 Gb, 2.5 Gb, 10 Gb, etc.

6.10 Get Interface Utilization (IFC_INTERFACE_UTILIZATION)

This parameter is used to retrieve the utilization of the line speed of an interface (as may be retrieved using IFC_INTERFACE_LINE_SPEED). The sample period for determining the utilization is not specified by this standard and may be chosen by the manufacturer.

If the interface is disabled or not present, the Responder shall NACK with a NACK Reason Code of NR_INTERFACE_UNAVAILABLE.

Controller: (GET)

(Port ID) 0x01 – 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001-0xFFF0
(CC) GET_COMMAND	(PID) IFC_INTERFACE_UTILIZATION	(PDL) 0x04
(PD) Interface ID (32-bit)		

Response:

(Response Type) ACK	(Message Count) 0x00 – 0xFF	(Sub-Device) Copy of Controller SD
(CC) GET_COMMAND_ RESPONSE	(PID) IFC_INTERFACE_UTILIZATION	(PDL) 0x0C
(PD) Interface ID (32-bit) Utilization (64-bit) 		

Data Description:

Interface ID:
The identifier of the interface.

Utilization:
The utilization of the interface in bits per second.

Note: User interfaces may wish to use this in conjunction with the line speed to present it as a utilization percentage.

6.11 Get Interface Duplex Mode (IFC_INTERFACE_DUPLEX_MODE)

This parameter is used to retrieve the duplex mode of an interface, i.e., whether it is possible to transmit and receive at the same time.

If the interface is disabled or not present, the Responder shall NACK with a NACK Reason Code of NR_INTERFACE_UNAVAILABLE.

Controller: (GET)

(Port ID) 0x01 – 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001-0xFFF0
(CC) GET_COMMAND	(PID) IFC_INTERFACE_DUPLEX_MODE	(PDL) 0x04
(PD) Interface ID (32-bit)		

Response:

(Response Type) ACK	(Message Count) 0x00 – 0xFF	(Sub-Device) Copy of Controller SD
(CC) GET_COMMAND_ RESPONSE	(PID) IFC_INTERFACE_DUPLEX_MODE	(PDL) 0x05
(PD) Interface ID (32-bit) Duplex Mode (8-bit)		

Data Description:

Interface ID:
The identifier of the interface.

Duplex Mode:
Whether the interface can send and receive at the same time.

Duplex Modes are enumerated in Table A-**8**.

6.12 Get/Set Energy Efficient Ethernet Enable (IFC_INTERFACE_EEE)

This parameter is used to retrieve or set whether Energy Efficient Ethernet is enabled on an interface.

If the interface is disabled or not present, the Responder shall NACK with a NACK Reason Code of NR_INTERFACE_UNAVAILABLE.

Note: The process by which Energy Efficient Ethernet operates can result in an interface temporarily ceasing to receive or transmit data, which adds latency to protocols such as [sACN]

which rely on a consistent stream of data. Care should be taken when supporting this parameter and informing users of the trade-off between latency and power saving.

Controller: (GET)

(Port ID) 0x01 – 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001-0xFFF0
(CC) GET_COMMAND	(PID) IFC_INTERFACE_EEE	(PDL) 0x04
(PD) Interface ID (32-bit)		

Response:

(Response Type) ACK	(Message Count) 0x00 – 0xFF	(Sub-Device) Copy of Controller SD
(CC) GET_COMMAND_ RESPONSE	(PID) IFC_INTERFACE_EEE	(PDL) 0x05
(PD) Interface ID (32-bit) EEE State (8-bit)		

Controller: (SET)

(Port ID) 0x01 – 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001-0xFFF0
(CC) SET_COMMAND	(PID) IFC_INTERFACE_EEE	(PDL) 0x05
(PD) Interface ID (32-bit) EEE State (8-bit)		

Response:

(Response Type) ACK	(Message Count) 0x00 – 0xFF	(Sub-Device) Copy of Controller SD
(CC) SET_COMMAND_ RESPONSE	(PID) IFC_INTERFACE_EEE	(PDL) 0x04
(PD) Interface ID (32-bit)		

Data Description:

Interface ID:
The identifier of the interface.

EEE State:
For GET_COMMAND_RESPONSE, whether Energy Efficient Ethernet is enabled or disabled.

For SET_COMMAND, whether the Energy Efficient Ethernet should be enabled, or disabled.

EEE States are enumerated in Table A-9.

6.13 Get/Set Interface Power Enable (IFC_INTERFACE_POWER_ENABLE)

This parameter is used to retrieve or set whether a network interface that is capable of providing power to connected equipment is enabled on an interface.

Examples of such technologies include, but are not limited to, Clause 145 “Power over Ethernet” or Clause 104 “Power over Data Lines (PoDL) of Single-Pair Ethernet” from IEEE 802.3 [802.3].

Controller: (GET)

(Port Id) 0x01 – 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001-0xFFF0
(CC) GET_COMMAND	(PID) IFC_INTERFACE_POWER_ENABLE	(PDL) 0x04
(PD) Interface ID (32-bit)		

Response:

(Response Type) ACK	(Message Count) 0x00 – 0xFF	(Sub-Device) Copy of Controller SD
(CC) GET_COMMAND_ RESPONSE	(PID) IFC_INTERFACE_POWER_ENABLE	(PDL) 0x05
(PD) Interface ID (32-bit) Power State (8-bit)		

Controller: (SET)

(Port Id) 0x01 – 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001-0xFFF0
(CC) SET_COMMAND	(PID) IFC_INTERFACE_POWER_ENABLE	(PDL) 0x05
(PD) Interface ID (32-bit) Power State (8-bit)		

Response:

(Response Type) ACK	(Message Count) 0x00 – 0xFF	(Sub-Device) Copy of Controller SD
(CC) SET_COMMAND_ RESPONSE	(PID) IFC_INTERFACE_POWER_ENABLE	(PDL) 0x04
(PD) Interface ID (32-bit)		

Data Description:

Interface ID:
The identifier of the interface.

Power State:
For GET_COMMAND_RESPONSE, returns whether interface power is enabled or disabled.

For SET_COMMAND, whether interface power should be enabled or disabled.

Power States are enumerated in Table A-10

Table A-10.

6.14 Get Interface Power Status (IFC_INTERFACE_POWER_STATUS)

This parameter is used to retrieve the capabilities and current state of power delivery performed by the addressed network interface. This applies only to network interfaces that are capable of providing power to connected equipment or consuming power from connected equipment.

Examples of such technologies include, but are not limited to, Clause 145 “Power over Ethernet” or Clause 104 “Power over Data Lines (PoDL) of Single-Pair Ethernet” from IEEE 802.3 [802.3].

Controller: (GET)

(Port Id) 0x01 – 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001-0xFFF0
(CC) GET_COMMAND	(PID) IFC_INTERFACE_POWER_STATUS	(PDL) 0x04
(PD) Interface ID (32-bit)		

Response:

(Response Type) ACK	(Message Count) 0x00 – 0xFF	(Sub-Device) Copy of Controller SD
(CC) GET_COMMAND_ RESPONSE	(PID) IFC_INTERFACE_POWER_STATUS	(PDL) 0x0E

(PD)	
Interface ID (32-bit)	
IFC Power Type (8-bit)	IFC Power Role (8-bit)
Maximum Power Capability (16-bit)	
Power Requested (16-bit)	
Power Granted (16-bit)	
Power Consumed (16-bit)	

Data Description:

Interface ID:

The identifier of the interface.

IFC Power Type:
Defines the Interface Power Delivery technology in use. Defined power deliveries technologies are enumerated in Table A-11.

IFC Power Role:
Defines the roles played by the interface in a powered system, such as power sourcing equipment or powered device. Defined power roles are enumerated in Table A-12.

Maximum Power Capability:
For power sourcing equipment, this field reports the maximum amount of power that the interface is capable of sourcing to connected device(s). This value is expressed in units of 0.1 Watts.

For powered devices, this field reports the maximum amount of power that the interface is capable of drawing from power sourcing equipment in any mode of operation. This value is expressed in units of 0.1 Watts.

Power Requested:
For power sourcing equipment, this field reports the amount of power that the interface is aware of being requested by connected device(s). This value is expressed in units of 0.1 Watts.

For powered devices, this field reports the amount of power currently requested from the power sourcing equipment for the current mode of operation. This value is expressed in units of 0.1 Watts.

Power Granted:
For power sourcing equipment, this field reports the amount of power that the interface has granted to connected device(s). This value is expressed in units of 0.1 Watts.

For powered devices, this field reports the amount of power currently granted to the device by the power sourcing equipment. This value is expressed in units of 0.1 Watts.

Power Consumed:
For power sourcing equipment, this field reports the amount of power being consumed at the time of the request by the connected device(s). This value is expressed in units of 0.1 Watts.

For powered devices, this field reports the amount of power currently consumed by the device from the power sourcing equipment. This value is expressed in units of 0.1 Watts.

The reserved value of 0xFFFF shall be reported in the Power Requested, Power Granted, or Power Consumed if the interface does not have the required information available to report.

Note: This 0.1 W unit scheme was chosen to be consistent with IEEE 802.3 [802.3] 79.3.2 Power Via MDI TLV definitions which is the most common power reporting format for Power over Ethernet systems.

7 IPv4 Configuration Messages

The parameters in this section provide methods for retrieving and setting the IPv4 capabilities of a Responder.

7.1 Get/Set IPv4 Address (IFC_IPV4_ADDRESS)

This parameter is used to retrieve or set the IPv4 addresses present on an interface.

If the interface is disabled or not present, the Responder shall NACK with a NACK Reason Code of NR_INTERFACE_UNAVAILABLE.

Controller: (GET)

(Port Id) 0x01 – 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001-0xFFF0
(CC) GET_COMMAND	(PID) IFC_IPV4_ADDRESS	(PDL) 0x04
(PD) Interface ID (32-bit)		

Response:

(Response Type) ACK	(Message Count) 0x00 – 0xFF	(Sub-Device) Copy of Controller SD
(CC) GET_COMMAND_ RESPONSE	(PID) IFC_IPV4_ADDRESS	(PDL) Variable (0x04-0xE2) 0x04 + (Multiple of 0x06)
(PD) Interface ID (32-bit) Packed List of IPv4 Addresses IPv4 Address (32-bit) Netmask Size (8-bit) | Source (8-bit)		

Controller: (SET)

(Port Id) 0x01 – 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001-0xFFF0
(CC) SET_COMMAND	(PID) IFC_IPV4_ADDRESS	(PDL) Variable (0x04-0xE5) 0x04 + (Multiple of 0x05)
(PD) Interface ID (32-bit) Packed List of IPv4 Addresses IPv4 Address (32-bit) Netmask Size (8-bit)		

Response:

(Response Type) ACK	(Message Count) 0x00 – 0xFF	(Sub-Device) Copy of Controller SD
(CC) SET_COMMAND_ RESPONSE	(PID) IFC_IPV4_ADDRESS	(PDL) 0x04
(PD) Interface ID (32-bit)		

Data Description:

For GET_COMMAND_RESPONSE, a packed list of IPv4 addresses (including netmask and source) present on the interface.

For SET_COMMAND, a packed list of IPv4 addresses (including netmask) which should be present on the interface.

If more IPv4 addresses are provided than can be accepted, the Responder shall NACK with a NACK Reason Code of NR_IPV4_LIMIT_EXCEEDED.

Interface ID:
The identifier of the interface.

IPv4 Address:
For GET_COMMAND_RESPONSE, returns an IPv4 address assigned to the interface.

For SET_COMMAND, an IPv4 address to assign to the interface.

IPv4 addresses are defined in [IPv4] Section 3.2.

For example, an IPv4 address represented as “10.5.8.11” in dot-decimal notation would be:

0x0A05080B

Netmask Size:
For GET_COMMAND_RESPONSE, returns the netmask size in bits used with this IPv4 address.

For SET_COMMAND, the netmask size in bits to be used with this IPv4 address.

For example, a netmask of “255.255.252.0” (22 bits) would be:

0x16

Source:
The source of the IPv4 address, for example DHCP or Static (manual) assignment.

Sources are enumerated in Table A-13.

7.2 Get/Set IPv4 Gateway (IFC_IPV4_GATEWAY)

This parameter is used to retrieve or set the IPv4 gateway assigned to an interface.

If the interface is disabled or not present, the Responder shall NACK with a NACK Reason Code of NR_INTERFACE_UNAVAILABLE.

Controller: (GET)

(Port Id) 0x01 – 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001-0xFFF0
(CC) GET_COMMAND	(PID) IFC_IPV4_GATEWAY	(PDL) 0x04
(PD) Interface ID (32-bit)		

Response:

(Response Type) ACK	(Message Count) 0x00 – 0xFF	(Sub-Device) Copy of Controller SD
(CC) GET_COMMAND_ RESPONSE	(PID) IFC_IPV4_GATEWAY	(PDL) 0x08
(PD) Interface ID (32-bit) IPv4 Gateway (32-bit)		

Controller: (SET)

(Port Id) 0x01 – 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001-0xFFF0
(CC) SET_COMMAND	(PID) IFC_IPV4_GATEWAY	(PDL) 0x08
(PD) Interface ID (32-bit) IPv4 Gateway (32-bit)		

Response:

(Response Type) ACK	(Message Count) 0x00 – 0xFF	(Sub-Device) Copy of Controller SD
(CC) SET_COMMAND_ RESPONSE	(PID) IFC_IPV4_GATEWAY	(PDL) 0x04
(PD) Interface ID (32-bit)		

Data Description:

Interface ID:
The identifier of the interface.

IPv4 Gateway:
For GET_COMMAND_RESPONSE, returns the IPv4 gateway assigned to the interface.

For SET_COMMAND, the new IPv4 gateway to assign to the interface.

IPv4 gateways are IPv4 addresses as defined in [IPv4] Section 3.2.

For example, an IPv4 gateway represented as “10.5.8.1” in dot-decimal notation would be:

0x0A050801

7.3 Set Remove IPv4 Gateway (IFC_IPV4_GATEWAY_REMOVE)

This parameter is used to remove the IPv4 gateway from an interface.

If the interface is disabled or not present, the Responder shall NACK with a NACK Reason Code of NR_INTERFACE_UNAVAILABLE.

Controller: (SET)

(Port Id) 0x01 – 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001-0xFFF0
(CC) SET_COMMAND	(PID) IFC_IPV4_GATEWAY_REMOVE	(PDL) 0x04
(PD) Interface ID (32-bit)		

Response:

(Response Type) ACK	(Message Count) 0x00 – 0xFF	(Sub-Device) Copy of Controller SD
(CC) SET_COMMAND_ RESPONSE	(PID) IFC_IPV4_GATEWAY_REMOVE	(PDL) 0x04
(PD) Interface ID (32-bit)		

Data Description:

Interface ID:
The identifier of the interface.

7.4 Get/Set IPv4 DNS Address (IFC_IPV4_DNS)

This parameter is used to retrieve or set the IPv4 DNS server addresses present on an interface.

If the interface is disabled or not present, the Responder shall NACK with a NACK Reason Code of NR_INTERFACE_UNAVAILABLE.

Controller: (GET)

(Port Id) 0x01 – 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001-0xFFF0
(CC) GET_COMMAND	(PID) IFC_IPV4_DNS	(PDL) 0x04
(PD) Interface ID (32-bit)		

Response:

(Response Type) ACK	(Message Count) 0x00 – 0xFF	(Sub-Device) Copy of Controller SD
(CC) GET_COMMAND_ RESPONSE	(PID) IFC_IPV4_DNS	(PDL) Variable (0x04-0xE4) Multiple of 0x04
(PD) Interface ID (32-bit) Packed List of IPv4 DNS Addresses IPv4 DNS Address (32-bit)		

Controller: (SET)

(Port Id) 0x01 – 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001-0xFFF0
(CC) SET_COMMAND	(PID) IFC_IPV4_DNS	(PDL) Variable (0x04-0xE4) Multiple of 0x04
(PD) Interface ID (32-bit) Packed List of IPv4 DNS Addresses IPv4 DNS Address (32-bit)		

Response:

(Response Type) ACK	(Message Count) 0x00 – 0xFF	(Sub-Device) Copy of Controller SD
(CC) SET_COMMAND_ RESPONSE	(PID) IFC_IPV4_DNS	(PDL) 0x04
(PD) Interface ID (32-bit)		

Data Description:

For GET_COMMAND_RESPONSE, returns a packed list of IPv4 DNS server addresses present on the interface.

For SET_COMMAND, a packed list of IPv4 DNS server addresses which should be present on the interface.

If more IPv4 DNS server addresses are provided than can be accepted, the Responder shall NACK with a NACK Reason Code of NR_IPV4_LIMIT_EXCEEDED.

Interface ID:
The identifier of the interface.

IPv4 DNS Address:
For GET_COMMAND_RESPONSE, returns an IPv4 DNS server address assigned to the interface.

For SET_COMMAND, an IPv4 DNS server address to assign to the interface.

IPv4 DNS server addresses are IPv4 addresses as defined in [IPv4] Section 3.2.

For example, an IPv4 DNS server address represented as “10.5.4.21” in dot-decimal notation would be:

0x0A050415

7.5 Get IPv4 Supported Modes (IFC_IPV4_SUPPORTED_MODES)

This parameter is used to retrieve the supported IPv4 address assignment modes of an interface.

If the interface is disabled or not present, the Responder shall NACK with a NACK Reason Code of NR_INTERFACE_UNAVAILABLE.

Controller: (GET)

(Port Id) 0x01 – 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001-0xFFF0
(CC) GET_COMMAND	(PID) IFC_IPV4_SUPPORTED_MODES	(PDL) 0x04
(PD) Interface ID (32-bit)		

Response:

(Response Type) ACK	(Message Count) 0x00 – 0xFF	(Sub-Device) Copy of Controller SD
(CC) GET_COMMAND_ RESPONSE	(PID) IFC_IPV4_SUPPORTED_MODES	(PDL) Variable (0x04-0x08) 0x04 + (Multiple of 0x01)
(PD) Interface ID (32-bit) Packed List of Supported Modes Supported Mode (8-bit)		

Data Description:

A packed list of supported IPv4 address assignment modes for the interface.

Interface ID:
The identifier of the interface.

Supported Mode:
A supported IPv4 address assignment mode for this interface.

Supported Modes are enumerated in Table A-14.

7.6 Get/Set IPv4 Mode (IFC_IPV4_MODE)

This parameter is used to retrieve or set the IPv4 address assignment mode of an interface.

If the interface is disabled or not present, the Responder shall NACK with a NACK Reason Code of NR_INTERFACE_UNAVAILABLE.

Controller: (GET)

(Port Id) 0x01 – 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001-0xFFF0
(CC) GET_COMMAND	(PID) IFC_IPV4_MODE	(PDL) 0x04
(PD) Interface ID (32-bit)		

Response:

(Response Type) ACK	(Message Count) 0x00 – 0xFF	(Sub-Device) Copy of Controller SD
(CC) GET_COMMAND_ RESPONSE	(PID) IFC_IPV4_MODE	(PDL) 0x05
(PD) Interface ID (32-bit) Mode (8-bit)		

Controller: (SET)

(Port Id) 0x01 – 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001-0xFFF0
(CC) SET_COMMAND	(PID) IFC_IPV4_MODE	(PDL) 0x05
(PD) Interface ID (32-bit) Mode (8-bit)		

Response:

(Response Type) ACK	(Message Count) 0x00 – 0xFF	(Sub-Device) Copy of Controller SD
(CC) SET_COMMAND_ RESPONSE	(PID) IFC_IPV4_MODE	(PDL) 0x04
(PD) Interface ID (32-bit)		

Data Description:

Interface ID:
The identifier of the interface.

Mode:
For GET_COMMAND_RESPONSE, returns the IPv4 address assignment mode for the interface.

For SET_COMMAND, the new IPv4 address assignment mode for the interface. If the mode is not supported, the Responder shall NACK with a NACK Reason Code of NR_MODE_UNSUPPORTED.

Each assignment mode consists of a primary method and a fallback method which may be none.

Modes are enumerated in Table A-15.

7.7 Get DHCPv4 Server Result (IFC_IPV4_DHCP_RESULT)

This parameter is used to retrieve the DHCPv4 server address that assigned the IPv4 address to the interface.

If the interface is disabled or not present, the Responder shall NACK with a NACK Reason Code of NR_INTERFACE_UNAVAILABLE.

Controller: (GET)

(Port Id) 0x01 – 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001-0xFFF0
(CC) GET_COMMAND	(PID) IFC_IPV4_DHCP_RESULT	(PDL) 0x04
(PD) Interface ID (32-bit)		

Response:

(Response Type) ACK	(Message Count) 0x00 – 0xFF	(Sub-Device) Copy of Controller SD
(CC) GET_COMMAND_ RESPONSE	(PID) IFC_IPV4_DHCP_RESULT	(PDL) 0x0D
(PD) Interface ID (32-bit) DHCPv4 Address (32-bit) Lease Time (32-bit) Flags (8-bit)		

Data Description:

Interface ID:
The identifier of the interface.

DHCPv4 Address:
The DHCPv4 server address that assigned the IPv4 address to this interface.

DHCPv4 server addresses are IPv4 addresses as defined in [IPv4] Section 3.2.

For example, a DHCPv4 server address represented as “10.5.4.21” in dot-decimal notation would be:

0x0A050415

Lease Time:
The time remaining on the lease of the IPv4 address provided by the DHCPv4 server.

Lease times are in seconds, as defined in [DHCP] Section 3.3. A value of 0xFFFFFFFF represents an infinite lease.

Responders which do not know the remaining time on the lease shall return a value of 0x00000000 and set Valid Lease Time to 0 in Flags. Controllers shall ignore the lease time if the value of Valid Lease Time in Flags is 0.

Flags:
Flags is a bit field indicating the validity and handling of other fields included in this parameter.

The Flags bit field is enumerated in Table A-16.

8 IPv6 Configuration Messages

The parameters in this section provide methods for retrieving and setting the IPv6 capabilities of a Responder.

8.1 Get/Set IPv6 Address (IFC_IPV6_ADDRESS)

This parameter is used to retrieve or set the IPv6 addresses present on an interface.

Note: This parameter does not retrieve the IPv6 link-local address on an interface. See Section 8.2 for further information.

If the interface is disabled or not present, the Responder shall NACK with a NACK Reason Code of NR_INTERFACE_UNAVAILABLE.

Controller: (GET)

<table>
<tr><td>(Port Id)
0x01 – 0xFF</td><td>(Message Count)
0x00</td><td colspan="2">(Sub-Device)
0x0000 (Root) or 0x0001-0xFFF0</td></tr>
<tr><td>(CC)
GET_COMMAND</td><td colspan="2">(PID)
IFC_IPV6_ADDRESS</td><td>(PDL)
0x04</td></tr>
<tr><td colspan="4">(PD)
Interface ID (32-bit)</td></tr>
</table>

Response:

(Response Type) ACK	(Message Count) 0x00 – 0xFF	(Sub-Device) Copy of Controller SD
(CC) GET_COMMAND_ RESPONSE	(PID) IFC_IPV6_ADDRESS	(PDL) Variable (0x04-0xDC) 0x04 + (Multiple of 0x12)

(PD)

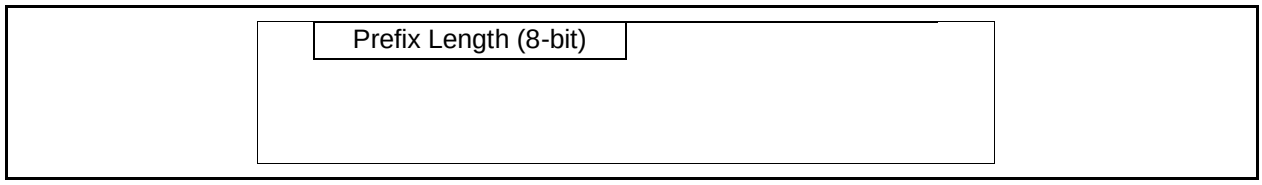
Interface ID (32-bit)	
Packed List of IPv6 Addresses	
IPv6 Address (128-bit)	
Prefix Length (8-bit)	Source (8-bit)

Controller: (SET)

(Port Id) 0x01 – 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001-0xFFF0
(CC) SET_COMMAND	(PID) IFC_IPV6_ADDRESS	(PDL) Variable (0x04-0xE3) 0x04 + (Multiple of 0x11)

(PD)

Interface ID (32-bit)
Packed List of IPv6 Addresses
IPv6 Address (128-bit)


Prefix Length (8-bit)

Response:

(Response Type) ACK	(Message Count) 0x00 – 0xFF	(Sub-Device) Copy of Controller SD
(CC) SET_COMMAND_ RESPONSE	(PID) IFC_IPV6_ADDRESS	(PDL) 0x04
(PD) Interface ID (32-bit)		

Data Description:

For GET_COMMAND_RESPONSE, returns a packed list of IPv6 addresses (including prefix length and source) present on the interface.

Responders shall not include the link-local address present on the interface in this list.

For SET_COMMAND, a packed list of IPv6 addresses (including prefix length) which should be present on the interface.

If more IPv6 addresses are provided than can be accepted, the Responder shall NACK with a NACK Reason Code of NR_IPV6_LIMIT_EXCEEDED.

Interface ID:
The identifier of the interface.

IPv6 Address:
For GET_COMMAND_RESPONSE, an IPv6 address assigned to the interface.

For SET_COMMAND, an IPv6 address to assign to the interface.

IPv6 addresses are defined in [IPv6-Addressing].

For example, an IPv6 address represented as “2001:0000:130F:0000:0000:09C0:876A:130B” or the short-form “2001:0:130F::9C0:876A:130B” would be:

0x20010000130F0000000009C0876A130B

Prefix Length:
For GET_COMMAND_RESPONSE, the prefix length in bits used with this IPv6 address.

For SET_COMMAND, the prefix length in bits to be used with this IPv6 address.

For example, a prefix length of “/64” (64 bits) would be:

0x40

Source:
The source of the IPv6 address, for example DHCPv6 or Static (manual) assignment.

Sources are enumerated in Table A-17.

8.2 Get IPv6 Link-Local Address (IFC_IPV6_ADDRESS_LINKLOCAL)

This parameter is used to retrieve the IPv6 link-local address assigned to an interface.

Note: IPv6 link-local addresses are not removable and cannot be modified.

Controller: (GET)

(Port Id) 0x01 – 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001-0xFFF0
(CC) GET_COMMAND	(PID) IFC_IPV6_ADDRESS_LINKLOCAL	(PDL) 0x04
(PD) Interface ID (32-bit)		

Response:

(Response Type) ACK	(Message Count) 0x00 – 0xFF	(Sub-Device) Copy of Controller SD
(CC) GET_COMMAND_ RESPONSE	(PID) IFC_IPV6_ADDRESS_LINKLOCAL	(PDL) 0x14
(PD) Interface ID (32-bit)		
IPv6 Link-Local Address (128-bit)		

Data Description:

Interface ID:
The identifier of the interface.

IPv6 Link-Local Address:

The IPv6 link-local address assigned to the interface.

IPv6 link-local addresses are IPv6 addresses as defined in [IPv6-Addressing] Section 2.5.6.

For example, an IPv6 address represented as “FE80:0000:0000:0000:0210:5AFF:FEAA:20A2” or the short-form “FE80::210:5AFF:FEAA:20A2” would be:

0xFE80000000000000210 5AFFFEAA20A2

Note: All IPv6 link-local addresses have a prefix length of 64 bits “fe80::/64”.

8.3 Get/Set IPv6 Gateway (IFC_IPV6_GATEWAY)

This parameter is used to retrieve or set the IPv6 gateway assigned to an interface.

If the interface is disabled or not present, the Responder shall NACK with a NACK Reason Code of NR_INTERFACE_UNAVAILABLE.

Controller: (GET)

(Port Id) 0x01 – 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001-0xFFF0
(CC) GET_COMMAND	(PID) IFC_IPV6_GATEWAY	(PDL) 0x04
(PD) Interface ID (32-bit)		

Response:

(Response Type) ACK	(Message Count) 0x00 – 0xFF	(Sub-Device) Copy of Controller SD
(CC) GET_COMMAND_ RESPONSE	(PID) IFC_IPV6_GATEWAY	(PDL) 0x14
(PD) Interface ID (32-bit) IPv6 Gateway (128-bit) 		

Controller: (SET)

(Port Id) 0x01 – 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001-0xFFF0
(CC) SET_COMMAND	(PID) IFC_IPV6_GATEWAY	(PDL) 0x14
(PD) Interface ID (32-bit) IPv6 Gateway (128-bit) 		

Response:

(Response Type) ACK	(Message Count) 0x00 – 0xFF	(Sub-Device) Copy of Controller SD
(CC) SET_COMMAND_ RESPONSE	(PID) IFC_IPV6_GATEWAY	(PDL) 0x04
(PD) Interface ID (32-bit)		

Data Description:

Interface ID:
The identifier of the interface.

IPv6 Gateway:

For GET_COMMAND_RESPONSE, returns the IPv6 gateway assigned to the interface.

For SET_COMMAND, the new IPv6 gateway to assign to the interface.

IPv6 gateways are IPv6 addresses as defined in [IPv6-Addressing].

For example, an IPv6 gateway represented as “2001:0000:130F:0000:0000:09C0:876A:130B” or the short-form “2001:0:130F::9C0:876A:130B” would be:

0x20010000130F0000000009C0876A130B

8.4 Set Remove IPv6 Gateway (IFC_IPV6_GATEWAY_REMOVE)

This parameter is used to remove the IPv6 gateway from an interface.

Controller: (SET)

(Port Id) 0x01 – 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001-0xFFF0
(CC) SET_COMMAND	(PID) IFC_IPV6_GATEWAY_REMOVE	(PDL) 0x04
(PD) Interface ID (32-bit)		

Response:

(Response Type) ACK	(Message Count) 0x00 – 0xFF	(Sub-Device) Copy of Controller SD
(CC) SET_COMMAND_ RESPONSE	(PID) IFC_IPV6_GATEWAY_REMOVE	(PDL) 0x04
(PD) Interface ID (32-bit)		

Data Description:

Interface ID:
The identifier of the interface.

8.5 Get/Set IPv6 DNS Address (IFC_IPV6_DNS)

This parameter is used to retrieve or set the IPv6 DNS server addresses present on an interface.

If the interface is disabled or not present, the Responder shall NACK with a NACK Reason Code of NR_INTERFACE_UNAVAILABLE.

Controller: (GET)

(Port Id) 0x01 – 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001-0xFFF0
(CC) GET_COMMAND	(PID) IFC_IPV6_DNS	(PDL) 0x04
(PD) Interface ID (32-bit)		

Response:

(Response Type) ACK	(Message Count) 0x00 – 0xFF	(Sub-Device) Copy of Controller SD
(CC) GET_COMMAND_ RESPONSE	(PID) IFC_IPV6_DNS	(PDL) Variable (0x04-0xDC) 0x04 + (Multiple of 0x12)

(PD)

Interface ID (32-bit)

Packed List of IPv6 DNS Addresses

IPv6 DNS Address (128-bit)

Controller: (SET)

(Port Id) 0x01 – 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001-0xFFF0
(CC) SET_COMMAND	(PID) IFC_IPV6_DNS	(PDL) Variable (0x04-0xDC) 0x04 + (Multiple of 0x12)

(PD)

Interface ID (32-bit)

Packed List of IPv6 DNS Addresses

IPv6 DNS Address (128-bit)

Response:

(Response Type) ACK	(Message Count) 0x00 – 0xFF	(Sub-Device) Copy of Controller SD
(CC) SET_COMMAND_ RESPONSE	(PID) IFC_IPV6_DNS	(PDL) 0x04

(PD)

Interface ID (32-bit)

Data Description:

For GET_COMMAND_RESPONSE, returns a packed list of IPv6 DNS server addresses present on the interface.

For SET_COMMAND, a packed list of IPv6 DNS server addresses which should be present on the interface.

If more IPv6 DNS server addresses are provided than can be accepted, the Responder shall NACK with a NACK Reason Code of NR_IPV6_LIMIT_EXCEEDED.

Interface ID:
The identifier of the interface.

IPv4 DNS Address:
For GET_COMMAND_RESPONSE, returns an IPv6 DNS server address assigned to the interface.

For SET_COMMAND, an IPv6 DNS server address to assign to the interface.

IPv4 DNS server addresses are IPv6 addresses as defined in [IPv6-Addressing].

For example, an IPv6 address represented as “2001:0000:130F:0000:0000:09C0:876A:130B” or the short-form “2001:0:130F::9C0:876A:130B” would be:

0x20010000130F0000000009C0876A130B

8.6 Get IPv6 Supported Modes (IFC_IPV6_SUPPORTED_MODES)

This parameter is used to retrieve the supported IPv6 address assignment modes of an interface.

If the interface is disabled or not present, the Responder shall NACK with a NACK Reason Code of NR_INTERFACE_UNAVAILABLE.

Controller: (GET)

<table>
<tr><td>(Port Id)
0x01 – 0xFF</td><td>(Message Count)
0x00</td><td colspan="2">(Sub-Device)
0x0000 (Root) or 0x0001-0xFFF0</td></tr>
<tr><td>(CC)
GET_COMMAND</td><td colspan="2">(PID)
IFC_IPV6_SUPPORTED_MODES</td><td>(PDL)
0x04</td></tr>
<tr><td colspan="4">(PD)
Interface ID (32-bit)</td></tr>
</table>

Response:

(Response Type) ACK	(Message Count) 0x00 – 0xFF	(Sub-Device) Copy of Controller SD
(CC) GET_COMMAND_ RESPONSE	(PID) IFC_IPV6_SUPPORTED_MODES	(PDL) Variable (0x04-0x07) 0x04 + (Multiple of 0x01)
(PD) Interface ID (32-bit) Packed List of Supported Modes Supported Mode (8-bit)		

Data Description:

A packed list of supported IPv6 address assignment modes for the interface.

Interface ID:
The identifier of the interface.

Supported Mode:
A supported IPv6 address assignment mode for this interface.

Supported Modes are enumerated in Table A-18.

8.7 Get/Set IPv6 Mode (IFC_IPV6_MODE)

This parameter is used to retrieve or set the IPv6 address assignment mode of an interface.

Controller: (GET)

(Port Id) 0x01 – 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001-0xFFF0
(CC) GET_COMMAND	(PID) IFC_IPV6_MODE	(PDL) 0x04
(PD) Interface ID (32-bit)		

Response:

(Response Type) ACK	(Message Count) 0x00 – 0xFF	(Sub-Device) Copy of Controller SD
(CC) GET_COMMAND_ RESPONSE	(PID) IFC_IPV6_MODE	(PDL) 0x05
(PD) Interface ID (32-bit) Mode (8-bit)		

Controller: (SET)

(Port Id) 0x01 – 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001-0xFFF0
(CC) SET_COMMAND	(PID) IFC_IPV6_MODE	(PDL) 0x05
(PD) Interface ID (32-bit) Mode (8-bit)		

Response:

(Response Type) ACK	(Message Count) 0x00 – 0xFF	(Sub-Device) Copy of Controller SD
(CC) SET_COMMAND_ RESPONSE	(PID) IFC_IPV6_MODE	(PDL) 0x04
(PD) Interface ID (32-bit)		

Data Description:

Interface ID:
The identifier of the interface.

Mode:
For GET_COMMAND_RESPONSE, returns the IPv6 address assignment mode for the interface.

For SET_COMMAND, the new IPv6 address assignment mode for the interface. If the mode is not supported, the Responder shall NACK with a NACK Reason Code of NR_MODE_UNSUPPORTED.

Each assignment mode consists of a primary method and a fallback method, which may be none.

Modes are enumerated in Table A-19.

8.8 Get DHCPv6 Server Result (IFC_IPV6_DHCP_RESULT)

This parameter is used to retrieve the DHCPv6 server address which assigned an IPv6 address to an interface.

If the interface is disabled or not present, the Responder shall NACK with a NACK Reason Code of NR_INTERFACE_UNAVAILABLE.

Controller: (GET)

(Port Id) 0x01 – 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001-0xFFF0
(CC) GET_COMMAND	(PID) IFC_IPV6_DHCP_RESULT	(PDL) 0x04
(PD) Interface ID (32-bit)		

Response:

(Response Type) ACK	(Message Count) 0x00 – 0xFF	(Sub-Device) Copy of Controller SD
(CC) GET_COMMAND_ RESPONSE	(PID) IFC_IPV6_DHCP_RESULT	(PDL) 0x19

(PD)

Interface ID (32-bit)
DHCPv6 Address (128-bit)
Lease Time (32-bit)
Flags (8-bit)

Data Description:

Interface ID:
The identifier of the interface.

DHCPv6 Address:
The DHCPv6 server address which assigned an IPv6 address to this interface.

DHCPv6 server addresses are IPv6 addresses as defined in [IPv6-Addressing].

For example, a DHCPv6 server address represented as “2001:0000:130F:0000:0000:09C0:876A:130B” or the short-form “2001:0:130F::9C0:876A:130B” would be:

0x20010000130F0000000009C0876A130B

Lease Time:
The time remaining on the lease of the IPv6 address provided by the DHCPv6 server.

Lease times are in seconds, as defined in [DHCP] Section 3.3. A value of 0xFFFFFFFF represents an infinite lease.

Responders which do not know the remaining time on the lease shall return a value of 0x00000000 and set Valid Lease Time to 0 in Flags. Controllers shall ignore the lease time if the value of Valid Lease Time in Flags is 0.

Flags:
The Flags bit field represent the validity and handling of other fields included in this parameter.

The Flags bit field is enumerated in Table A-20.

9 DNS Configuration Messages

The parameters in this section provide methods for retrieving and setting the DNS capabilities of a Responder.

Throughout this section, DNS refers to the Domain Name System specified in [DNS] and [DNSImplementation].

9.1 Get DNS Capability (IFC_DNS_CAPABILITIES)

This parameter is used to retrieve a list of Interfaces that support setting DNS configuration.

Controller: (GET)

(Port ID) 0x01 – 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001-0xFFF0
(CC) GET_COMMAND	(PID) IFC_DNS_CAPABILITIES	(PDL) 0x00
(PD) Not Present		

Response:

(Response Type) ACK / ACK_OVERFLOW	(Message Count) 0x00 – 0xFF	(Sub-Device) Copy of Controller SD
(CC) GET_COMMAND_ RESPONSE	(PID) IFC_DNS_CAPABILITIES	(PDL) Variable (0x00-0xE4) Multiple of 0x04
(PD) Packed List of Interface IDs Interface ID (32-bit)		

Data Description:

A packed list of Interface IDs.

Responders shall only return Interface IDs that support setting the DNS configuration using the parameters in this section.

Note: Depending on the Responder's operating system, DNS configuration may only exist on the root device (0x0000), while in other cases it may be per-interface.

9.2 Get/Set DNS Hostname (Label) (IFC_DNS_LABEL)

This parameter is used to retrieve or set the DNS hostname (label). For example: "maindns" or "core-services-1".

Note: [DNS] calls the hostname the "label".

If the interface is disabled or not present, the Responder shall NACK with a NACK Reason Code of NR_INTERFACE_UNAVAILABLE.

Controller: (GET)

(Port ID) 0x01 – 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001-0xFFF0
(CC) GET_COMMAND	(PID) IFC_DNS_LABEL	(PDL) 0x04
(PD) Interface ID (32-bit)		

Response:

(Response Type) ACK	(Message Count) 0x00 – 0xFF	(Sub-Device) Copy of Controller SD
(CC) GET_COMMAND_ RESPONSE	(PID) IFC_DNS_LABEL	(PDL) Variable (0x04 – 0x43)
(PD) Interface ID (32-bit) Label (Variable)		

Controller: (SET)

(Port ID) 0x01 – 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001-0xFFF0
(CC) SET_COMMAND	(PID) IFC_DNS_LABEL	(PDL) Variable (0x04 – 0x43)
(PD) Interface ID (32-bit) Label (Variable)		

Response:

(Response Type) ACK	(Message Count) 0x00 – 0xFF	(Sub-Device) Copy of Controller SD
(CC) SET_COMMAND_ RESPONSE	(PID) IFC_DNS_LABEL	(PDL) 0x04
(PD) Interface ID (32-bit)		

Data Description:

Interface ID:
The identifier of the interface.

Label:
The DNS hostname (label) as defined in [DNS] Section 3.1 and [DNSImplementation] Section 2.3.1.

This field shall be a maximum of 63 bytes.

9.3 Get/Set DNS Domain (IFC_DNS_DOMAIN)

This parameter is used to retrieve or set the DNS domain suffix for an interface. For example: “local” or “example.com”.

If the interface is disabled or not present, the Responder shall NACK with a NACK Reason Code of NR_INTERFACE_UNAVAILABLE.

Controller: (GET)

(Port ID) 0x01 – 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001-0xFFF0
(CC) GET_COMMAND	(PID) IFC_DNS_DOMAIN	(PDL) 0x04
(PD) Interface ID (32-bit)		

Response:

(Response Type) ACK	(Message Count) 0x00 – 0xFF	(Sub-Device) Copy of Controller SD
(CC) GET_COMMAND_ RESPONSE	(PID) IFC_DNS_DOMAIN	(PDL) Variable (0x04 – 0xE7)
(PD) Interface ID (32-bit) DNS Domain		

Controller: (SET)

(Port ID) 0x01 – 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001-0xFFF0
(CC) SET_COMMAND	(PID) IFC_DNS_DOMAIN	(PDL) Variable (0x04 – 0xE7)
(PD) Interface ID (32-bit) DNS Domain		

Response:

(Response Type) ACK	(Message Count) 0x00 – 0xFF	(Sub-Device) Copy of Controller SD
(CC) SET_COMMAND_ RESPONSE	(PID) IFC_DNS_DOMAIN	(PDL) 0x04
(PD) Interface ID (32-bit)		

Data Description:

Interface ID:
The identifier of the interface.

DNS Domain:
The DNS domain suffix shall be a valid DNS domain as defined in [DNS] Section 3.1 and [DNSImplementation] Section 2.3.1, minus the DNS Hostname (See Section 9.2).

The domain shall be compliant with "<domain>" as defined in [DNSImplementation] Section 2.3.1. Each node shall be a maximum of 63 bytes.

This field shall be a maximum length of 227 bytes.

Note: Implementors should be aware that due to the maximum length supported by this field, Responders supporting this parameter may need to restrict the length of the DNS domain suffix to less than the overall total domain length of 255 bytes (including length bytes) as defined in [DNSImplementation].

9.4 Get/Set DNS Search Domain (IFC_DNS_SEARCH_DOMAIN)

This parameter is used to retrieve or set the DNS search domains present on an interface.

If the interface is disabled or not present, the Responder shall NACK with a NACK Reason Code of NR_INTERFACE_UNAVAILABLE.

Controller: (GET)

(Port Id) 0x01 – 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001-0xFFF0
(CC) GET_COMMAND	(PID) IFC_DNS_SEARCH_DOMAIN	(PDL) 0x04
(PD)		

Interface ID (32-bit)

Response:

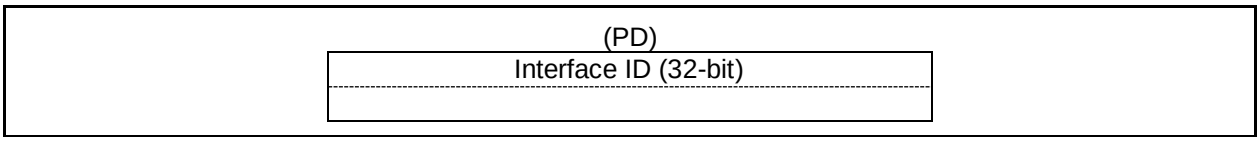
(Response Type) ACK	(Message Count) 0x00 – 0xFF	(Sub-Device) Copy of Controller SD	
(CC) GET_COMMAND_ RESPONSE	(PID) IFC_DNS_SEARCH_DOMAIN		(PDL) Variable (0x04-0xE4)
(PD) Interface ID (32-bit) Packed List of DNS Search Domains DNS Search Domain			

Controller: (SET)

(Port Id) 0x01 – 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001-0xFFF0	
(CC) SET_COMMAND	(PID) IFC_DNS_SEARCH_DOMAIN		(PDL) Variable (0x04-0xE4)
(PD) Interface ID (32-bit) Packed List of DNS Search Domains DNS Search Domain			

Response:

(Response Type) ACK	(Message Count) 0x00 – 0xFF	(Sub-Device) Copy of Controller SD	
(CC) SET_COMMAND_ RESPONSE	(PID) IFC_DNS_SEARCH_DOMAIN		(PDL) 0x04



Data Description:

For GET_COMMAND_RESPONSE, returns a packed list of DNS search domains present on the interface.

For SET_COMMAND, a packed list of DNS search domains which should be present on the interface.

If more DNS search domains are provided than can be accepted, the Responder shall NACK with a NACK Reason Code of NR_DNS_LIMIT_EXCEEDED.

Interface ID:
The identifier of the interface.

DNS Search Domain:
DNS search domains shall be valid DNS domains as defined in [DNS] Section 3.1 and [DNSImplementation] Section 2.3.1.

Search domains shall be compliant with “<domain>” as defined in [DNSImplementation] Section 2.3.1.

This field shall be a maximum length of 56 bytes, therefore a maximum of 4 DNS search domains are supported within the packed list.

For DNS search domains less than 56 bytes, the trailing bytes shall be padded with 0x00 to 56 bytes.

Note: Implementors should be aware that due to the maximum length supported by this field, Responders supporting this parameter should restrict the length of DNS search domains to less than the overall total domain length of 255 bytes (including length bytes) as defined in [DNSImplementation].

10 VLAN Configuration Messages

The parameters in this section provide methods for retrieving and setting the VLAN capabilities of a Responder.

Throughout this section, the term “VLAN” refers to the VLAN Identifier contained in 802.1Q tags as defined in [VLAN], and the term “Padded VLAN ID” refers to the 12-bit VID component of an 802.1Q tag zero-padded to 16 bits. The term “tagged” refers to frames which have an 802.1Q tag applied, while “native” or “untagged” indicates frames without such a tag. See Figure 10-1.

<table>
<tr><td>16 Bits</td><td>3 Bits</td><td>1 Bit</td><td>12 Bits</td></tr>
<tr><td rowspan="2">Tag Protocol Identifier (TPID)</td><td colspan="3">Tag Control Information (TCI)</td></tr>
<tr><td>Priority Code Point (PCP)</td><td>Drop Eligible Indicator</td><td>VLAN Identifier</td></tr>
</table>

Figure 10-1: 802.1Q Tag Format

VIDs are 12-bit values in the range 0 to 4095 (0x000 – 0xFFF) inclusive. 0 (0x000) and 4095 (0xFFF) are reserved and VID 4094 (0xFFE) is often reserved by manufacturers.

Padded VLAN IDs shall be a 16-bit value in the range 0 (0x0000) to 4093 (0x0FFD) inclusive, i.e., padding the VID with a leading zero. 0 (0x0000) is a special case meaning no Padded VLAN ID.

For example, a VLAN identifier of 1 (0x001) shall be encoded as 0x0001, while a VLAN identifier of 4093 (0xFFD) shall be encoded as 0x0FFD.

10.1 Get VLAN Capabilities (IFC_VLAN_CAPABILITIES)

This parameter is used to retrieve the VLAN capabilities for an interface.

If the interface is disabled or not present, the Responder shall NACK with a NACK Reason Code of NR_INTERFACE_UNAVAILABLE.

Controller: (GET)

<table>
<tr><td>(Port Id)
0x01 – 0xFF</td><td>(Message Count)
0x00</td><td colspan="2">(Sub-Device)
0x0000 (Root) or 0x0001-0xFFF0</td></tr>
<tr><td>(CC)
GET_COMMAND</td><td colspan="2">(PID)
IFC_VLAN_CAPABILITIES</td><td>(PDL)
0x04</td></tr>
<tr><td colspan="4">(PD)
Interface ID (32-bit)</td></tr>
</table>

Response:

(Response Type) ACK	(Message Count) 0x00 – 0xFF	(Sub-Device) Copy of Controller SD
(CC) GET_COMMAND_ RESPONSE	(PID) IFC_VLAN_CAPABILITIES	(PDL) 0x07
(PD) Interface ID (32-bit) Capabilities (8-bit) Tagged Quantity (16-bit)		

Data Description:

Interface ID:
The identifier of the interface.

Capabilities:
This bit field indicates VLAN capabilities of the interface.

Note: If tagged VLANs are supported, the quantity supported is indicated by the Tagged Quantity field.

The Capabilities bit field is enumerated in Table A-21.

Tagged Quantity:
The number of VLANs which may be tagged on the interface.

Responders shall return a value of 0x0000 if they indicate they do not support tagged VLANs using the Capabilities field. Controllers shall ignore this field if the Capabilities field indicates tagged VLANs are not supported.

Responders shall return a value between 0x0000 to 0x0FFD (0 – 4093). Controllers shall ignore any value greater than 0x0FFD (4093).

10.2 Get/Set VLAN Configuration (IFC_VLAN_CONFIGURATION)

This parameter is used to retrieve or set the VLAN configuration of an interface.

If the interface is disabled or not present, the Responder shall NACK with a NACK Reason Code of NR_INTERFACE_UNAVAILABLE.

Controller: (GET)

(Port Id) 0x01 – 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001-0xFFF0
(CC) GET_COMMAND	(PID) IFC_VLAN_CONFIGURATION	(PDL) 0x04
(PD) Interface ID (32-bit)		

Response:

(Response Type) ACK	(Message Count) 0x00 – 0xFF	(Sub-Device) Copy of Controller SD
(CC) GET_COMMAND_ RESPONSE	(PID) IFC_VLAN_CONFIGURATION	(PDL) Variable (0x06 – 0xE6)
(PD) Interface ID (32-bit) Native Padded VLAN ID (16-bit) Packed List of Tagged VLANs Tagged Padded VLAN ID (16-bit) Tagged VLAN Flags (8-bit) Tagged VLAN Interface ID (32-bit)		

Controller: (SET)

(Port ID) 0x01 – 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001-0xFFF0
(CC) SET_COMMAND	(PID) IFC_VLAN_CONFIGURATION	(PDL) Variable (0x06 – 0xE6)
(PD) Interface ID (32-bit) Native Padded VLAN ID (16-bit) Packed List of Tagged VLANs Tagged Padded VLAN ID (16-bit)		

Response:

(Response Type) ACK	(Message Count) 0x00 – 0xFF	(Sub-Device) Copy of Controller SD
(CC) SET_COMMAND_ RESPONSE	(PID) IFC_VLAN_CONFIGURATION	(PDL) 0x04
(PD) Interface ID (32-bit)		

Data Description:

Interface ID:
The identifier of the interface.

Native Padded VLAN ID:
For GET_COMMAND_RESPONSE, returns the native (untagged) Padded VLAN ID assigned to the interface.

For SET_COMMAND, the native (untagged) Padded VLAN ID which should be assigned to the interface.

If a Responder indicated it does not support assigning a native (untagged) VLAN to the interface in its response to IFC_VLAN_CAPABILITIES, Controllers and Responders shall set this field to 0 (0x0000) and ignore it.

If a Responder indicated it supports assigning native (untagged) VLANS but does not support assigning tagged VLANs to the interface in its response to IFC_VLAN_CAPABILITIES, Controllers and Responders shall not set this field to 0 (0x0000).

If a Responder indicated it supports assigning both native (untagged) and tagged VLANs to the interface in its response to IFC_VLAN_CAPABILITIES:

For GET_COMMAND_RESPONSE, a value of 0 (0x0000) shall mean no native (untagged) VLAN is assigned to the interface. Responders shall include at least 1 item in the packed list of Tagged VLANs when returning a value of 0 (0x0000).

For SET_COMMAND, a value of 0 (0x0000) shall mean no native (untagged) VLAN should be assigned to the interface. Responders shall include at least 1 item in the packed list of Tagged VLANs when setting a value of 0 (0x0000).

Tagged VLAN:
If a Responder indicated support for assigning tagged VLANs in its response to IFC_VLAN_CAPABILITIES, the count of items in the packed list shall be no greater than the Tagged Quantity indicated in the same response. If a Responder indicated it does not support assigning tagged VLANs, the count shall be 0.

If more items are included in the packed list than indicated, Controllers and Responders shall NACK with a NACK Reason Code of NR_VLAN_LIMIT_EXCEEDED.

Tagged Padded VLAN ID:
For GET_COMMAND_RESPONSE, returns a tagged Padded VLAN ID assigned to the interface.

For SET_COMMAND, a tagged Padded VLAN ID which should be assigned to the interface.

Tagged VLAN Flags:
Flags indicating the validity and handling of other fields included in this parameter.

Valid Interface ID (Bit 0):
This bit field Indicates whether Tagged VLAN Interface ID is valid.

The Tagged VLAN Flags bit field is enumerated in Table A-22.

Tagged VLAN Interface ID:
The identifier of the virtual interface created for the Tagged Padded VLAN ID.

Some devices may create additional virtual interfaces which are associated with each tagged VLAN. If no interface was created, or this information cannot be provided, Responders shall return a value of 0 (0x00000000) and set Valid Interface ID (Bit 0) in Tagged VLAN Flags to 0. Controllers shall ignore this field if Valid Interface ID (Bit 0) is set to 0.

10.3 Get/Set VLAN Label (IFC_VLAN_LABEL)

This parameter is used to retrieve or set a label for a VLAN. For example: “Management” or “Lighting Network”.

If the interface is disabled or not present, the Responder shall NACK with a NACK Reason Code of NR_INTERFACE_UNAVAILABLE.

Controller: (GET)

(Port ID) 0x01 – 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001-0xFFF0
(CC) GET_COMMAND	(PID) IFC_VLAN_LABEL	(PDL) 0x06
(PD) Interface ID (32-bit) Padded VLAN ID (16-bit)		

Response:

(Response Type) ACK	(Message Count) 0x00 – 0xFF	(Sub-Device) Copy of Controller SD
(CC) GET_COMMAND_ RESPONSE	(PID) IFC_VLAN_LABEL	(PDL) Variable (0x06 – 0xE7)
(PD) Interface ID (32-bit) Padded VLAN ID (16-bit) Label (Variable)		

Controller: (SET)

(Port ID) 0x01 – 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001-0xFFF0
(CC) SET_COMMAND	(PID) IFC_VLAN_LABEL	(PDL) Variable (0x06 – 0xE7)
(PD) Interface ID (32-bit) Padded VLAN ID (16-bit) Label (Variable)		

Response:

(Response Type) ACK	(Message Count) 0x00 – 0xFF	(Sub-Device) Copy of Controller SD
(CC) SET_COMMAND_ RESPONSE	(PID) IFC_VLAN_LABEL	(PDL) 0x06
(PD) Interface ID (32-bit) Padded VLAN ID (16-bit)		

Data Description:

Devices may maintain a label for each VLAN per interface or may contain a single label for that VLAN across the device. Implementors should be aware that changes to VLAN labels on a Responder may impact other interfaces to which that VLAN is assigned. Responders that only maintain a single label for a VLAN may ignore the Interface ID provided.

In many configurations, VLAN labels are synchronized to all devices on the network. Implementors should be aware that changes to VLAN labels on a Responder may impact other Responders and network devices.

For SET_COMMAND, if the Label string is longer than supported, the Responder shall NACK with a NACK Reason Code of NR_DATA_OUT_OF_RANGE as defined in [RDM].

Interface ID:
The identifier of the interface.

Label:
The Label is a text field as defined in Section 3.3.

Appendix A: Defined Parameters (Normative)

Table A-1: RDM Parameter ID Defines

GET Allowed	SET Allowed	RDM Parameter ID's (Slot 21-22)	Value	Required Root	Required Sub-Device	Packing Disallowed
✓	✓	IFC_SESSION_CONTROL	0x			✓
✓		IFC_INTERFACE_LIST	0x			✓
		IFC_INTERFACE_TOPOLOGY	0x			✓
✓		IFC_INTERFACE_PARENT	0x			✓
✓	✓	IFC_INTERFACE_ENABLE	0x			✓
✓		IFC_INTERFACE_HARDWARE_ADDRESS	0x			✓
✓		IFC_INTERFACE_FIXED_LABEL	0x			✓
✓	✓	IFC_INTERFACE_USER_LABEL	0x			✓
✓		IFC_INTERFACE_TYPE	0x			✓
✓		IFC_INTERFACE_LINE_SPEED	0x			✓
✓		IFC_INTERFACE_UTILIZATION	0x			✓
✓		IFC_INTERFACE_DUPLEX_MODE	0x			✓
✓	✓	IFC_INTERFACE_EEE	0x			✓
✓	✓	IFC_INTERFACE_POWER_ENABLE	0x			✓
✓		IFC_INTERFACE_POWER_STATUS	0x			✓
✓	✓	IFC_IPV4_ADDRESS	0x			✓
✓	✓	IFC_IPV4_GATEWAY	0x			✓
	✓	IFC_IPV4_GATEWAY_REMOVE	0x			✓
✓	✓	IFC_IPV4_DNS	0x			✓
✓		IFC_IPV4_SUPPORTED_MODES	0x			✓
✓	✓	IFC_IPV4_MODE	0x			✓
✓		IFC_IPV4_DHCP_RESULT	0x			✓
✓	✓	IFC_IPV6_ADDRESS	0x			✓
✓		IFC_IPV6_ADDRESS_LINKLOCAL	0x			✓
✓	✓	IFC_IPV6_GATEWAY	0x			✓
	✓	IFC_IPV6_GATEWAY_REMOVE	0x			✓
✓	✓	IFC_IPV6_DNS	0x			✓
✓		IFC_IPV6_SUPPORTED_MODES	0x			✓
✓	✓	IFC_IPV6_MODE	0x			✓
✓		IFC_IPV6_DHCP_RESULT	0x			✓
✓		IFC_DNS_CAPABILITIES	0x			✓
✓	✓	IFC_DNS_LABEL	0x			✓
✓	✓	IFC_DNS_DOMAIN	0x			✓
✓	✓	IFC_DNS_SEARCH_DOMAIN	0x			✓
✓		IFC_VLAN_CAPABILITIES	0x			✓
✓	✓	IFC_VLAN_CONFIGURATION	0x			✓
✓	✓	IFC_VLAN_LABEL	0x			✓

PID Values will be assigned after the Standard has been ratified.

Table A-2: Defines

Define	Value
SESSION_WINDOW	60 seconds

Table A-3: Session Operations

Session Operations	Value	Comment
DISCARD	0x00	The Responder shall discard any changes.
COMMIT_RUNNING	0x01	The Responder shall commit any changes to running configuration.
COMMIT_RUNNING_PERSISTENT	0x02	The Responder shall commit any changes to running and persistent configuration.
CONFIRM	0x03	The Responder changes shall not be rolled back.

Table A-4: Topology Flags

Topology Flag	Name	Description
Bit 0 (0x01)	Topology Changed	Whether a change has occurred to the interface topology of the Responder.
Bit 1 (0x02)	Reserved	Reserved for future use. Responders shall set this bit to 0, Controllers shall ignore.
Bit 2 (0x04)	Reserved	Reserved for future use. Responders shall set this bit to 0, Controllers shall ignore.
Bit 3 (0x08)	Reserved	Reserved for future use. Responders shall set this bit to 0, Controllers shall ignore.
Bit 4 (0x10)	Reserved	Reserved for future use. Responders shall set this bit to 0, Controllers shall ignore.
Bit 5 (0x20)	Reserved	Reserved for future use. Responders shall set this bit to 0, Controllers shall ignore.
Bit 6 (0x40)	Reserved	Reserved for future use. Responders shall set this bit to 0, Controllers shall ignore.
Bit 7 (0x80)	Reserved	Reserved for future use. Responders shall set this bit to 0, Controllers shall ignore.

Table A-5: Interface States

Interface State	Value	Description
DISABLE	0x00	Get: The interface is disabled. Set: The interface should be disabled.
ENABLE	0x01	Get: The interface is enabled. Set: The interface should be enabled.
BOUNCE	0x02	Set: The interface should be disabled, then enabled after the specified time.
NOT_PRESENT	0x03	Get: The interface is not present.
UNKNOWN	0x04	Get: The interface state is unknown.

Table A-6: Hardware Address Types

Hardware Address Type	Value	Hardware Address Length	Description
EUI48	0x00	0x06	EUI-48, otherwise known as a MAC address.
EUI64	0x01	0x08	EUI-64

Table A-7: Interface Types

Interface Type	Value	Description
UNKNOWN	0x00	The interface type is not specified by the device.
VIRTUAL	0x01	The interface is virtual and has no physical presence.
WIRED	0x02	The interface is electrical. Data is carried by one or more electrical conductors.
OPTICAL_FIBER	0x03	The interface is optical. Data is carried by one or more fiber optic transmission paths.
WIRELESS_RF	0x04	The interface is wireless. Data is transmitted via radio frequency via one or more antennas.
WIRELESS_OPTICAL	0x05	The interface is wireless. Data is transmitted via a line-of-sight optical interface such as laser or infrared emitter/detector pairs.
COMBINATION	0x06	The interface can be configured to use more than one medium (e.g. Copper or Optical combo port).

Table A-8: Duplex Modes

Duplex Mode	Value	Comment
DUPLEX_MODE_HALF	0x00	The interface can transmit and receive, but not at the same time.
DUPLEX_MODE_FULL	0x01	The interface can transmit and receive at the same time.

Table A-9: EEE States

EEE States	Value	Comment
DISABLE	0x00	Energy Efficient Ethernet is disabled.
ENABLE	0x01	Energy Efficient Ethernet is enabled.

Table A-10: Power States

Interface State	Value	Description
DISABLE	0x00	Get: Interface power is disabled. Set: Interface power should be disabled.
ENABLE	0x01	Get: Interface power is enabled. Set: Interface power should be enabled.

Table A-11: IFC Power Type

Interface State	Value	Description
NONE	0x00	Interface does not support power.
POE_2P_TYPE_1	0x01	Type 1 802.3af 2-pair 15.4W Power over Ethernet
POE_2P_TYPE_2	0x02	Type 2 802.3at 2-pair 30W Power over Ethernet
POE_4P_TYPE_3	0x03	Type 3 802.3bt 4-pair 60W Power over Ethernet
POE_4P_TYPE_4	0x04	Type 4 802.3bt 4-pair 90W Power over Ethernet
SPOE_PODL	0x05	802.3bu Power over Data Line for Single-Pair Ethernet
NON_STANDARD	0x06	Non-standard powering technology

Table A-12: IFC Power Role

Interface State	Value	Description
NONE	0x00	Interface does not support power.
POWER_SOURCING_EQUIPMENT	0x01	Interface is capable of supplying power.
POWERED_DEVICE	0x02	Interface is capable of being powered by Power Sourcing Equipment.
OTHER	0x03	Interface has power features but is neither Power Sourcing Equipment nor a Powered Device.

Table A-13: IPv4 Sources

Source	Value	Description
DHCP	0x00	The IPv4 address was provided by DHCP.
STATIC	0x01	The IPv4 address is a static assignment.
ZEROCONF	0x02	The IPv4 was assigned by a zero-configuration method.
BOOTP	0x03	The IPv4 address was provided by BOOTP.

Table A-14: IPv4 Supported Modes

Supported Mode	Value	Description
DHCP	0x00	The interface supports assigning an IPv4 address using DHCP.
STATIC	0x01	The interface supports assigning a static IPv4 address.
ZEROCONF	0x02	The interface supports assigning an IPv4 address using some zero-configuration method.
BOOTP	0x03	The interface supports assigning an IPv4 address using BOOTP.

Table A-15: IPv4 Modes

Mode	Value	Description
DHCP_NONE	0x00	An attempt to get an IPv4 address via DHCP shall be performed first, with no fallback.
DHCP_ZEROCONF	0x01	An attempt to get an IPv4 address via DHCP shall be performed first, followed by assigning an address using zero-configuration.
DHCP_STATIC	0x02	An attempt to get an IPv4 address via DHCP shall be performed first, followed by using static IPv4 address assignment.
DHCP_BOOTP	0x03	An attempt to get an IPv4 address via DHCP shall be performed first, followed by an attempt to get an IPv4 address via BOOTP.
BOOTP_STATIC	0x04	An attempt to get an IPv4 address via BOOTP shall be performed first, followed by using static IPv4 address assignment.
BOOTP_NONE	0x05	An attempt to get an IPv4 address via BOOTP shall be performed first, with no fallback.
STATIC_NONE	0x06	A static IPv4 address assignment shall be used, with no fallback.
ZEROCONF_NONE	0x07	An attempt to assign an IPv4 address using zero-configuration shall be performed first, with no fallback.

Table A-16: DHCPv4 Flags

DHCPv4 Flag	Name	Description
Bit 0 (0x01)	Valid Address	Whether an IPv4 address was received from the DHCPv4 server address.
Bit 1 (0x02)	Valid Lease Time	Whether the lease time provided is valid.
Bit 2 (0x04)	Reserved	Reserved for future use. Responders shall set this bit to 0, Controllers shall ignore.
Bit 3 (0x08)	Reserved	Reserved for future use. Responders shall set this bit to 0, Controllers shall ignore.
Bit 4 (0x10)	Reserved	Reserved for future use. Responders shall set this bit to 0, Controllers shall ignore.
Bit 5 (0x20)	Reserved	Reserved for future use. Responders shall set this bit to 0, Controllers shall ignore.
Bit 6 (0x40)	Reserved	Reserved for future use. Responders shall set this bit to 0, Controllers shall ignore.
Bit 7 (0x80)	Reserved	Reserved for future use. Responders shall set this bit to 0, Controllers shall ignore.

Table A-17: IPv6 Sources

Source	Value	Description
DHCP	0x00	The IPv6 address was provided by DHCPv6.
STATIC	0x01	The IPv6 address is a static assignment.
SLAAC	0x02	The IPv4 was assigned by Stateless Address Auto-configuration (SLAAC).

Table A-18: IPv6 Supported Modes

Supported Mode	Value	Description
DHCP	0x00	The interface supports assigning an IPv6 address using DHCP.
STATIC	0x01	The interface supports assigning a static IPv6 address.
SLAAC	0x02	The interface supports assigning an IPv6 address using Stateless Address Auto-configuration (SLAAC).

Table A-19: IPv6 Mode Defines

Method Define	Value	Description
DHCP_NONE	0x00	An attempt to get an IPv6 address via DHCP shall be performed first, with no fallback.
DHCP_SLAAC	0x01	An attempt to get an IPv6 address via DHCP shall be performed first, followed by assigning an address using Stateless Address Auto-configuration (SLAAC).
DHCP_STATIC	0x02	An attempt to get an IPv6 address via DHCP shall be performed first, followed by using static IPv6 address assignment.
SLAAC_NONE	0x03	An attempt to get an IPv6 address via Stateless Address Auto-configuration (SLAAC) shall be performed first, with no fallback.
SLAAC_DHCP	0x04	An attempt to get an IPv6 address via Stateless Address Auto-configuration (SLAAC) shall be performed first, followed by an attempt to get an IPv6 address via DHCP.
SLAAC_STATIC	0x05	An attempt to get an IPv6 address via Stateless Address Auto-configuration (SLAAC) shall be performed first, followed by using static IPv6 address assignment.
STATIC_NONE	0x06	A static IPv6 address assignment shall be used, with no fallback.

Table A-20: DHCPv6 Flags

DHCPv6 Flag	Name	Description
Bit 0 (0x01)	Valid Address	Whether an IPv6 address was received from the DHCPv6 server address.
Bit 1 (0x02)	Valid Lease Time	Whether the lease time provided is valid.
Bit 2 (0x04)	Reserved	Reserved for future use. Responders shall set this bit to 0, Controllers shall ignore.
Bit 3 (0x08)	Reserved	Reserved for future use. Responders shall set this bit to 0, Controllers shall ignore.
Bit 4 (0x10)	Reserved	Reserved for future use. Responders shall set this bit to 0, Controllers shall ignore.
Bit 5 (0x20)	Reserved	Reserved for future use. Responders shall set this bit to 0, Controllers shall ignore.
Bit 6 (0x40)	Reserved	Reserved for future use. Responders shall set this bit to 0, Controllers shall ignore.
Bit 7 (0x80)	Reserved	Reserved for future use. Responders shall set this bit to 0, Controllers shall ignore.

Table A-21: VLAN Capabilities

VLAN Capabilities	Name	Description
Bit 0 (0x01)	Native	Whether a native (untagged) VLAN may be assigned to the interface.
Bit 1 (0x02)	Tagged	Whether tagged VLANs may be assigned to the interface.
Bit 2 (0x04)	Reserved	Reserved for future use. Responders shall set this bit to 0, Controllers shall ignore.
Bit 3 (0x08)	Reserved	Reserved for future use. Responders shall set this bit to 0, Controllers shall ignore.
Bit 4 (0x10)	Reserved	Reserved for future use. Responders shall set this bit to 0, Controllers shall ignore.
Bit 5 (0x20)	Reserved	Reserved for future use. Responders shall set this bit to 0, Controllers shall ignore.
Bit 6 (0x40)	Reserved	Reserved for future use. Responders shall set this bit to 0, Controllers shall ignore.
Bit 7 (0x80)	Reserved	Reserved for future use. Responders shall set this bit to 0, Controllers shall ignore.

Table A-22: VLAN Tagged VLAN Flags

VLAN Capabilities	Name	Description
Bit 0 (0x01)	Valid Interface ID	Whether the Interface ID provided is valid.
Bit 1 (0x02)	Reserved	Reserved for future use. Responders shall set this bit to 0, Controllers shall ignore.
Bit 2 (0x04)	Reserved	Reserved for future use. Responders shall set this bit to 0, Controllers shall ignore.
Bit 3 (0x08)	Reserved	Reserved for future use. Responders shall set this bit to 0, Controllers shall ignore.
Bit 4 (0x10)	Reserved	Reserved for future use. Responders shall set this bit to 0, Controllers shall ignore.
Bit 5 (0x20)	Reserved	Reserved for future use. Responders shall set this bit to 0, Controllers shall ignore.
Bit 6 (0x40)	Reserved	Reserved for future use. Responders shall set this bit to 0, Controllers shall ignore.
Bit 7 (0x80)	Reserved	Reserved for future use. Responders shall set this bit to 0, Controllers shall ignore.

Table A-23: Additional Response NACK Reason Codes

Additional Response NACK Reason Codes*	Value	Description
NR_INTERFACE_UNAVAILABLE	0x	The interface is disabled or not present.
NR_INTERFACE_INVALID_SESSION	0x	Another session is currently active.
NR_IPV4_LIMIT_EXCEEDED	0x	More IPv4 addresses or IPv4 DNS server addresses were provided than the interface supports.
NR_IPV6_LIMIT_EXCEEDED	0x	More IPv6 addresses or IPv6 DNS server addresses were provided than the interface supports.
NR_VLAN_LIMIT_EXCEEDED	0x	More Tagged VLANs are included in the packed list provided than the maximum indicated by in the capabilities field.
NR_MODE_UNSUPPORTED	0x	The IPv4 or IPv6 address mode is not supported.
NR_GET_QUEUED_ONLY	0x	
		*These are in addition to NACK Reason Codes defined in Table A-17 of E1.20.

NR Code Values will be assigned after the Standard has been ratified.