

# Superseded by New Specification CCIF2.0

## OpenCable™ Multi-Stream CableCARD Interface Specification

**OC-SP-MC-IF-I02-040831**

**ISSUED**

### **Notice**

This document is the result of a cooperative effort undertaken at the direction of Cable Television Laboratories, Inc. for the benefit of the cable industry and its customers. This document may contain references to other documents not owned or controlled by CableLabs. Use and understanding of this document may require access to such other documents. Designing, manufacturing, distributing, using, selling, or servicing products, or providing services, based on this document may require intellectual property licenses from third parties for technology referenced in the document.

Neither CableLabs nor any member company is responsible to any party for any liability of any nature whatsoever resulting from or arising out of use or reliance upon this document, or any document referenced herein. This document is furnished on an "AS IS" basis and neither CableLabs nor its members provides any representation or warranty, express or implied, regarding the accuracy, completeness, or fitness for a particular purpose of this document, or any document referenced herein.

© Copyright 2003-2004 Cable Television Laboratories, Inc.  
All rights reserved.

## Document Status Sheet

<b>Document Control Number:</b>	OC-SP-MC-IF-I02-040831			
<b>Document Title:</b>	OpenCable™ Multi-Stream CableCARD Interface Specification			
<b>Revision History:</b>	I02 – August 31, 2004			
	I01 – September 5, 2003			
<b>Date:</b>	August 31, 2004			
<b>Responsible Editor</b>	Steve Young			
<b>Status:</b>	<del>Work in Progress</del>	<del>Draft</del>	Issued	<del>Closed</del>
<b>Distribution Restrictions:</b>	<del>Author Only</del>	<del>CL/Member</del>	<del>CL/Member/Vendor</del>	Public

### Key to Document Status Codes:

- Work in Progress** An incomplete document, designed to guide discussion and generate feedback, which may include several alternative requirements for consideration.
- Draft** A document in specification format considered largely complete, but lacking review by Members and vendors. Drafts are susceptible to substantial change during the review process.
- Issued** A document that has undergone rigorous Member and vendor review, suitable for product design and development, cross-vendor interoperability, and for certification testing.
- Closed** A stable document, reviewed, tested and validated, suitable to enable cross-vendor interoperability.

### TRADE MARKS:

DOCSIS®, eDOCSIS™, PacketCable™, CableHome®, CableOffice™, OpenCable™, CableCARD™, and CableLabs® are trademarks of Cable Television Laboratories, Inc.

# Contents

<b>1 INTRODUCTION AND OVERVIEW</b> .....	<b>1</b>
1.1 Scope .....	2
1.2 Historical Perspective (Informative).....	3
1.3 Organization of Document .....	3
1.4 Requirements (Conformance Notation).....	4
1.5 Numerical.....	5
1.6 Abbreviations and Acronyms .....	5
<b>2 REFERENCES</b> .....	<b>8</b>
2.1 Normative References .....	8
2.2 Informative References .....	8
2.3 Reference Acquisition .....	9
<b>3 MODEL OF OPERATION (INFORMATIVE)</b> .....	<b>10</b>
3.1 CableCARD Device Functional Description.....	10
3.2 Physical.....	12
3.3 Network Connectivity .....	12
3.4 SCTE 28 Backward Compatible Mode.....	12
3.4.1 MPEG Data Flow.....	13
3.4.2 OOB Data Flow .....	13
3.5 Multi-Stream CableCARD Mode .....	14
3.5.1 MPEG Data Flow.....	14
3.5.2 OOB Data Flow .....	15
3.6 Multiple CableCARD Device Support .....	15
<b>4 PHYSICAL INTERFACE (NORMATIVE)</b> .....	<b>16</b>
4.1 Interface Functional Description .....	16
4.1.1 Connector Pin Assignment.....	16
4.1.2 Multi-Stream CableCARD Device Signal Descriptions.....	18
4.1.3 CableCARD Device Type Identification.....	19
4.1.4 Card Information Structure .....	22
4.1.5 MPEG Transport Interface.....	22
4.1.6 CableCARD-Host OOB Interface .....	25
4.1.7 CPU Interface.....	25
4.2 Electrical Specifications .....	31
4.2.1 DC Characteristics .....	31
4.2.2 AC Characteristics.....	34
4.3 Mechanical Specifications .....	39
4.3.1 Form Factor .....	39
4.3.2 Connector.....	39

4.3.3 Environmental .....	39
<b>5 COPY PROTECTION.....</b>	<b>40</b>
<b>6 COMMAND CHANNEL OPERATION (NORMATIVE).....</b>	<b>41</b>
<b>6.1 Link Layer .....</b>	<b>41</b>
6.1.1 Backward Compatible Mode.....	41
6.1.2 Multi-Stream CableCARD Mode.....	41
<b>6.2 Transport Layer.....</b>	<b>41</b>
6.2.1 Backward Compatible Mode.....	41
6.2.2 Multi-Stream CableCARD Mode.....	41
<b>6.3 Session Layer.....</b>	<b>41</b>
6.3.1 Backward Compatible Mode.....	42
6.3.2 Multi-Stream CableCARD Mode.....	42
<b>6.4 Application Layer .....</b>	<b>46</b>
6.4.1 Resource Identifier Structure.....	46
<b>6.5 APDUs .....</b>	<b>47</b>
6.5.1 Backward Compatibility Mode .....	47
6.5.2 Multi-Stream CableCARD Mode.....	48
<b>6.6 Resource Manager .....</b>	<b>49</b>
6.6.1 profile_inq() .....	49
6.6.2 profile_reply() .....	50
6.6.3 profile_changed() .....	50
<b>6.7 Application Information.....</b>	<b>50</b>
6.7.1 application_info_req().....	51
6.7.2 application_info_cnf() .....	53
6.7.3 server_query() .....	54
6.7.4 server_reply() .....	55
<b>6.8 Low Speed Communication .....</b>	<b>56</b>
<b>6.9 CA Support .....</b>	<b>57</b>
6.9.1 ca_info_inquiry .....	57
6.9.2 ca_info.....	58
6.9.3 ca_pmt.....	59
6.9.4 ca_pmt_reply.....	63
6.9.5 ca_update.....	65
<b>6.10 Host Control.....</b>	<b>66</b>
6.10.1 OOB_TX_tune_req.....	67
6.10.2 OOB_TX_tune_cnf .....	67
6.10.3 OOB_RX_tune_req .....	68
6.10.4 OOB_RX_tune_cnf.....	69
6.10.5 inband_tune_req .....	69
6.10.6 inband_tune_cnf.....	70
<b>6.11 System Time .....</b>	<b>71</b>
6.11.1 system_time_inq .....	71
6.11.2 system_time .....	72
<b>6.12 Man-Machine Interface (MMI) .....</b>	<b>72</b>

6.12.1 open_mmi_req .....	73
6.12.2 open_mmi_cnf.....	74
6.12.3 close_mmi_req .....	75
6.12.4 close_mmi_cnf .....	75
<b>6.13 CableCARD Device Capability Discovery .....</b>	<b>76</b>
6.13.1 stream_profile APDU .....	77
6.13.2 stream_profile_cnf APDU .....	77
6.13.3 request_pids_cnf APDU .....	80
<b>6.14 Copy Protection.....</b>	<b>81</b>
<b>6.15 Extended Channel Support .....</b>	<b>81</b>
6.15.1 new_flow_req APDU .....	83
6.15.2 new_flow_cnf APDU.....	85
6.15.3 delete_flow_req APDU .....	86
6.15.4 delete_flow_cnf APDU .....	86
6.15.5 lost_flow_ind APDU.....	87
6.15.6 lost_flow_cnf APDU.....	87
6.15.7 inquire_DSG_mode APDU .....	88
6.15.8 set_DSG_mode APDU .....	88
6.15.9 inquire_multi_mode APDU .....	90
6.15.10 set_multi_mode APDU .....	91
6.15.11 DSG_packet_error APDU.....	91
<b>6.16 Generic Feature Control .....</b>	<b>92</b>
6.16.1 Parameter Storage .....	92
6.16.2 Parameter Operation .....	92
6.16.3 Generic Feature Control Resource Identifier.....	95
6.16.4 Feature ID .....	95
6.16.5 Generic Feature Control APDUs .....	96
<b>6.17 Generic Diagnostic Support.....</b>	<b>105</b>
6.17.1 diagnostic_req APDU .....	106
6.17.2 diagnostic_cnf APDU .....	106
6.17.3 Diagnostic Report Definition.....	108
<b>6.18 Specific Application Support .....</b>	<b>115</b>
6.18.1 SAS_connect_rqst APDU.....	117
6.18.2 SAS_connect APDU.....	117
6.18.3 SAS_data_rqst APDU .....	118
6.18.4 SAS_data_av APDU.....	118
6.18.5 SAS_data_cnf APDU.....	119
6.18.6 SAS_server_query APDU .....	119
6.18.7 SAS_server_reply APDU.....	119
<b>6.19 CableCARD Device Firmware Upgrade .....</b>	<b>120</b>
<b>6.20 Support for Common Download Specification.....</b>	<b>120</b>
<b>7 EXTENDED CHANNEL OPERATION (NORMATIVE) .....</b>	<b>121</b>
<b>7.1 Link Layer .....</b>	<b>121</b>
7.1.1 Maximum PDUs.....	121
<b>7.2 Modem Models .....</b>	<b>122</b>

7.2.1 Unidirectional Host Model .....	122
7.2.2 Bidirectional With Modem in CableCARD Device .....	122
7.2.3 Bidirectional With Modem in Host .....	122
<b>7.3 SI Requirements .....</b>	<b>122</b>
<b>7.4 EAS Requirements .....</b>	<b>122</b>
<b>APPENDIX A REFERENCE TABLES .....</b>	<b>123</b>
<b>A.1 Multi-Stream CableCARD Device Resource Identifiers .....</b>	<b>123</b>
<b>A.2 Application Object Tag Values .....</b>	<b>123</b>
<b>APPENDIX B BASELINE HTML SUPPORT .....</b>	<b>126</b>
<b>B.1 Format .....</b>	<b>126</b>
<b>B.2 Supported User Interactions .....</b>	<b>126</b>
<b>B.3 Characters .....</b>	<b>127</b>
<b>APPENDIX C ERROR HANDLING.....</b>	<b>132</b>
<b>APPENDIX D CRC-8 REFERENCE MODEL.....</b>	<b>135</b>
<b>APPENDIX E REVISION HISTORY .....</b>	<b>136</b>

## Figures

Figure 1 - A Multiple Transport Stream Host and CableCARD Device Example Block Diagram .....	1
Figure 2 - M-Host and M-CARD Device Block Diagram Example .....	11
Figure 3 - One Way S-Host and M-CARD Block Diagram Example .....	11
Figure 4 - Block Diagram of M-Host with Two CableCARD Devices and Legacy OOB Communication Path.....	12
Figure 5 - Backward Compatible Mode CableCARD–Host Interface Diagram .....	14
Figure 6 - Multi-Stream CableCARD Mode: CableCARD-Host Interface Diagram .....	15
Figure 7 - CableCARD Device Type Detection Signals .....	21
Figure 8 - CMP Diagram .....	23
Figure 9 - MPEG Transport Stream Pre-Header.....	23
Figure 10 - Serial Interface Protocol Diagram.....	30
Figure 11 - M-CARD Power-On and Reset Timing Diagram .....	35
Figure 12 - Serial Interface Timing Diagram .....	38
Figure 13 - Command Channel Protocol Layers.....	41
Figure 14 - Program Index Table 1 .....	61
Figure 15 - Program Index Table 2 .....	62
Figure 16 - Program Index Table 3 .....	63
Figure 17 - DSG Packet Format Across CableCARD–Host Interface.....	90
Figure 18 - Generic Feature List Exchange .....	93
Figure 19 - CableCARD Device Feature List Change .....	93
Figure 20 - Host Feature List Change.....	93
Figure 21 - Host to CableCARD Device Feature Parameters.....	94
Figure 22 - Host Parameter Update .....	94
Figure 23 - Headend to Host Feature Parameters.....	95
Figure 24 - Specific Application Support Connection Sequence .....	115
Figure 25 - Specific Application Support Alternate Connection Sequence .....	116
Figure 26 - 8 bit CRC generator/checker model .....	135

## Tables

Table 1 - CableCARD-Host Resource Communication .....	2
Table 2 - Example Request From the Host to Decrypt 11 Elementary Streams From 6 Programs Across 3 Transport Streams.....	2
Table 3 - Numerical Representation .....	5
Table 4 - Multi-Stream and Single-Stream CableCARD Connector Pin Assignment.....	16
Table 5 - VPP Pin Configurations, and Associated CableCARD Device Operating Mode.....	21
Table 6 - CPU Interface Packet Format.....	25
Table 7 - M-CARD Power Supply DC Characteristics .....	31
Table 8 - CableCARD Signal Types by Mode.....	32
Table 9 - DC Signal Requirements .....	32
Table 10 - DC Signaling Characteristics for the “LogicPC” Signaling Level .....	33
Table 11 - DC Signaling Characteristics for the “LogicCB” Signaling Level .....	33
Table 12 - M-CARD and M-Host Pullups and Pulldowns.....	34
Table 13 - M-CARD Power-On and Reset Timing Requirements.....	36
Table 14 - M-CARD Power-On and Reset Timing Requirements.....	37
Table 15 - Serial Interface Timing Requirements .....	38
Table 16 - SPDU Structure Syntax .....	42
Table 17 - open_session_request() Syntax .....	43
Table 18 - open_session_response() Syntax .....	43
Table 19 - create_session() Syntax .....	44
Table 20 - create_session_response() Syntax .....	44
Table 21 - close_session_request() Syntax.....	45
Table 22 - close_session_response() Syntax.....	45
Table 23 - session_number() Syntax .....	45
Table 24 - Summary of SPDU Tags .....	46
Table 25 - Public Resource Identifier.....	46
Table 26 - Private Resource Identifier .....	47
Table 27 - resource_identifier() Syntax.....	47
Table 28 - APDU Structure Syntax .....	47
Table 29 - Comparison of Resources Between S-CARD and M-CARD .....	48
Table 30 - Resource Manager Resource Identifier .....	49
Table 31 - Resource Manager APDU List.....	49
Table 32 - profile_inq() APDU Syntax.....	49
Table 33 - profile_reply() APDU Syntax.....	50
Table 34 - profile_changed() APDU Syntax.....	50
Table 35 - Application Information Resource Identifier .....	51
Table 36 - Application Information APDU List.....	51
Table 37 - applicatio_info_req() APDU Syntax .....	51
Table 38 - application_info_cnf() APDU Syntax.....	53
Table 39 - server_query() APDU Syntax.....	54
Table 40 - server_reply() APDU Syntax.....	56
Table 41 - A Low Speed Communication Resource .....	57
Table 42 - CA Support Resource.....	57
Table 43 - CA Support APDUs .....	57
Table 44 - ca_info_inquiry() APDU Syntax .....	58
Table 45 - ca_info() APDU Syntax.....	58
Table 46 - ca_pmt() APDU Syntax.....	59

Table 47 - ca_pmt_reply() APDU Syntax .....	64
Table 48 - ca_update() APDU Syntax .....	65
Table 49 - Host Control Support Resource .....	66
Table 50 - Host Control Support APDUs .....	66
Table 51 - OO_TX_tune_req() APDU Syntax .....	67
Table 52 - OOB Transmit Rate Format .....	67
Table 53 - OOB_TX_tune_cnf() APDU Syntax .....	68
Table 54 - OO_RX_tune_req() APDU Syntax .....	68
Table 55 - OOB Transmit Rate Format .....	69
Table 56 - OO_RX_tune_cnf() APDU Syntax .....	69
Table 57 - inband_tune_req() APDU Syntax .....	70
Table 58 - inband_tune_cnf() APDU Syntax .....	70
Table 59 - System Time Support Resource .....	71
Table 60 - System Time Support APDUs .....	71
Table 61 - Transmission of system_time_inq .....	71
Table 62 - system_time APDU .....	72
Table 63 - MMI Support Resource .....	73
Table 64 - MMI Support APDUs .....	73
Table 65 - open_mmi_req() .....	73
Table 66 - open_mmi_cnf .....	74
Table 67 - close_mmi_req .....	75
Table 68 - close_mmi_cnf .....	76
Table 69 - CableCARD Device Resources Resource .....	76
Table 70 - CableCARD Resources Support APDUs .....	77
Table 71 - stream_profile APDU Syntax .....	77
Table 72 - stream_profile_cnf APDU .....	78
Table 73 - program_profile APDU .....	78
Table 74 - programs_profile_cnf APDU .....	78
Table 75 - es_profile APDU Syntax .....	79
Table 76 - es_profile_cnf APDU Syntax .....	79
Table 77 - request_pids APDU .....	80
Table 78 - request_pids_cnf APDU .....	80
Table 79 - Extended Channel Support Resource .....	82
Table 80 - Extended Channel Support APDUs (Version 1) .....	82
Table 81 - Extended Channel Support APDUs (Version 2) .....	82
Table 82 - Extended Channel Support APDUs (Version 3) .....	82
Table 83 - new_flow_req APDU Syntax .....	84
Table 84 - new_flow_cnf APDU Syntax .....	85
Table 85 - delete_flow_req APDU Syntax .....	86
Table 86 - delete_flow_cnf APDU Syntax .....	87
Table 87 - lost_flow_ind APDU Syntax .....	87
Table 88 - lost_flow_cnf APDU Syntax .....	88
Table 89 - 01-0xFF Reserved inquire_DSG_mode APDU Syntax .....	88
Table 90 - set_DSG_mode APDU Syntax .....	89
Table 91 - inquire_multi_mode APDU Syntax .....	91
Table 92 - set_multi_mode APDU Syntax .....	91
Table 93 - DSG_packet_error APDU Syntax .....	92
Table 94 - Generic Feature Control Resource .....	95
Table 95 - Feature IDs .....	96
Table 96 - Generic Feature Control APDUs .....	96
Table 97 - feature_list_req APDU Syntax .....	97

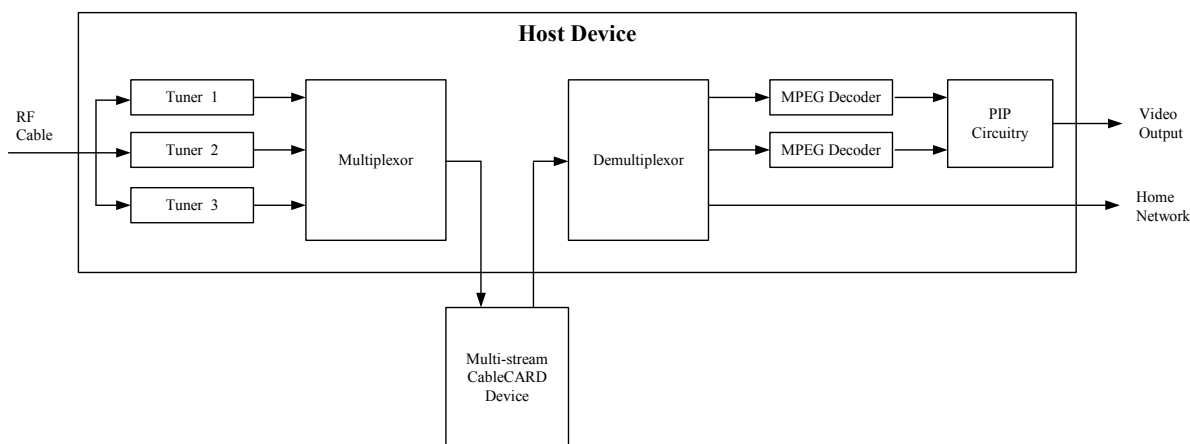
Table 98 - feature_list APDU Syntax .....	97
Table 99 - feature_list_cnf APDU Syntax .....	97
Table 100 - feature_list_changed APDU Syntax .....	98
Table 101 - feature_parameters_req APDU Syntax .....	98
Table 102 - feature_parameters APDU Syntax .....	99
Table 103 - rf_output_channel .....	100
Table 104 - p_c_pin .....	100
Table 105 - p_c_settings .....	100
Table 106 - purchase_pin .....	101
Table 107 - time_zone .....	101
Table 108 - daylight_savings .....	102
Table 109 - ac_outlet .....	102
Table 110 - language .....	102
Table 111 - rating_region .....	102
Table 112 - reset_pin .....	103
Table 113 - cable_urls .....	103
Table 114 - EA_location_code .....	104
Table 115 - feature_parameters_cnf APDU Syntax .....	104
Table 116 - Generic Diagnostic Support Resource .....	105
Table 117 - Generic Diagnostic Support APDUs .....	105
Table 118 - Diagnostic IDs .....	105
Table 119 - diagnostic_req APDU Syntax .....	106
Table 120 - diagnostic_cnf APDU Syntax .....	107
Table 121 - Table Status Field Values .....	108
Table 122 - memory_report .....	108
Table 123 - software_ver_report .....	109
Table 124 - firmware_ver_report .....	110
Table 125 - MAC_address_report .....	110
Table 126 - FAT_status_report .....	111
Table 127 - FDC_status_report .....	112
Table 128 - current_channel_report .....	112
Table 129 - 1394_port_report .....	113
Table 130 - DVI Status Report Syntax .....	114
Table 131 - Aspect Ratio Associated With the Video Format On the DVI Link .....	115
Table 132 - Specific Application Support Resource .....	116
Table 133 - Specific Application Support APDUs .....	116
Table 134 - SAS_connect_rqst APDU Syntax .....	117
Table 135 - SAS_connect APDU Syntax .....	117
Table 136 - SAS_data_rqst APDU Syntax .....	118
Table 137 - SAS_data_av APDU Syntax .....	118
Table 138 - SAS_data_cnf APDU Syntax .....	119
Table 139 - SAS_server_query APDU Syntax .....	119
Table 140 - SAS_server_reply APDU Syntax .....	120
Table 141 - Extended Channel Link Layer Packet .....	121
Table 142 - HTML Keyword List .....	127
Table 143 - Characters .....	128

## 1 INTRODUCTION AND OVERVIEW

The CableCARD-Host interface specification defines the interface between a digital cable receiver or set-top terminal (Host device), and the CableCARD device distributed by the cable operator. The Multi-Stream CableCARD device (M-CARD) is a second generation variant of the original Single-Stream CableCARD device (S-CARD). With the advent of multi-tuner Hosts, Digital Video Recorders (DVR), and other new technologies, multiple transport streams will need to pass through a CableCARD device.

This document defines the M-CARD that can support multiple transport streams. The M-CARD is backward compatible with Single-Stream CableCARDS as defined via [SCTE28], the Host-POD Interface Standard [OC-CC] and the POD Copy Protection System [SCTE41], but adds support for multiple program decryption from multiple transport streams. The typical application is a Host device with multiple tuners and QAM demodulators. The MPEG transport streams from these multiple tuners are temporally multiplexed and sent to the CableCARD device for decryption. In the following diagram, a Host with three tuners supports multiple streams serving a picture-in-picture service for the television, as well as a network connected device, simultaneously.

**Note:** An M-CARD can also be used in devices with only a single tuner.



**Figure 1 - A Multiple Transport Stream Host and CableCARD Device Example Block Diagram**

The multiplexor sends one complete MPEG transport packet at a time across the MPEG interface to the M-CARD. The Host identifies each of the transport stream packets in the multiplex by appending a pre-header to the MPEG-2 transport header. This pre-header contains an 8-bit transport stream ID value, indicating which transport stream the appended packet belongs to.

In order to support multiple streams, the bandwidth of the physical interface between the CableCARD device and the Host device is increased to support up to 200Mbps of aggregate MPEG transport packet data both into and out of the CableCARD device simultaneously. This is sufficient for five simultaneous transport streams from up to five 256-QAM tuners/demodulators, or up to six streams from six 64-QAM tuners/demodulators.

In addition to increasing the number of transport streams that can be transmitted across the interface, this specification also adds capability to the CableCARD-Host command interface to enable the decryption of multiple programs within each transport stream. Therefore, it is possible for the Host to request that multiple programs from each transport stream be decrypted. Through the CA\_PMT structure, which is passed from the Host to the CableCARD device, the Host indicates which programs and which individual elementary streams within each transport stream are to be decrypted by the CableCARD device. Conversely, upon initialization, the CableCARD device indicates to the Host the number of simultaneous programs and simultaneous elementary streams (PIDs) it can decrypt. The Host can, therefore, keep track of how many of each of these resources it has used to determine

how many additional programs and elementary streams it can request to be decoded. These commands are diagrammed in Table 1.

**Table 1 - CableCARD-Host Resource Communication**

Host		CableCARD device
	←	Maximum number of transport streams supported
	←	Maximum number of simultaneous programs supported
	←	Maximum number of simultaneous elementary streams (PIDs) supported
CA_PMT structure(s) (one per program)	→	

An example of the resources that need to be tracked are shown in Table 2, where the Host has requested six programs consisting of two movies, one multi-angle movie, and three music programs, for a total of six programs, eleven elementary streams, and three transport streams. If the CableCARD device had indicated that it could support 4 transport streams, 16 programs, and 32 elementary streams, the Host knows that there are sufficient resources for additional decryption requests. If the CableCARD device only supported simultaneous decrypt of at most six programs, the Host would know not to send any additional program decrypt requests. If the CableCARD device indicated that it could only support three simultaneous streams, the Host could optionally re-multiplex two separate transport streams into one and pass it to the CableCARD device instead, as long as the decryption request remained under the maximum limits indicated by the CableCARD device.

**Table 2 - Example Request From the Host to Decrypt 11 Elementary Streams From 6 Programs Across 3 Transport Streams.**

Transport Layer	Program Layer	Elementary Layer
Transport Stream 1	Program 1	Video ES
		Audio ES
	Program 2	Video ES
		Audio ES
Transport Stream 2	Program 3	Video ES 1
		Video ES 2
		Audio ES 1
		Audio ES 2
Transport Stream 3	Program 4	Audio ES
	Program 5	Audio ES
	Program 6	Audio ES

## 1.1 Scope

This standard defines the characteristics and normative specifications for the interface between the M-CARD and the Host device. M-CARDS are planned to be owned and distributed by the cable operator while the Host MAY be owned by the cable subscriber or by the cable operator. The Host device will be used to access multi-channel

television programming carried on North American cable systems. While analog television channels MAY be tuned, only digital television channels will be passed through the CableCARD device for descrambling of authorized conditional access channels, and passed back to the Host. The CableCARD device will not only provide the conditional access decryption of the digital television channel, but MAY also provide the network interface between the Host and the cable system.

This standard supports a variety of conditional access scrambling systems. Entitlement management messages (EMMs) for such scrambling systems are carried in the cable out-of-band channel as defined by [SCTE55-2], [SCTE55-1], and the DOCSIS® Set-top Gateway Specification [DSG].

The interface will support Emergency Alert messages transmitted over the out-of-band channel to the CableCARD device, which will deliver the message to the Host using the format defined in [J042].

This standard defines, sometimes by reference, the physical interface, signal timing, link interface, and application interface for the CableCARD-Host interface.

## 1.2 Historical Perspective (Informative)

This specification has its origins in [SCTE28]. The original version of the CableCARD device provided only enough bandwidth for a single transport stream. As DVRs, picture-in-picture, and other multiple transport stream features were developed, it was realized that the original single transport stream CableCARD had inadequate bandwidth for some of these features, and could not grow to support multi-tuner gateway scenarios.

A CableCARD device based on this specification provides the higher transport data throughput rates that would be required to support future features, such as multiple-tuner Hosts, Hosts with DVRs, Hosts with picture-in-picture capability, and future extensions of existing conditional access functions to include Digital Rights Management.

## 1.3 Organization of Document

This section provides an overview of the structure of the document.

The major headings include the following:

- Introduction and Overview - provides scope, historical perspective and notation acronym descriptions.
- References - provides a listing of the normative and informative references used by this specification.
- Model of Operation (Informative) - provides background information to describe the high level application of products based on this specification and provides perspective and frame of reference for the normative requirements of this document.
- Physical Interface (Normative)
  - Interface Functional Description - provides the physical requirements of the M-CARD and the signaling interface between the M-CARD and the Host including the following items:
    - Connector Pin Assignment
    - Signal Description of the interface
    - CableCARD Device Type Identification - a description of the steps that take place between the M-CARD being inserted into a Host and initialization of the CableCARD-Host interface.
    - Card Information Structure
    - MPEG Transport Interface Description
    - OOB Interface Description

- CPU Interface Description
- Electrical Specifications
  - Power
  - DC Characteristics
  - AC Characteristics
- Mechanical Specifications
  - Form Factor
  - Connector
  - Environmental Specifications
- Copy Protection
- Command Channel Operation (Normative)
- Extended Channel Operation (Normative)

## 1.4 Requirements (Conformance Notation)

Throughout this document, the words that are used to define the significance of particular requirements are capitalized. These words are:

“SHALL/MUST”	These words or the adjective “REQUIRED” means that the item is an absolute requirement of this specification.
“SHALL NOT/MUST NOT”	This phrase means that the item is an absolute prohibition of this specification.
“SHOULD”	This word or the adjective “SHOULD” means that there may exist valid reasons in particular circumstances to ignore this item, but the full implications should be understood and the case carefully weighed before choosing a different course.
“SHOULD NOT”	This phrase means that there may exist valid reasons in particular circumstances when the listed behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.
“MAY”	This word or the adjective “MAY” means that this item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because it enhances the product, for example; another vendor may omit the same item.

## 1.5 Numerical

All numbers without a prefix are base 10 (decimal). The following prefixes are to be used to designate different bases.

**Table 3 - Numerical Representation**

Prefix	Base	Name
0b	2	Binary
0d	10	Decimal
0x	16	Hexadecimal

## 1.6 Abbreviations and Acronyms

This specification uses the following abbreviations:

<b>AES</b>	Advanced Encryption Standard
<b>ANSI</b>	American National Standards Institute
<b>APDU</b>	Application Protocol Data Unit
<b>API</b>	Application Program Interface
<b>ASCII</b>	American Standard Code for Information Interchange
<b>ATSC</b>	American Television System Committee
<b>CA</b>	Conditional Access
<b>CEA</b>	Consumer Electronic Alliance
<b>CIS</b>	Card Information Structure
<b>CMOS</b>	Complementary Metal Oxide Silicon
<b>CMP</b>	CableCARD MPEG Packet
<b>CPU</b>	Central Processing Unit
<b>CRC</b>	Cyclic Redundancy Check
<b>CVDT</b>	Code Version Download Table
<b>CVT</b>	Code Version Table
<b>DHCP</b>	Dynamic Host Configuration Protocol
<b>DOCSIS®</b>	Data Over Cable Service Interface Specification
<b>DRAM</b>	Dynamic Random Access Memory
<b>DRM</b>	Digital Rights Management
<b>DSG</b>	DOCSIS Set-top Gateway
<b>DSM-CC</b>	Digital Storage Medium – Command and Control
<b>DVB</b>	Digital Video Broadcast
<b>DVR</b>	Digital Video Recorder
<b>DVS</b>	Digital Video Subcommittee
<b>EAS</b>	Emergency Alert System
<b>ECM</b>	Entitlement Control Message
<b>EIA</b>	Electronics Industries Alliance

<b>EIT</b>	Event Information Table
<b>EMM</b>	Entitlement Management Message
<b>ETT</b>	Extended Text Table
<b>FAT</b>	Forward Applications Transport
<b>FCC</b>	Federal Communications Commission
<b>FDC</b>	Forward Data Channel
<b>HTML</b>	HyperText Markup Language
<b>HTTP</b>	HyperText Transport Protocol
<b>I/O</b>	Input/Output
<b>IB</b>	Inband
<b>ID</b>	Identifier/Identity/Identification
<b>IP</b>	Internet Protocol
<b>IPG</b>	Interactive Program Guide
<b>IPPV</b>	Impulse Pay-Per-View
<b>IQB</b>	Interface Query Byte
<b>kHz</b>	Kilohertz
<b>LPDU</b>	Link Protocol Data Unit
<b>LSB</b>	Least Significant Byte or Bit
<b>LTS</b>	Local Time Stamp
<b>LTSID</b>	Local Transport Stream ID
<b>M-CARD</b>	Multi-Stream CableCARD device
<b>M-Host</b>	Multi-Stream Host
<b>MAC</b>	Media Access Control
<b>MGT</b>	Master Guide Table
<b>MHz</b>	Megahertz
<b>MMI</b>	Man-Machine Interface
<b>MPEG</b>	Moving Picture Experts Group
<b>MSB</b>	Most Significant Byte or Bit
<b>MSO</b>	Multiple System Operator
<b>NTSC</b>	National Television Systems Committee
<b>NVM</b>	Non-Volatile Memory
<b>OCAP</b>	OpenCable Application Platform
<b>OOB</b>	Out-of-Band
<b>OUI</b>	Organizationally Unique Identifier
<b>PC</b>	Personal Computer
<b>PCMCIA</b>	Personal Computer Memory Card International Association
<b>PDU</b>	Protocol Data Unit
<b>PHY</b>	Physical Layer
<b>PID</b>	Packet Identifier
<b>PIN</b>	Personal Identification Number
<b>PNG</b>	Portable Network Graphics

---

<b>PPV</b>	Pay-Per-View
<b>PSI</b>	Program Specific Information
<b>PSIP</b>	Program and System Information Protocol
<b>QAM</b>	Quadrature Amplitude Modulation
<b>QPSK</b>	Quadrature Phase Shift Key
<b>RDC</b>	Reverse Data Channel
<b>RF</b>	Radio Frequency
<b>RFC</b>	Request For Comments
<b>ROM</b>	Read Only Memory
<b>RPC</b>	Remote Procedure Call
<b>RX</b>	Receive
<b>S-CARD</b>	Single-Stream CableCARD device compliant to SCTE 28
<b>S-Host</b>	Single-Stream Host device compliant to SCTE 28
<b>SAS</b>	Specific Application Support
<b>SCTE</b>	Society of Cable Telecommunications Engineers
<b>SI</b>	System Information
<b>SRAM</b>	Static Random Access Memory
<b>TPDU</b>	Transport Protocol Data Unit
<b>TSID</b>	Transport Stream ID
<b>TX</b>	Transmit
<b>UDP</b>	User Datagram Protocol
<b>URL</b>	Uniform Resource Locator
<b>VCR</b>	Video Cassette Recorder
<b>VCT</b>	Virtual Channel Table

## 2 REFERENCES

In order to claim compliance with this specification, it is necessary to conform to the following standards and other works as indicated, in addition to the other requirements of this specification. Notwithstanding, intellectual property rights may be required to use or implement such normative references.

### 2.1 Normative References

[DSG]	DOCSIS® Set-top Gateway (DSG) Interface Specification, CM-SP-DSG-I02-040804
[J042]	American National Standard, J-STD-042-2002, Emergency Alert Message for Cable (SCTE 18 and EIA/CEA 814)
[NRSSB]	EIA-679-B Part B, National Renewable Security Standard
[ISO/IEC10646-1]	ISO/IEC 10646-1: 1993 Information technology - Universal Multiple-Octet Coded Character Set (UCS) - Part 1: Architecture and Basic Multilingual Plane.
[ISO/IEC13818-1]	ISO/IEC 13818-1 Generic Coding of Moving Pictures and Associated Audio: Systems
[ISO696-1]	ISO 639-1: 2002 Codes for the representation of names of Languages - Part 1: Alpha-2 code
[ISO639-2]	ISO 639-2: 2002 Codes for the representation of names of Languages - Part 1: Alpha-3 code
[OC-CC]	OpenCable CableCARD™ Interface Specification, OC-SP-CC-IF-I17-040831
[OC-MCCP]	OpenCable Multi-Stream CableCARD Copy Protection System, OC-SP-MCCP-IF-I03-040831
[OC-CD]	OpenCable Common Download Specification, OC-SP-CD-IF-I08-040831
[PCMCIA2]	PCMCIA PC Card Standard Volume 2 Release 8.0, April 2001 Electrical Specification
[PCMCIA3]	PCMCIA PC Card Standard Volume 3 Release 8.0, April 2001 Physical Specification
[RFC2132]	DHCP Options and BOOTP Vendor Extensions, March 1997
[RFC2396]	RFC 2396, August 1998, T. Berners-Lee, R. Fielding, and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax"
[SCTE28]	ANSI/SCTE 28 2003 (formerly DVS 295) Host-POD Interface
[SCTE41]	ANSI/SCTE 41 2003 (formerly DVS 301) POD Copy Protection System
[SCTE55-1]	ANSI/SCTE 55-1 2002 (formerly DVS 178) Digital Broadband Delivery System: Out Of Band Transport Part 1: Mode A
[SCTE55-2]	ANSI/SCTE 55-2 2002 (formerly DVS 167) Digital Broadband Delivery System: Out Of Band Transport Part 2: Mode B
[SCTE65]	ANSI/SCTE 65 2002 (formerly DVS 234) Service Information Delivered Out Of Band

### 2.2 Informative References

[PHILA]	CableLabs POD-Host Interface License Agreement, December 14, 2000
---------	---

## 2.3 Reference Acquisition

- CableLabs Specifications

Cable Television Laboratories, Inc. 400 Centennial Parkway, Louisville, CO 80027;  
Telephone: +1-303-661-9100; Internet: <http://www.cablelabs.com/>

- ISO/IEC Specifications

ISO Central Secretariat: International Organization for Standardization (ISO), 1, rue de Varembe, Case postale 56, CH-1211 Geneva 20, Switzerland; Internet: <http://www.iso.ch/>

- ANSI/SCTE Standards

SCTE - Society of Cable Telecommunications Engineers Inc., 140 Philips Road, Exton, PA 19341;  
Telephone: 610-363-6888 / 800-542-5040; Fax: 610-363-5898; Internet: <http://www.scte.org/>

- ANSI/EIA Standards

American National Standards Institute, Customer Service, 11 West 42<sup>nd</sup> Street, New York, NY 10036;  
Telephone 212-642-4900; Facsimile 212-302-1286; E-mail [sales@ansi.org](mailto:sales@ansi.org); URL <http://www.ansi.org>

- EIA Standards: United States of America

Global Engineering Documents, World Headquarters, 15 Inverness Way East, Englewood, CO USA 80112-5776; Telephone: 800-854-7179; Facsimile: 303-397-2740; E-mail: [global@ihs.com](mailto:global@ihs.com); URL: <http://global.ihs.com>

- Internet Specifications

The Internet Engineering Task Force, IETF Secretariat, c/o Corporation for National Research Initiatives, 1895 Preston White Drive, Suite 100, Reston, VA 20101-5434; Telephone: 703-620-8990; Facsimile: 703-620-9071; E-mail: [ietf-secretariat@ietf.org](mailto:ietf-secretariat@ietf.org); URL: <http://www.ietf.org/rfc>

### 3 MODEL OF OPERATION (INFORMATIVE)

At the subscriber premises, a cable reception system includes a cable navigation device, which is called the Host. Some examples of Host devices are a set-top box, television, VCR, etc. In order to receive scrambled services, the Host uses a security module interface to the CableCARD device. The CableCARD device provides the conditional access operation and the network connectivity for the Host.

Cable system deployments utilize the OOB paths for reception of the EMMs, SI data, EAS, and network connectivity.

#### 3.1 CableCARD Device Functional Description

The following interfaces are required between the Host and the M-CARD:

- Inband MPEG-2 Transport Stream input and output
- A method of receiving OOB data under three different delivery methods (but not simultaneously)
  - SCTE 55-1
  - SCTE 55-2
  - DSG
- A CPU Interface supporting:
  - Command Channel (also called Data Channel)
  - Extended Channel

Additionally, the following is required for interactive Hosts:

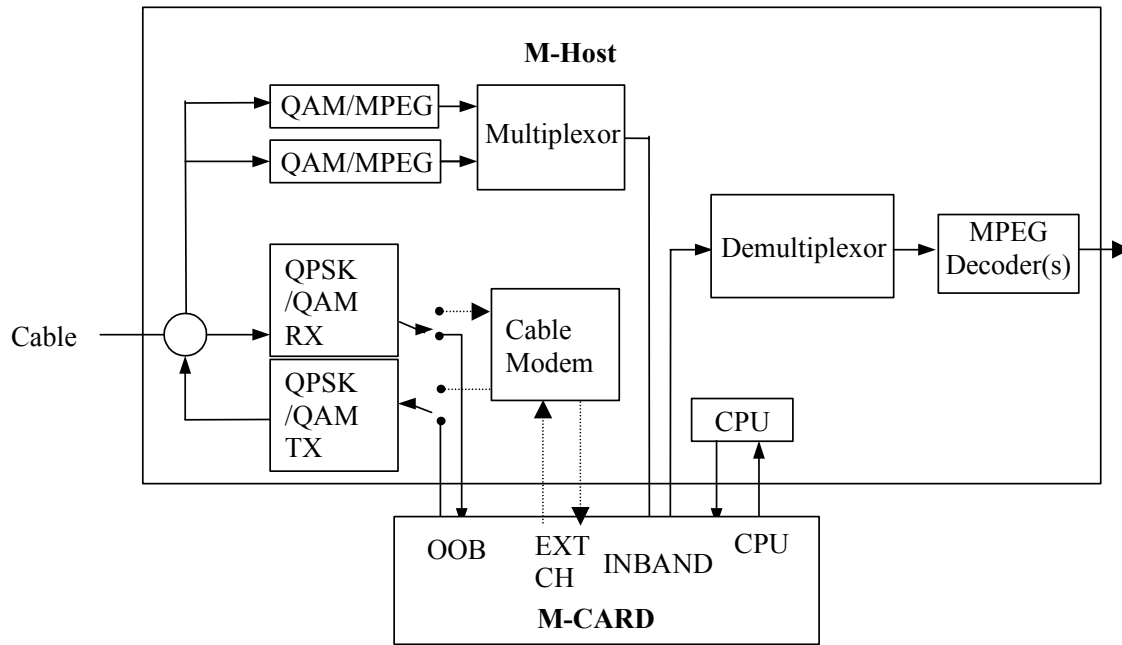
- Network connectivity under three delivery methods (but not simultaneously):
  - SCTE 55-1
  - SCTE 55-2
  - DSG

Copy protection is required for protection of high-valued content across the CableCARD-Host interface.

The M-CARD physical interface will operate in two modes depending on the capabilities of the Host:

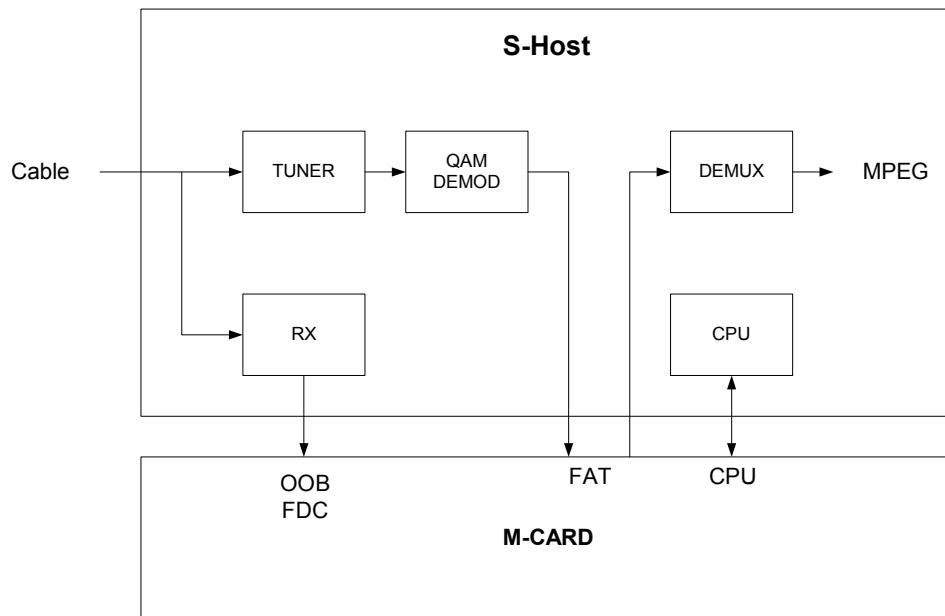
- Single-Stream Backward Compatibility Mode (compliant with [SCTE28])
- Multi-Stream mode (Both single- and multi-stream functionality, compliant with this specification)

The functional elements of a Host and Multi-Stream CableCARD system are shown in the following figure. The Multi-Stream Host (M-Host) shown in the diagram is an example of a possible implementation that includes two tuners and a cable modem.



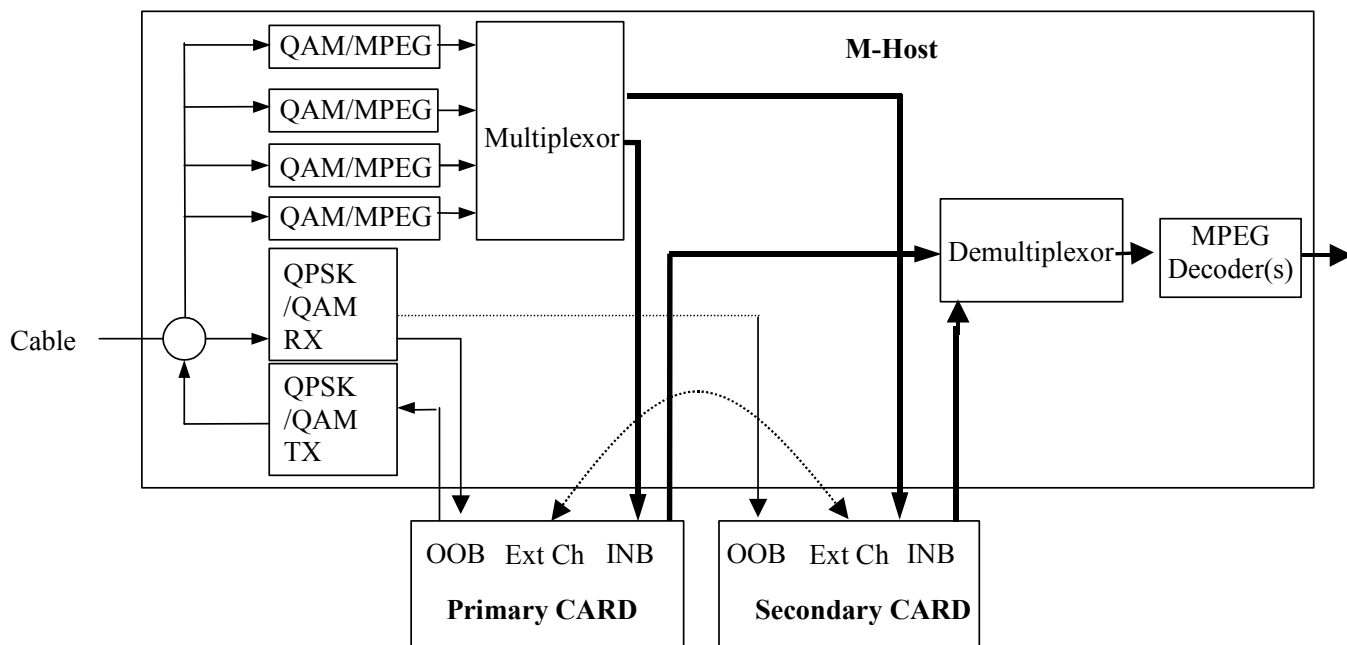
**Figure 2 - M-Host and M-CARD Device Block Diagram Example**

The following figure illustrates an example of an M-CARD interfacing with a One Way Host utilizing the S-CARD interface.



**Figure 3 - One Way S-Host and M-CARD Block Diagram Example**

The following figure illustrates a Host device hosting two CableCARD devices on a plant using a legacy OOB. If the cable plant uses a legacy OOB communication method (non DSG), then only one CableCARD, called the Primary, has access to the reverse transmitter. The two CableCARD devices communicate with one another via the Extended Data Channel.



**Figure 4 - Block Diagram of M-Host with Two CableCARD Devices and Legacy OOB Communication Path**

## 3.2 Physical

The Multi-Stream CableCARD device will use the PCMCIA Cardbus Type II physical form factor. The behavior of the VSS and VPP pins on power-up will be used to distinguish a S-CARD from an M-CARD.

## 3.3 Network Connectivity

One of three types of signaling MAY be utilized, Legacy OOB [SCTE55-2] or [SCTE55-1], and [DSG].

In the legacy OOB modes, the signaling functions are split between the Host and the CableCARD device such that only the RF processing and QPSK demodulation and modulation are done in the Host. The remainder of the processing, including all of the Data-link and MAC protocols, is implemented in the CableCARD device.

In the DSG mode of operation, all of the Data-link and MAC level protocols are implemented in the embedded cable modem in the Host. In this case, the CableCARD device is not responsible for implementing these protocols, since they are provided, via the embedded cable modem. The OOB messaging is transported as follows:

The forward OOB messaging is transported via one or more DSG Tunnels to the Host. The Host filters the IP packets on the DSG Tunnels identified by the Ethernet MAC addresses, specified by the CableCARD device. The Host optionally removes the IP headers of these packets, as instructed by the CableCARD device (the CableCARD device specifies the number of bytes to be removed from the header of the IP packet).

The resulting data packets are serialized into a bit-stream and delivered to the CableCARD device over the CPU/Data interface.

## 3.4 SCTE 28 Backward Compatible Mode

The original Single-Stream CableCARD device was developed to meet the FCC Report and Order regarding security modules. The CableCARD-Host interface was based on the [NRSSB] security module, with the following major modifications:

- Power supply changed from 5V to 3.3V
- OOB connectivity supported
- Copy protection added
- Extended channel added
- Minor signaling changes

S-CARDS and the corresponding compatible single-stream Hosts are built in compliance with [SCTE28].

The M-CARD defined in this specification will function in a single-stream Host built in compliance with [SCTE28]. An M-CARD operating in such a Host will be said to be operating in Backward Compatible Mode.

### **3.4.1 MPEG Data Flow**

In backward compatible mode the MPEG data flow uses two separate 8-bit buses; one for input, and one for output for the MPEG data. All of the Host MPEG data is transmitted through the CableCARD device. The CableCARD device only descrambles the selected program indicated by the Host. If the program is marked as high value content, that program is scrambled across the CableCARD-Host interface.

The CableCARD-Host interface in this mode SHALL be required to support transport stream interface data rates of 26.97035 Mb/s and 38.81070 Mb/s averaged over the period between the sync bytes of successive transport packets with allowable jitter of +/- one MCLKI clock period.

### **3.4.2 OOB Data Flow**

The S-CARD device was modified from the NRSS defined module to include signaling for the SCTE 55-1 and 55-2 OOB methods operation. A later modification added the DSG functionality where the OOB data can be transmitted over the extended channel interface to the CableCARD device.

#### **3.4.2.1 QPSK**

The common modulation method for SCTE 55-1 and SCTE 55-2 is QPSK. This allows the Host to incorporate a common receiver and transmitter for support of the legacy QPSK signaling. The receive signals (data and clock) are passed to the CableCARD device, which performs all the necessary MAC and higher layers of operation. The reverse path transmit signals (I/Q data, enable, and clock) is passed from the CableCARD device to the Host transmitter.

#### **3.4.2.2 DSG**

The DOCSIS Set-top Gateway (DSG) is provided for a Host to be developed, which supports DOCSIS without requiring an additional QPSK receiver. The Host performs all the DOCSIS operations and through the use of Ethernet tunneling, allows for the transmission of the OOB broadcast data to the CableCARD device.

The following diagram shows the connections for the MPEG transport and Out-Of-Band (OOB) data flows under the Backward Compatible Mode:

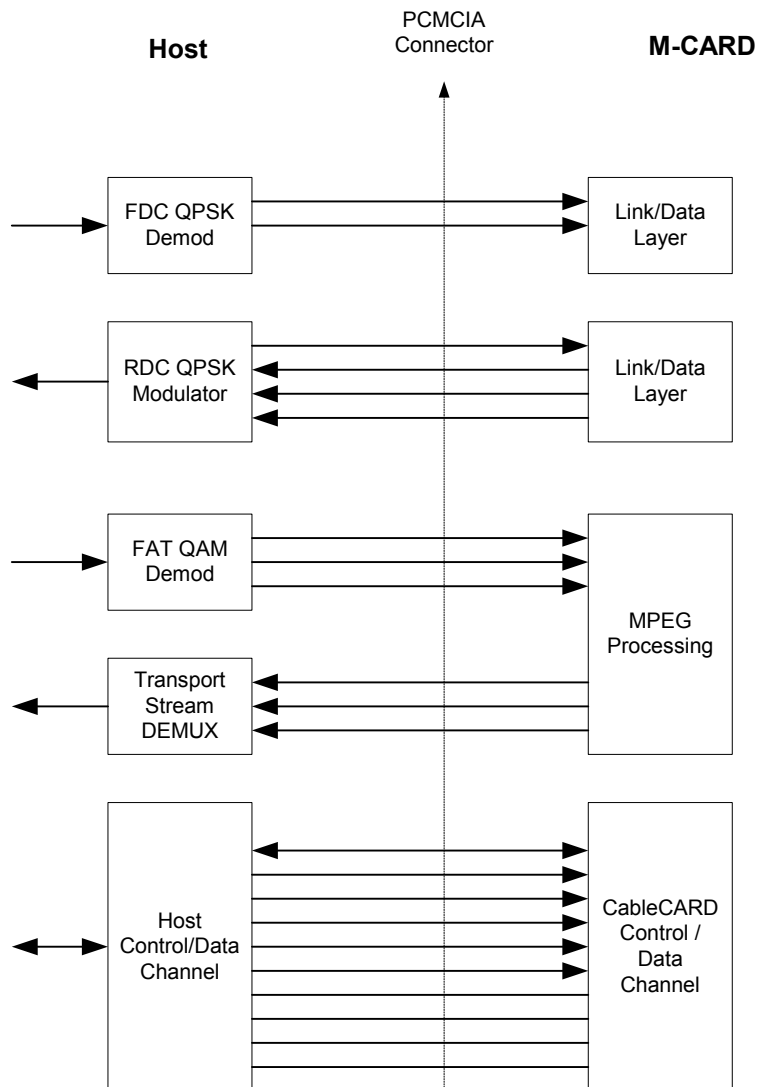


Figure 5 - Backward Compatible Mode CableCARD–Host Interface Diagram

### 3.5 Multi-Stream CableCARD Mode

The M-CARD physical interface is compatible with the Single-Stream CableCARD device. The MPEG data flow has been modified to support multiple streams and new APDUs have been added. The command and control interface is a serial interface in this mode, versus the parallel interface in the backward compatible mode.

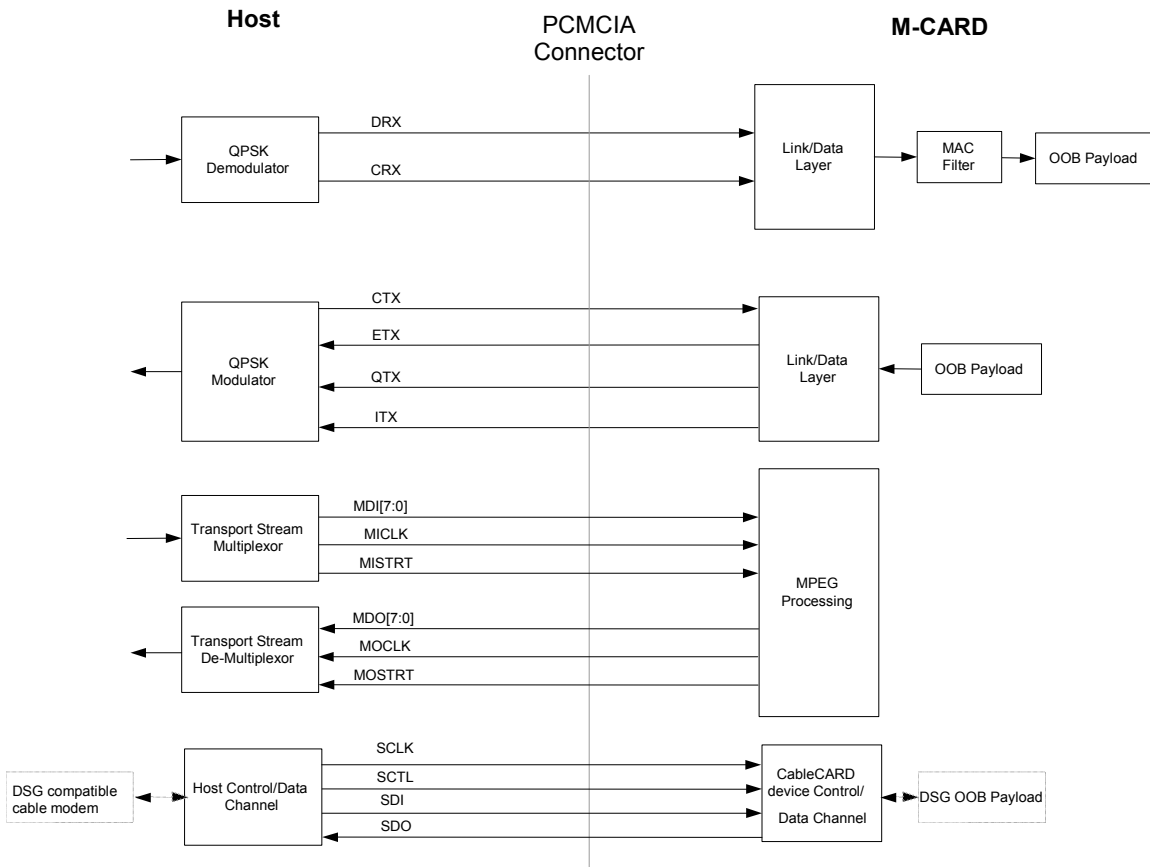
#### 3.5.1 MPEG Data Flow

Packets from multiple transport streams are temporally multiplexed and sent across the parallel MPEG transport interface. In addition, a header is added before each packet for identification. The clock rate of the interface is increased in order to support the increased number of packets.

### 3.5.2 OOB Data Flow

The M-CARD supports all three OOB signaling methods and they operate in the same way as in the backward compatible mode.

The following diagram shows the connections for the MPEG transport and Out-Of-Band (OOB) data flows under the M-CARD Mode:



**Figure 6 - Multi-Stream CableCARD Mode: CableCARD-Host Interface Diagram**

### 3.6 Multiple CableCARD Device Support

The M-CARD, when in multi-stream mode, will support Hosts which allow multiple M-CARDS. In DSG OOB mode, all downstream packets in the DSG tunnel SHALL be delivered to all M-CARDS. The Host SHALL also provide an IP flow to each M-CARD. In legacy OOB modes, the Host will deliver downstream OOB information to all M-CARDS, but only one M-CARD, called the primary M-CARD, will be connected to the legacy OOB reverse transmitter. The other M-CARDS, called the secondary M-CARDS, will send upstream data to/from the primary via the Host utilizing the extended channel. The Extended channel resource has been modified to support this operation.

## 4 PHYSICAL INTERFACE (NORMATIVE)

### 4.1 Interface Functional Description

#### 4.1.1 Connector Pin Assignment

##### 4.1.1.1 Backward Compatible Mode

When operating in the backward compatible single-stream mode, the CableCARD device and Host Pinout will follow that in Section 6.1.3 of [SCTE28].

##### 4.1.1.2 Multi-Stream CableCARD Mode

When operating in multi-stream mode, the M-CARD will have the connector pin assignment as described in Table 4. Note that the PCMCIA memory configuration and [SCTE28] S-CARD pin assignments are included for reference.

**Table 4 - Multi-Stream and Single-Stream CableCARD Connector Pin Assignment**

Pin	PC Card Memory Signals	PC Card Memory I/O <sup>1</sup>	S-CARD Signals	S-CARD I/O*	M-CARD Signals	M-CARD I/O*
1	GND	DC	GND	DC	GND	DC
2	D3	I/O	D3	I/O	Unused	
3	D4	I/O	D4	I/O	Unused	
4	D5	I/O	D5	I/O	Unused	
5	D6	I/O	D6	I/O	Unused	
6	D7	I/O	D7	I/O	Unused	
7	CE1#	I	CE1#	I	Unused	
8	A10	I	Unused		Unused	
9	OE#	I	OE#	I	Unused	
10	A11	I	Unused		Unused	
11	A9	I	DRX	I	DRX	I
12	A8	I	CRX	I	CRX	I
13	A13	I	Unused		MOCLK	O
14	A14	I	MCLKO	O	Unused	
15	WE#	I	WE#	I	Unused	
16	READY	O	IREQ#	O	Unused	
17	VCC	DC in	VCC	DC in	VCC	DC in
18	VPP1	DC in	VPP1	DC in	VPP1	I
19	A16	I	MIVAL	I	Unused	
20	A15	I	MCLKI	I	Unused	
21	A12	I	Unused		MICKL	I
22	A7	I	QTX	O	QTX	O

Pin	PC Card Memory Signals	PC Card Memory I/O <sup>1</sup>	S-CARD Signals	S-CARD I/O <sup>2</sup>	M-CARD Signals	M-CARD I/O <sup>3</sup>
23	A6	I	ETX	O	ETX	O
24	A5	I	ITX	O	ITX	O
25	A4	I	CTX	I	CTX	I
26	A3	I	Unused		Unused	
27	A2	I	Unused		SCTL	I
28	A1	I	A1		SCLK	I
29	A0	I	A0		SDI	I
30	D0	I/O	D0	I/O	Unused	
31	D1	I/O	D1	I/O	Unused	
32	D2	I/O	D2	I/O	Unused	
33	WP	O	IOIS16#	O	MDET	O
34	GND	DC	GND	DC	GND	DC
35	GND	DC	GND	DC	GND	DC
36	CD1#	O	CD1#	O	CD1#	O
37	D11	I/O	MDO3	O	MDO3	O
38	D12	I/O	MDO4	O	MDO4	O
39	D13	I/O	MDO5	O	MDO5	O
40	D14	I/O	MDO6	O	MDO6	O
41	D15	I/O	MDO7	O	MDO7	O
42	CE2#	I	CE2#	I	Unused	
43	VS1#	O	VS1#	O	VS1#	O
44	RFU		IORD#	I	Unused	
45	RFU		IOWR#	I	Unused	
46	A17	I	MISTRTR	I	MISTRTR	I
47	A18	I	MDI0	I	MDI0	I
48	A19	I	MDI1	I	MDI1	I
49	A20	I	MDI2	I	MDI2	I
50	A21	I	MDI3	I	MDI3	I
51	VCC	DC in	VCC	DC in	VCC	DC in
52	VPP2	DC in	VPP2	DC in	VPP2	DC in
53	A22	I	MDI4	I	MDI4	I
54	A23	I	MDI5	I	MDI5	I
55	A24	I	MDI6	I	MDI6	I
56	A25	I	MDI7	I	MDI7	I
57	VS2#	O	VS2#	O	VS2#	I/O
58	RESET	I	RESET	I	RESET	I

Pin	PC Card Memory Signals	PC Card Memory I/O <sup>1</sup>	S-CARD Signals	S-CARD I/O <sup>1</sup>	M-CARD Signals	M-CARD I/O <sup>1</sup>
59	WAIT#	O	WAIT#	O	Unused	
60	RFU		INPACK#	O	SDO	O
61	REG#	I	REG#	I	Unused	
62	BVD2	O	MOVAL	O	Unused	
63	BVD1	O	MOSTRT	O	MOSTRT	O
64	D8	I/O	MDO0	O	MDO0	O
65	D9	I/O	MDO1	O	MDO1	O
66	D10	I/O	MDO2	O	MDO2	O
67	CD2#	O	CD2#	O	CD2#	O
68	GND	DC	GND	DC	GND	DC

\* “I” Indicates the signal is an input to the CableCARD device, “O” indicates the signal is an output from the CableCARD device.

#### 4.1.2 Multi-Stream CableCARD Device Signal Descriptions

##### 4.1.2.1 Power

- VCC** If interfacing to a Host that supports the S-CARD interface, these pins are power pins that initially supply 3.3V per [SCTE28]. If interfacing to a Host using the M-CARD interface, the VCC pins are at a High-Z until CableCARD type identification and discovery is performed. After a M-CARD is detected these pins are powered up to 3.3V.
- GND** As defined in section 2.1.1 of [PCMCIA2].
- VPP1** If interfacing to a Host that supports the S-CARD interface, this pin is a power pin that initially supplies 3.3V and can be switched to 5V per [SCTE28]. If interfacing to a Host using the M-CARD interface, this pin is at a High-Z until CableCARD type identification and discovery is performed. The VPP1 pin is then set to a logic low to indicate that the M-CARD will be interfacing to an M-Host.
- VPP2** If interfacing to a Host that supports the S-CARD interface, this pin is a power pin that initially supplies 3.3V and can be switched to 5V per [SCTE28]. If interfacing to a Host using the M-CARD interface, this pin is at a High-Z until CableCARD type identification and discovery is performed. The VPP2 pin then configured to a 5V supply pin.

##### 4.1.2.2 Sense

- CD2::1#** As defined in section 2.2.1 of [PCMCIA2].
- VS2::1#** As defined in section 2.2.2 of [PCMCIA2]. Only 3.3V will be supported (VS1# = GND, VS2# = High=Z). VS2# is also used during CableCARD device type detection and tied directly to MDET.
- MDET/IOIS16#** M-CARD detect signal used by the Host to identify if CableCARD device is S-CARD or M-CARD. Tied to VS2#.

##### 4.1.2.3 PCMCIA Signals

As defined in Section 4.4 of [PCMCIA2].

#### 4.1.2.4 Single-Stream CableCARD Device Signals

As defined in [SCTE28].

#### 4.1.2.5 Multi-Stream CableCARD Device Signals

<b>DRX</b>	QPSK receive data input to the CableCARD from the S-Host or M-Host
<b>CRX</b>	QPSK receive clock input to the CableCARD from the S-Host or M-Host
<b>ITX</b>	QPSK transmit I-signal output from the CableCARD to the S-Host or M-Host
<b>QTX</b>	QPSK transmit Q-signal output from the CableCARD to the S-Host or M-Host
<b>ETX</b>	QPSK transmit enable output from the CableCARD to the S-Host or M-Host
<b>CTX</b>	QPSK transmit clock input to the CableCARD from the S-Host or M-Host
<b>MCLKI</b>	MPEG transport stream clock from S-Host to the M-CARD used in single-stream mode as defined in [SCTE28].
<b>MICLK</b>	MPEG transport stream clock from M-Host to the M-CARD. The MCLKI signal SHALL always operate at 27 MHz. The MISTRT and MDI[7:0] data signals SHALL be clocked into the M-CARD on the rising edge of MICLK.
<b>MISTRT</b>	MPEG transport stream input packet start indicator from the Host to the CableCARD. Used in single-stream mode as defined in [SCTE28]. Used in multi-stream mode to indicate the start of a CableCARD MPEG Packet (CMP). It is asserted at the same time as the first byte of the CMP header.
<b>MDI[7::0]</b>	An 8 bit wide MPEG transport stream input data bus from the S-Host or M-Host to the M-CARD
<b>MCLKO</b>	MPEG transport stream clock from M-CARD to the S-Host used in single-stream mode as defined in [SCTE28]
<b>MOCLK</b>	MPEG transport stream clock from M-CARD to the M-Host. The MOSTRT and MDO[7::0] signals SHALL be clocked into the Host on the rising edge of MOCLK. This signal SHALL be derived from MCLKI and SHOULD operate at 27 MHz
<b>MOSTRT</b>	MPEG transport stream output packet start indicator from the CableCARD to the Host. Used in single-stream mode as defined in [SCTE28]. Used in multi-stream mode to indicate the start of a CableCARD MPEG Packet (CMP). It is asserted at the same time as the first byte of the CMP header.
<b>MDO[7::0]</b>	An 8 bit MPEG transport stream output data bus from the M-CARD to the S-Host or M-Host.
<b>SCLK</b>	CPU interface serial clock from the M-Host to the M-CARD. The clock is a continuously running clock (not gapped) and has a nominal frequency of 6.75 MHz
<b>SCTL</b>	CPU interface serial interface control signal from the M-Host into the M-CARD. It signals the start of a byte of data and the beginning of a packet being transferred across the interface
<b>SDI</b>	CPU interface serial data from the M-Host into the M-CARD
<b>SDO</b>	CPU interface serial data from the M-CARD out to the M-Host
<b>RESET</b>	Reset input to the M-CARD. RESET is active high or asserted when a logic high

#### 4.1.3 CableCARD Device Type Identification

To allow legacy Hosts (Hosts built to [SCTE28]) to work with Multi-Stream CableCARD devices, the physical interface, the PC Card keying mechanisms and the initialization steps for the legacy Hosts MUST be the same as

those defined in [SCTE28]. Thus, a new unique step is added to the existing initialization steps that allows new Multi-Stream Hosts to reject old CableCARD devices and allows Multi-Stream CableCARD devices to determine what type of Host configuration to support.

#### **4.1.3.1 Backward Compatible Mode**

When the M-CARD is powered up with or inserted into a Host that is operating in the backward compatible single-stream mode, the hot insertion, card keying, card detect mechanism and CPU Interface will follow the identification procedure in Section 6.2 of [SCTE28].

#### **4.1.3.2 Multi-Stream CableCARD Mode**

When the M-CARD is powered up with or inserted into a Host supporting the Multi-Stream CableCARD-Host Interface, the Host MUST determine whether the CableCARD device is a Single-Stream CableCARD device or a Multi-Stream CableCARD device prior to applying power to the CableCARD device. This will allow the Multi-Stream Host to reject the Single Stream CableCARD device prior to applying power. The approach for new Multi-Stream Hosts is the following:

1. The Multi-Stream Host will High-Z the VCC pins until after the CableCARD device type is determined. The VPP1 signal is at a logic low (grounded or pulled down) while VPP2 is at High-Z.
2. Pullup resistors similar to those called out on the PC Card Spec are kept on the Host for CD1#, CD2#, VS1# and VS2#. MDET (WP/IOIS16#) will also have a pullup resistor. These pullup resistors should be tied to the Host 3.3V supply such that if no CableCARD device is installed the pins will be at a logic high.
3. The Host will detect the presence of a CableCARD device by using the CD1# and CD2# pins of the CableCARD device, which are GND.
4. The Host will read the sense of the VS1# pins and the VS2# pins to determine if it could be a CableCARD device (VS1# is GND and VS2# is High-Z) or some other PC Card that is not a CableCARD device. If the state of these is not correct the Host MAY reject the PC Card that has been inserted.
5. A M-CARD then provides a unique identifier mechanism that is not defined under the current PC Card standard. This would be to tie VS2# directly to MDET. The Multi-Stream Host can test this by toggling VS2# and watching to see that MDET tracks VS2#.
6. Once the Multi-Stream Host determines that the CableCARD device is an M-CARD, power is applied to the VCC pins (3.3V) and VPP2 (5.0V). VPP2 is provided to support an M-CARD that contains an optional smart card.
7. The Multi-Stream Host will not be required to support PC Card memory only interface; it can immediately operate in the M-CARD/Host interface.
8. As power is applied to the M-CARD, all interface pins on the M-CARD SHALL be defined in a manner such that it will not contend with the Host until it knows what mode to operate in. The operating mode is determined by the M-CARD detecting the logic levels of the VPP1 and VPP2 pins when RESET is de-asserted. If the VPP1 and VPP2 pins are at a logic high (VCC = 3V), then the CableCARD device SHALL assume it is in legacy single-stream mode. If the VPP1 pin is at a logic low and the VPP2 pin is at logic high (5V), the CableCARD device SHALL initialize to multi-stream mode.
9. If the VPP1/VPP2 configuration is one of the two reserved settings in Table 5, the M-CARD shall not assert the READY line (pin 16) and shall keep SDO (pin 60) low. If one of these signals has not gone active after 5 seconds, the Host may use this as a indication that there is a problem with the M-CARD/Host interface. Note that in S-CARD Hosts, only the READY line has significance, and SDO has no meaning.
10. If a M-Host supports multiple M-CARDS, the “master” M-CARD SHOULD be initialized first.

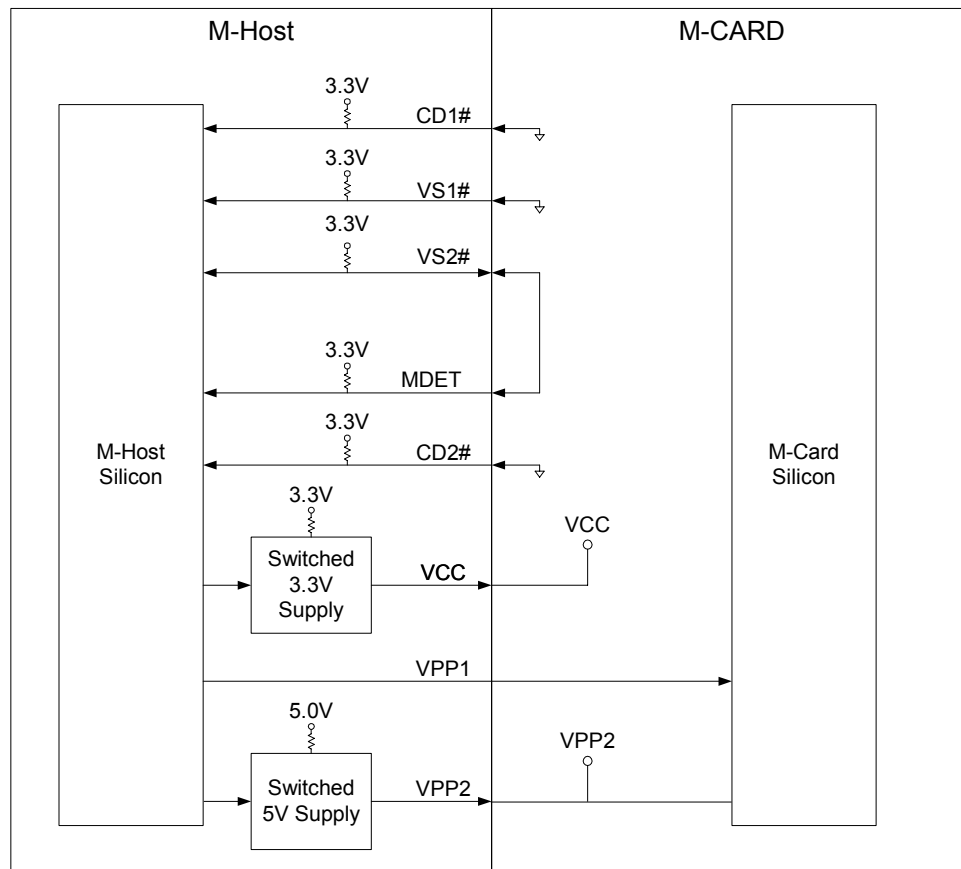
The following table summarizes the VPP pin configurations, and associated CableCARD device operating mode.

**Table 5 - VPP Pin Configurations, and Associated CableCARD Device Operating Mode.**

VPP1	VPP2	CableCARD device Configuration
Low	5V	M-CARD
Low	Low[KB1]	Reserved
High	Low	Reserved
High	High	S-CARD

Note:  
If a M-Host supports multiple M-CARDs, the “master” M-CARD SHOULD be initialized first.

Figure 7 shows the CableCARD device type detection and identification signals for Host operating in multi-stream mode and an M-CARD.



**Figure 7 - CableCARD Device Type Detection Signals**

#### 4.1.4 Card Information Structure

##### 4.1.4.1 Backward Compatible Mode

When operating in the backward compatible single-stream mode, the CableCARD device will provide the CIS table defined in Appendix D of [SCTE28] with the following modification:

Section D.2.6 CCST\_CIF SHALL be modified so that the STCI\_STR SHALL be “CableCARD\_2.00”.

##### 4.1.4.2 Multi-Stream CableCARD Mode

In the Multi-Stream CableCARD Mode of operation, the CIS structure is not read by the Host, and does not need to be presented by the CableCARD device.

#### 4.1.5 MPEG Transport Interface

##### 4.1.5.1 Backward Compatible Mode

In this mode, the M-CARD will implement an MPEG Transport Interface, as defined in [SCTE28].

##### 4.1.5.2 Multi-Stream CableCARD Mode

The combination of the new header and the MPEG packet will be called a CableCARD MPEG Packet (CMP), and will total 200 bytes.

###### 4.1.5.2.1 MPEG Flow – Physical Interface

The MPEG interface SHALL consist of an input clock (MCLKI), an input start of packet signal (MISTRRT), an eight-bit input bus (MDI[7::0]), an output clock (MCLKO), an output start of packet signal (MOSTRT), and an eight-bit output bus (MDO[7::0]). Note that ‘input’ and ‘output’ labels are from the perspective of the M-CARD.

MDI[7::0] data and MISTRRT are clocked into the M-CARD on the rising edge of MICKL. Similarly, MDO[7::0] and MOSTRT are clocked into the Host on the rising edge of MOCLK.

MOCLK SHALL be derived from MICKL. It SHALL NOT have a separate timing source.

The MISTRRT and MOSTRT signals indicate the start of an MPEG packet from the M-CARD to the Host. They are asserted at the same time as the first byte of the CableCARD MPEG Packet header.

###### 4.1.5.2.2 MPEG Flow Model

A CableCARD MPEG Packet (CMP) consists of a 188-byte MPEG packet with a 12 byte header pre-appended to MPEG packet. Each CableCARD MPEG Packet (CMP) SHALL be transmitted with no gaps after the packet start signal is active. The packet must consist of 200 contiguous bytes. In other words, the packets MUST be buffered.

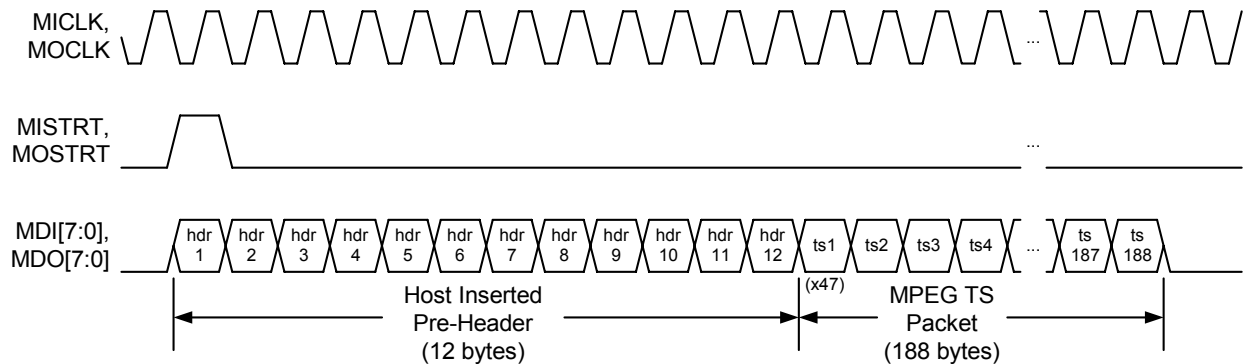
Each packet header and packet SHALL have a fixed delay through the CableCARD device with no more than one MCLK period worth of jitter.

The M-CARD SHALL keep track of the number of bytes received, starting with the packet start signal, and form a packet from the first 200 bytes. During the next byte, if the packet start signal is active, that marks the beginning of the next packet, otherwise, the AMS-POD SHALL discard the sampled data until the next packet start signal is active.

If the M-CARD receives less than 200 bytes prior to MISTRT being asserted, it is not required to pass that packet to the Host and may drop the bytes. However, this SHOULD NOT affect the delay through the M-CARD for any other packets.

The Host SHALL NOT transmit any MPEG packets from different sources with the same transport stream ID. Possible sources include tuners, hard disk drives, or IEEE-1394 interfaces.

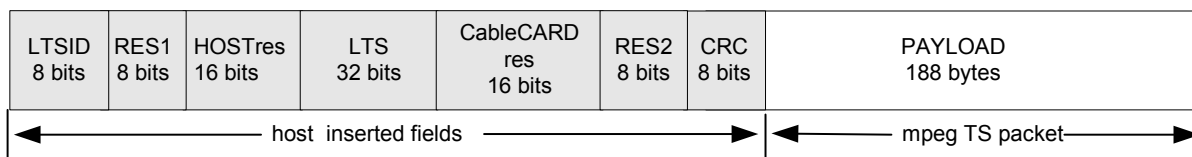
Figure 8 shows an example of a single CMP packet transmitted across the interface.



**Figure 8 - CMP Diagram**

#### 4.1.5.2.3 MPEG Transport Stream Pre-Header

A 12-byte field SHALL be pre-pended by the Host to each MPEG packet sent across the CableCARD-Host interface. This pre-header provides identification information to allow packets from multiple transport streams to be multiplexed to the CableCARD device. The data format is shown in Figure 9. The transport streams are identified by local transport stream ID's (LTSID), which are inserted by the Host. This LTSID is not required to be the same as the QAM transport stream ID. These 8-bit ID's in the transport packet pre-header allow for correlation between the command APDUs and the transport streams.



**Figure 9 - MPEG Transport Stream Pre-Header**

**LTSID** Transport Stream ID – Each packet in a given transport stream SHALL be tagged with the same unique LTSID, in order to allow multiple transport streams to be multiplexed, transferred across the transport interface, correctly decrypted by the CableCARD device, and de-multiplexed, and correctly routed at their destination. The Host is responsible for generating the LTSID value. It is not required to be the same Transport Stream ID as assigned to the QAM transport stream. The CableCARD device SHALL NOT modify it.

**Res1, Res2** Two 8-bit fields reserved for future use. The CableCARD device and Host SHOULD ignore these fields. The default value for both is 0x00.

<b>Host_reserved</b>	A 16-bit field, containing data generated by the Host, identifying additional characteristics of the transport packets. The use of this field is optional for the Host. The M-CARD SHALL NOT modify the values in this field.
<b>LTS</b>	Local Time Stamp – A 32-bit local time stamp, whose value is set by the Host. The CableCARD device SHALL NOT modify this value. The Host MAY use the local time stamp to manage MPEG timing of the packets received from the CableCARD device. Setting a value in this field is optional for the Host.
<b>CableCARD_reserved</b>	A 16-bit field. The default value is 0x00. The usage of this field is optional for the CableCARD device.
<b>CRC</b>	Cyclic Redundancy Check – An 8-bit value calculated and inserted by the Host to provide the ability to check that the LTSID, CableCARD_reserved, Host_reserved, and LTS are transferred across the transport interface without error. If the CableCARD device inserts data into the CableCARD_reserved field, it SHALL recalculate the CRC for those packets. The CRC is defined in Appendix D.

#### 4.1.5.2.4 Multiple Transport Stream Operation and Control

##### 4.1.5.2.4.1 Conditional Access

Each transport stream SHALL be controlled by the ca\_pmt APDU with the associated transport stream id (LTSID). It is the Host's responsibility to manage the PID/program resources in the CableCARD device. The Host receives this information utilizing the profile\_req and profile\_cnf APDUs as defined in section 6.9.5. If the Host performs any filtering of elementary streams, it SHALL utilize the request\_pids, pids\_required, and pids\_required\_cnf APDUs as defined in section 6.9.5.

##### 4.1.5.2.4.2 MMI

There is a transport stream ID associated with the MMI operation so that any MMI dialog is only displayed on the appropriate transport stream.

##### 4.1.5.2.5 Data Throughput

The Host SHALL be responsible for controlling the data through the transport interface such that the data rate does not exceed the interface maximum. The Host MAY do this by removing selected packets from the outgoing transport stream, in order have sufficient bandwidth to multiplex programs from multiple transport streams onto the M-CARD transport interface.

The M-CARD MAY inform the Host which inband PIDs it requires through the CableCARD Capability Discovery resource (Section 6.13). The Host SHALL NOT remove those PIDs under any condition.

Note: The PIDs that the CableCARD device requires MAY change dynamically, therefore the CableCARD device MAY update these PIDs at any time and the Host SHALL respond accordingly to those updates.

##### 4.1.5.2.6 MPEG Packet Jitter

The Host is solely responsible for multiplexing and demultiplexing the transport packets, in a manner to minimize jitter of the MPEG PCR. However, packets sent across the interface from the Host to the CableCARD device SHOULD be returned to the Host with a uniform, fixed delay from receipt to transmission, back to the Host. The fixed delay MAY vary by up to, but not exceed, one MCLKI period (MCLKI is the MPEG transport interface input clock i.e.,  $\pm 1$  MCLK period).

All transport stream packets sent across the interface will be returned in the same order in which they are received by the CableCARD device.

#### 4.1.6 CableCARD-Host OOB Interface

The M-CARD out-of-band interface SHALL be as defined in Section 6.4 of [SCTE28].

The M-CARD interface will support DSG, SCTE 55-1 and SCTE 55-2 out-of-band methods defined in [DSG] [SCTE55-1] and [SCTE55-2] respectively. Only one method will be active at a time.

For the DSG operation, the Host SHALL support the DOCSIS PHY and MAC layers. Operation will follow the DSG specification defined in [DSG].

#### 4.1.7 CPU Interface

##### 4.1.7.1 Backward Compatible Mode

When operating in the backward compatible single-stream mode, the CPU Interface SHALL comply with Section 6.5 of [SCTE28].

##### 4.1.7.2 Multi-Stream CableCARD Mode

The M-CARD CPU interface consists of two logical channels, the data (or command) channel and the extended channel. The command channel is typically used for command and control transactions, while the extended channel is typically used for data transfers (SI, EAS, IP, etc.).

##### 4.1.7.3 Physical Interface

The physical interface for the CPU interface is a modified SPI (Serial Peripheral Interface). Since the only connection is between the Host and the M-CARD, the phase is fixed with the data changing on the falling edge of the clock (SCLK) and clocked in on the rising edge. A control signal (SCTL) is utilized to signal the start of a byte of data as well as the start of a new packet. There are separate data signals: M-Host to CableCARD data (SDI), and CableCARD to M-Host (SDO).

##### 4.1.7.3.1 Packet Format

When the start of a packet occurs, the first byte is defined to be the interface query byte, which includes the interface flags defined below.

After the interface query byte, the packet count consists of two bytes, which contain the number of data bytes following in the packet. In other words, the ‘length’ does not include the first three bytes of the packet. The MSB of the packet count is transmitted first. The maximum number of data bytes in a packet is 4,096. Therefore the three most significant bits of the 16-bit length should always be zero.

**Table 6 - CPU Interface Packet Format**

	Bit	7	6	5	4	3	2	1	0
Host	Query	X	HR	EC	L	F	DA	CR	X
	Length MSB	b7	b6	b5	b4	b3	b2	b1	b0
	Length LSB	b7	b6	b5	b4	b3	b2	b1	b0
	Data Byte(s)	b7	b6	b5	b4	b3	b2	b1	b0
M-CARD	Query	X	CR	EC	L	F	DA	ER	X

	Bit	7	6	5	4	3	2	1	0
	Length MSB	b7	b6	b5	b4	b3	b2	b1	b0
	Length LSB	b7	b6	b5	b4	b3	b2	b1	b0
	Data Bytes(s)	b7	b6	b5	b4	b3	b2	b1	b0

After the packet count, if the DA bit was set in the interface query byte, either the command channel or extended channel data SHALL follow.

#### 4.1.7.3.2 Interface Flags

- HR**      **Host Ready:** The Host SHALL set this flag when it is ready to transmit or receive data. The M-CARD SHALL monitor this flag and not set the CR flag until this flag is set. This flag allows the Host to control the throughput of packets from the CableCARD device.
- CR**      **CableCARD Ready:** The M-CARD SHALL set this flag when it is ready to transmit or receive data. After the RESET signal goes inactive, this signal will indicate to the Host when the M-CARD is ready. The M-CARD SHALL set this flag less than 5 seconds after RESET goes inactive. This flag also allows the M-CARD to control the throughput of packets from the Host.
- EC**      **Extended Channel:** 0 = Command Channel, 1 = Extended Channel. The Host or M-CARD SHALL use this flag to determine whether the data transmitted from the source is intended for the Command or Extended channels. It SHOULD be noted that it is possible for the Host to transmit Command Channel data while the M-CARD is transmitting Extended Channel data or vice versa.
- L**        **Last:** Indicates to the Host/M-CARD that the packet is the last one. Transactions are segmented into multiple packets only when the data size is greater than 4,096 bytes. Transactions that contain less than 4,096 data bytes SHALL set this flag.
- F**        **First:** Indicates to the Hosts/M-CARD that the packet is the first one. Transactions are segmented into multiple packets only when the data size is greater than 4,096 bytes. Transactions that contain less than 4,096 data bytes SHALL set this flag.
- DA**      **Data Available:** Indicates to the Host/M-CARD that data is available. When the Host detects that the M-CARD has set this flag, it SHALL read the following length bytes from the M-CARD and SHALL clock all of the remaining data in prior to indicating a start of message. The M-CARD MAY choose to ignore the Host's DA flag.
- ER**      **Error detected:** The Host or M-CARD has detected an error in the CPU interface. If the Host detects that the M-CARD has set this flag, it SHALL reset the M-CARD. The M-CARD MAY choose to ignore the Host's ER flag.

#### 4.1.7.3.3 Interface Model

The Host is the master of this interface. The Host SHALL always transmit the interface query byte (IQB), even when it does not have data to transmit. If the CableCARD device has data (DA = 1), then the Host is responsible for clocking and receiving the entire packet length and inputting all bytes as defined in the packet count. The intention is that the Host SHALL transmit the IQB immediately following the last byte of the previous packet byte, even if the previous packet did not include any data. This allows for a high-speed interface for both Host and M-CARD sourced data without resorting to separate interrupt signals

The Host shall repeatedly send the IQB followed by 2 bytes of packet count until a message transfer begins. If the Host does not have any data to send, the packet count following the IQB SHALL be set to 0.

The Host and CableCARD both SHALL have separate read and write buffers that are 4,096 bytes, excluding the interface query byte and packet count.

Each packet SHALL only be either a command or extended type, i.e., no mixing of types.

For the command channel, only one SPDU SHALL be allowed per packet.

For the extended channel, only one flow ID SHALL be allowed per packet.

If a packet is not segmented, then both the F and L bits SHALL be set.

Packets smaller than 4,096 bytes SHALL NOT be segmented.

If a packet is larger than 4,096 bytes, then it SHALL be segmented into contiguous packets. The F bit SHALL be set for the first packet. The L bit SHALL be set for the last packet.

The ready flags (HR & CR) SHALL be used for flow control.

The interface is assumed to be a reliable interface. Any error condition detected by the CableCARD device (i.e., under-run) SHALL cause the ER bit to be set. This SHOULD be considered to be a catastrophic failure during normal operation causing the Host to reset the M-CARD. However, during development there MAY be a need to not reset the interface so in the event that an error is detected, the M-CARD SHALL ignore the incorrect packet but proceed with normal operations. It is the Host's responsibility to perform the reset operation. There is an ER bit for the Host to set. However, the M-CARD SHOULD ignore this bit.

#### 4.1.7.3.4 Operation

The following is an example of operation of the CPU interface at initialization through opening the resource manager session. For the purpose of this example, it is assumed that the M-CARD and Host contains four buffers, one each for transmission and receiving for both channels, each buffer is 4,096 bytes. All flags are assumed to be initialized to zero and will remain unchanged unless specifically called out below. In the table, an X is defined to be "don't care" for the receiving device.

1. The Host brings RESET inactive. The Host SHALL bring its HR flag active within 1 second. It SHOULD be noted that it is expected for the Host to be fully operational prior to bringing RESET inactive.

	Bit	7	6	5	4	3	2	1	0
Host	Query	X	HR = 1	EC = X	L = X	F = X	DA = 0	ER = 0	X
	Length MSB	0	0	0	0	0	0	0	0
	Length LSB	0	0	0	0	0	0	0	0
M-CARD	Query	X	CR = 0	EC = X	L = X	F = X	DA = 0	ER = 0	X
	Length MSB	X	X	X	X	X	X	X	X
	Length LSB	X	X	X	X	X	X	X	X

2. The M-CARD brings its CR flag active within 5 seconds.

	Bit	7	6	5	4	3	2	1	0
Host	Query	X	HR = 1	EC = X	L = X	F = X	DA = 0	ER = 0	X
	Length MSB	0	0	0	0	0	0	0	0
	Length LSB	0	0	0	0	0	0	0	0
M-CARD	Query	X	CR = 1	EC = X	L = X	F = X	DA = 0	ER = 0	X
	Length MSB	X	X	X	X	X	X	X	X
	Length LSB	X	X	X	X	X	X	X	X

3. The M-CARD loads its open\_session\_request SPDU to open a session to the Host's resource manager resource into its command channel output buffer.
4. The M-CARD sets its DA flag and clears its EC flag on the next Host query. Since there are 6 bytes in the open\_session\_request, it will set the length to 0x0006. The Host will read the 2<sup>nd</sup> and 3<sup>rd</sup> bytes of data from the M-CARD. The Host SHALL then clock in 6 data bytes from the M-CARD.

	Bit	7	6	5	4	3	2	1	0
Host	Query	X	HR = 1	EC = X	L = X	F = X	DA = 0	ER = 0	X
	Length MSB	0	0	0	0	0	0	0	0
	Length LSB	0	0	0	0	0	0	0	0
M-CARD	Query	X	CR = 1	EC = 0	L = 1	F = 1	DA = 1	ER = 0	X
	Length MSB	0	0	0	0	0	0	0	0
	Length LSB	0	0	0	0	0	1	1	0

5. The Host reads the value in its command channel input buffer. The Host then responds by putting its open\_session\_response SPDU into its command channel output buffer. If the time to process this is significant, then the Host MAY clear its HR flag until it is ready to send/receive more data.
6. The Host then sets its DA flag and HR flag, and clears its EC flag on the next Host query. The 2<sup>nd</sup> and 3<sup>rd</sup> bytes contain the length of the data, which will be 0x0009 for the open\_session\_response SPDU. The Host will then clock out all data in its command channel output buffer. The M-CARD will clock this data into its command channel receive buffer.

	<b>Bit</b>	<b>7</b>	<b>6</b>	<b>5</b>	<b>4</b>	<b>3</b>	<b>2</b>	<b>1</b>	<b>0</b>
Host	Query	X	HR = 1	EC = 0	L = 1	F = 1	DA = 1	ER = 0	X
	Length MSB	0	0	0	0	0	0	0	0
	Length LSB	0	0	0	0	1	0	0	1
M-CARD	Query	X	CR = 1	EC = X	L = X	F = X	DA = 0	ER = 0	X
	Length MSB	0	0	0	0	0	0	0	0
	Length LSB	0	0	0	0	0	0	0	0

7. The M-CARD then clears its ready flag, CR, while it processes the data in its buffer.
8. Now that the resource manager session is open, the M-CARD will now load the profile\_inq APDU into its command channel output buffer and then bring the CR and DA flags active. From here, it will continue with the initialization procedure.

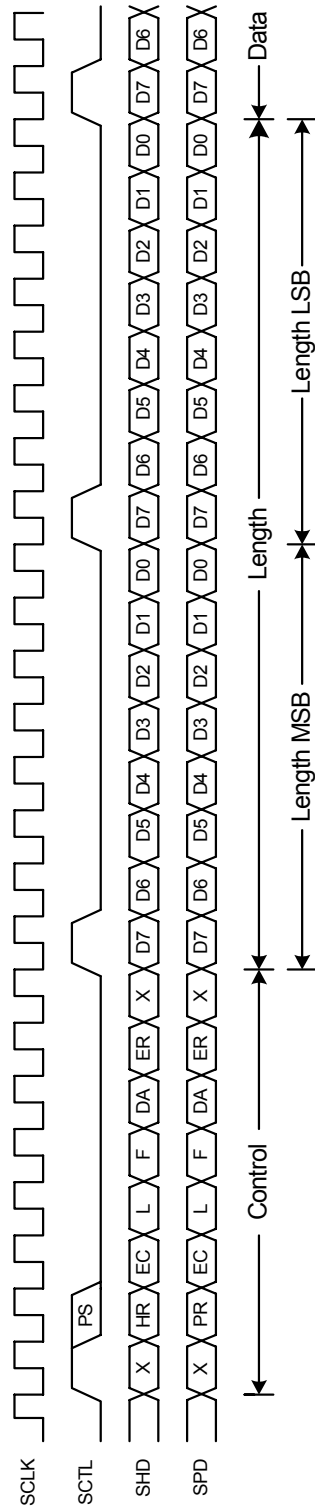


Figure 10 - Serial Interface Protocol Diagram

## 4.2 Electrical Specifications

### 4.2.1 DC Characteristics

#### 4.2.1.1 Power

##### 4.2.1.1.1 Backward Compatible Mode

The M-CARD SHALL comply to Section 6.1.2 the Power Management section of [SCTE28].

##### 4.2.1.1.2 Multi-Stream CableCARD Mode

The Host SHALL provide 3.3 V on VCC and 5V on VPP2 with the DC Characteristics as defined in Table 7. The Host SHALL apply 3.3 V on VCC and 5V on VPP2 as defined in section 4.1.3.2.

The Host SHALL be capable of providing up to 1 Amp total on the VCC pins (500 ma each) per M-CARD. The Host SHALL be capable of providing up to 125 ma on the VPP2 pin. There is no standby power mode for the M-CARD. The M-CARD SHALL NOT draw more than 2.5 watts averaged over a period of 10 seconds.

The Host OOB circuitry MUST continue to operate in all powering states of the Host.

**Table 7 - M-CARD Power Supply DC Characteristics**

Symbol	Parameter	Condition	Min	Max	Units	Notes
VCC	Supply Voltage		3.0	3.6	V	1
I <sub>CC</sub>	Supply Current		-	1.0	A	2
VPP2	Supply Voltage		4.75	5.25	V	1
I <sub>PP2</sub>	Supply Current		0	0.125	A	2
Notes:						
1. There is no standby power mode for the M-CARD						
2. The Host SHALL be capable of providing up to 1 Amp total on the VCC pins (500 ma each) per M-CARD.						

#### 4.2.1.2 Signaling Interfaces

##### 4.2.1.2.1 Backward Compatible Mode

The DC characteristics of the interface signals in S-CARD mode are defined in [SCTE28] with the exception of MICKL (A12 pin 21) which will have the characteristics of CCLK.

##### 4.2.1.2.2 M-CARD Mode

Table 8 shows the signal type for each of the interface signals when used in M-CARD mode.

**Table 8 - CableCARD Signal Types by Mode**

PC Card Memory Only Signals	S-CARD Mode Signals	M-CARD Mode Signals	Signal Type
A13, A12	Unused	MOCLK, MICKL	CCLK
A[25:17]	MDI[7:0], MISTR	MDI[7:0], MISTR	LogicCB
D[15:8], BVD1	MDO[7:0], MOSTRT	MDO[7:0], MOSTRT	LogicCB
A[9:4]	DRX, CRX, CTX, ITX, QTX, ETX	DRX, CRX, CTX, ITX, QTX, ETX	LogicPC
A[2:0],RFU	Unused, A[1:0], INPACK#	SCTL, SCLK, SDI, SDO,	LogicPC
VS1#, VS2#	VS1#, VS2#	VS1#, VS2#	Sense <sup>1</sup>
RESET	RESET	RESET	LogicPC
WP	IOIS16#	MDET	LogicPC <sup>2</sup>
VPP1, VPP2	VPP1, VPP2	VPP1, VPP2	VPP1, VPP2
CD[2:1]#	CD[2:1]#	CD[2:1]#	Sense
<p>Note:</p> <p><sup>1</sup>VS2# is also tied directly to MDET, thus MDET does not need to be sourced or driven from the CableCARD device but sourced or driven by the Host via VS2#.</p> <p><sup>2</sup>The LogicPC signal type applies to VPP1 and VPP2 only during CableCARD device type identification as described in section 4.1.3.2. Otherwise, VPP1 and VPP2 have the supply characteristics described in section 4.2.1.1.</p>			

Table 9 shows the DC characteristics of the signal interfaces for the M-CARD.

**Table 9 - DC Signal Requirements**

Signal Type	DC Signal Requirements
CCLK	DC Signal levels as defined in Table 5-7 DC Specifications for 3.3V Signaling in Section 5.3.2.1.1 of [PCMCIA2] Release 8.0
LogicCB	DC Signal levels as defined in Table 5-7 DC Specifications for 3.3V Signaling in Section 5.3.2.1.1 of [PCMCIA2] Release 8.0
LogicPC	DC Signal levels as defined in Table 4-15 DC Specifications for 3.3V Signaling in Section 4.7.1 of [PCMCIA2] Release 8.0.
Sense	Sense- Sense signals as defined in [PCMCIA2].

Table 10 is the DC signaling characteristics for the “LogicPC” signaling level.

**Table 10 - DC Signaling Characteristics for the “LogicPC” Signaling Level**

Symbol	Parameter	Condition	Min	Max	Units	Notes
VCC	Supply Voltage		3.0	3.6	V	
V <sub>IH</sub>			2.0	VCC + 0.3	V	
V <sub>IL</sub>			-0.3	0.8	V	
V <sub>OH</sub>			2.4 (VCC-0.2) <sup>1</sup>		V	
V <sub>OL</sub>				0.4 (0.2) <sup>1</sup>	V	
Note: All logic levels per JEDEC 8-1B. This table is for reference only. <sup>1</sup> For CMOS Loads						

Table 11 is the DC signaling characteristics for the “LogicCB” and “CCLK” signaling level.

**Table 11 - DC Signaling Characteristics for the “LogicCB” Signaling Level**

Symbol	Parameter	Condition	Min	Max	Units	Notes
VCC	Supply Voltage		3.0	3.6	V	
V <sub>IH</sub>			0.475V <sub>C</sub>	VCC + 0.5	V	
V <sub>IL</sub>			-0.5	0.325VCC	V	
I <sub>IL</sub>		0 < V <sub>in</sub> < VCC		+/-10	uA	
V <sub>OH</sub>		I <sub>out</sub> = -150uA	0.9VCC		V	
V <sub>OL</sub>		I <sub>out</sub> = 700uA		0.1VCC	V	
C <sub>card</sub>	Card Input Pin Capacitance		5	17	pF	
C <sub>host</sub>	System Load Capacitance		5	22	pF	
Note: All logic levels per [PCMCIA2]. This table is for reference only.						

Table 12 is the requirements for Pullups and Pulldowns for multi-stream mode Operation.

**Table 12 - M-CARD and M-Host Pullups and Pulldowns**

Item	Signal	M-CARD	M-Host	Notes
Card Detect	CD[2:1]#		pullup to Host 3.3V R.>= 10K	
Voltage Sense	VS[2:1]#		pullup to Host 3.3V 10K <= R.<= 100K	
Card Type Detect	MDET		pullup to Host 3.3V R >= 100K	
Control Signal	RESET	pullup to VCC R >= 100K		3
MPEG Interface	MICLK, MISTRT, MDI[7:0]	pulldown R >= 100K		1
	MOCLK	pulldown R >= 100K		1
	MOSTRT		pullup	1
	MDO[7:0]	pulldown R >= 100K		1
OOB Interface	CRX, DRX, CTX	pulldown R >= 100K		1, 5
	ITX, QTX, ETX	pulldown R >= 100K		1, 4
CPU Interface	SCTL, SCLK, SDI	pulldown R >= 100K		1, 5
	SDO			2, 4
Notes:				
<ol style="list-style-type: none"> <li>1. Due to PC Card Requirement for pullups and pulldowns on the Memory Interface</li> <li>2. S-Host has R &gt;= 10K pullup required for INPACK#. The M-Host is responsible for providing signal conditioning (i.e., a pullup) in a manner that meets the DC and AC requirements of this signal</li> <li>3. Note 3 of Table 4-16 of in Section 4.7.1 of [PCMCIA2] Release 8.0 applies.</li> <li>4. Note 4 of Table 4-16 of in Section 4.7.1 of [PCMCIA2] Release 8.0 applies.</li> <li>5. Note 5 of Table 4-16 of in Section 4.7.1 of [PCMCIA2] Release 8.0 applies.</li> </ol>				

## 4.2.2 AC Characteristics

### 4.2.2.1 Backward Compatible Mode

The AC characteristics of the M-CARD in this mode are defined in [SCTE28].

### 4.2.2.2 M-CARD Mode

#### 4.2.2.2.1 AC Signaling Characteristics

Table 1 above shows the signal type for each of the interface signals when used in M-CARD mode. The AC signaling characteristics of CCLK, LogicPC and Logic CB are described below.

Pin type CCLK has AC Signal levels as defined in Table 5-9 AC Specifications for 3.3V Signaling (CCLK) in Section 5.3.2.1.4 of [PCMCIA2] Release 8.0.

Pin type LogicPC - DC Signal levels as defined in Table 4-15 DC Specifications for 3.3V Signaling in Section 4.7.1 of [PCMCIA2] Release 8.0.

Pin type LogicCB- AC Signal levels as defined in Table 5-8 AC Specifications for 3.3V Signaling in Section 5.3.2.1.2 of [PCMCIA2] Release 8.0.

#### 4.2.2.2.2 Power and Reset Timing

The power-on sequence timing is similar to the 16 bit PC card standard, with the exception that the MCLKI and the SCLK are running prior to RESET being released and VPP2 will be 5V instead of 3.3V. Figure 11 and Table 13 show the power up timing requirements.

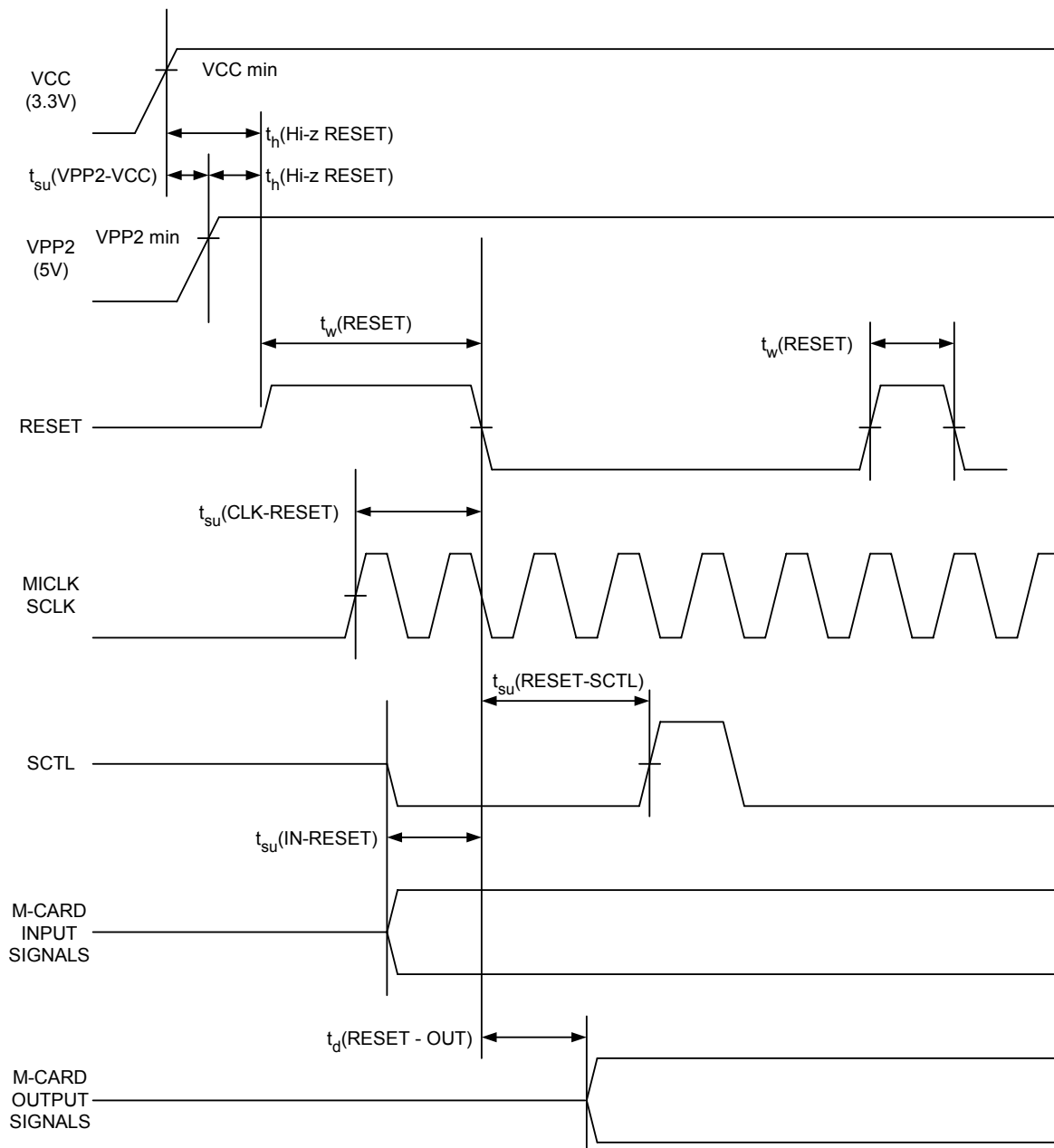


Figure 11 - M-CARD Power-On and Reset Timing Diagram

**Table 13 - M-CARD Power-On and Reset Timing Requirements**

Symbol	Parameter	Condition	Min	Max	Units	Notes
$t_{su}(VPP2-VCC)$	VCC valid to VPP2 valid		50		us	1
$t_h(\text{Hi-z RESET})$	VCC and VPP2 valid to RESET assert		1		ms	
$t_w(\text{RESET})$	Reset pulse width		10		us	
$t_{su}(\text{CLK-RESET})$	Clock valid to RESET negate		0		ms	
$t_{su}(\text{RESET-SCTL})$	Reset Negate to CPU Interface Active		20		ms	2
$t_{su}(\text{IN-RESET})$	Reset Negate to CPU Interface Active		0		ms	2
$t_d(\text{RESET-OUT})$	Reset Negate to M-CARD Output Signals Valid		0	20	ms	3

## Notes:

1. VPP2 MUST NOT exceed VCC=0.3V until VCC has reached VCC min as defined in Table 7 M-CARD Power Supply DC Characteristics.
2. The M-CARD input logic signals are Hi-z from when VCC and VPP2 are applied until RESET is asserted.
3. The M-CARD output logic signals are Hi-z from when VCC and VPP2 are applied until RESET is de-asserted/negated.

#### 4.2.2.2.3 MPEG Transport Timing

The setup and hold times of the MPEG transport interfaces will be similar to the Cardbus data to CCLK Timing Parameters.

**Table 14 - M-CARD Power-On and Reset Timing Requirements**

Symbol	Parameter	Condition	Min	Max	Units	Notes
$t_{cyc}$	MICLK and MOCLK Cycle Time		37.00	37.074	ns	1,2,3
$t_{high}$	MICLK and MOCLK High Time		15		ns	3
$t_{low}$	MICLK and MOCLK Low Time		15		ns	3
$t_{su}$	Input Setup time of MDI[7:0] and MISTRT to MICLK and Input Setup Time of MDO[7:0] and MOSTRT to MOCLK		7		ns	4
$t_h$	Input Hold time of MICLK to MDI[7:0] and MISTRT and Input Hold Time of MOCLK to MDO[7:0] and MOSTRT		0		ns	4
$t_{val}$	MICLK to and MOCLK to Output Signal Valid Delay.		2	18	ns	4
Notes:						
1. MOCLK will be derived directly from MICLK.						
2. The Nominal Frequency is 27MHz. As specified the M-CARD will be required to operate at 27MHz +/- 1000ppm although the M-Host may be required to operate at tighter tolerances to maintain MPEG timing.						
3. See Figure 5-28 CardBus PC Card Clock Waveform of [PCMCIA2] Release 8 for reference levels.						
See Figure 5-30 Input Timing Measurement Conditions and Table 5-12 Measurement and Test Condition Parameters of [PCMCIA2] Release 8 for reference levels.						

#### 4.2.2.2.4 OOB FDC and RDC Timing

The AC Timing characteristics of the OOB FDC and RDC timing in M-CARD mode is the same as defined in [SCTE28].

#### 4.2.2.2.5 CPU Interface Timing

##### 4.2.2.2.5.1 SCLK

The clock SHALL operate at a nominal rate of 6.75 MHz (27 MHz / 4).

##### 4.2.2.2.5.2 SCTL

The serial control signal SHALL change on the falling edge of the SCLK signal.

4.2.2.2.5.3 SDI

The serial Host data SHALL change on the falling edge of the SCLK signal. The CableCARD device SHALL input the data on the rising edge of the SCLK signal.

4.2.2.2.5.4 SDO

The serial CableCARD data SHALL change on the falling edge of the SCLK signal. The Host SHALL input the data on the rising edge of the SCLK signal.

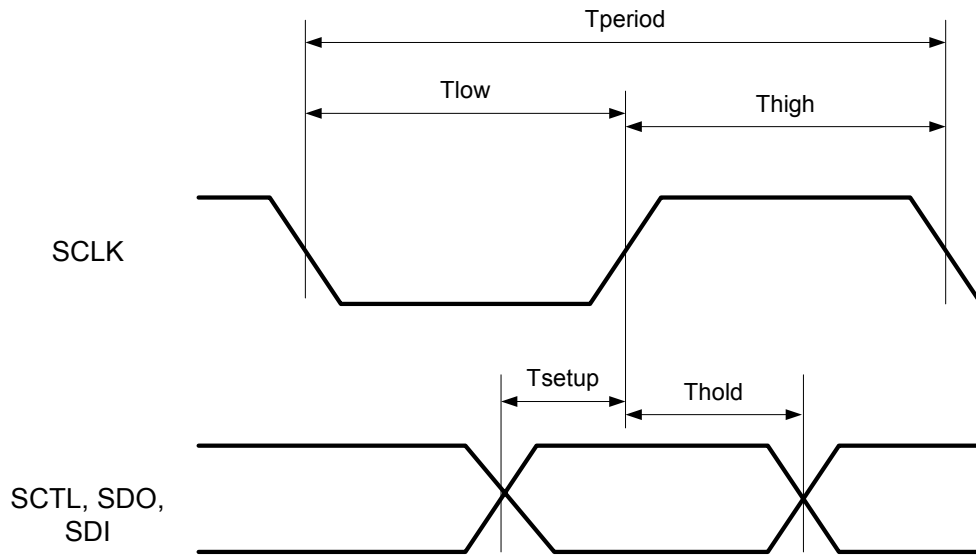


Figure 12 - Serial Interface Timing Diagram

Table 15 - Serial Interface Timing Requirements

Signal	Value	Nominal	Max	Min	Unit
SCLK	Frequency	6.75	7.00	6.50	MHz
	$T_{high}$	74	88	59	nsec
	$T_{low}$	74	88	59	nsec
SCTL	$T_{setup}$	n/a	n/a	7	nsec
	$T_{hold}$	n/a	n/a	0	nsec
SDI	$T_{setup}$	n/a	n/a	7	nsec
	$T_{hold}$	n/a	n/a	0	nsec
SDO	$T_{setup}$	n/a	n/a	7	nsec
	$T_{hold}$	n/a	n/a	0	nsec

Notes:

Minimum times are specified with 0 pF equivalent load; maximum times are specified with 30 pF equivalent load. Actual test capacitance MAY vary but results SHOULD be correlated to these specifications.

The duty cycle of SCLK SHALL be no less than 40%, no more than 60%.

## **4.3 Mechanical Specifications**

### **4.3.1 Form Factor**

The M-CARD SHALL comply with the Type II PC Card Package Dimensions with Low Voltage Keying as shown in Figure 11-3 of PC Card Standard, Release 8.0 Volume 3 [PCMCIA3].

The M-CARD SHALL include the CardBus/CardBay PC Card Recommended Connector Grounding as shown in Figure 11-40 [PCMCIA3]. Visual identification to distinguish between a Single-Stream CableCARD device and an Multi-Stream CableCARD device will be via the label on the card. Note the Mechanical Design section of [SCTE28].

### **4.3.2 Connector**

The M-CARD connector and guidance SHALL comply with the PCMCIA Cardbus Type II connector and guidance defined in [PCMCIA3].

### **4.3.3 Environmental**

When the Host and CableCARD device are operating at room temperature and humidity, no greater than 25°C ambient temperature, no greater than 95% RH non condensing, with a reference power load CableCARD device, the Host and CableCARD device SHALL NOT allow any external protruding surface point hotter than 50°C for metallic, and 60°C for non-metallic surfaces, and no non-accessible surface point hotter than 65°C.

## 5 COPY PROTECTION

Copy protection SHALL be provided for 'high value' content delivered in MPEG transport streams flowing from the CableCARD device to the Host. Such protection, including scrambling of content from CableCARD device to Host and authenticated deliver of messages through the CPU interface for permitted use of 'high value' content, is defined in [OC-MCCP].

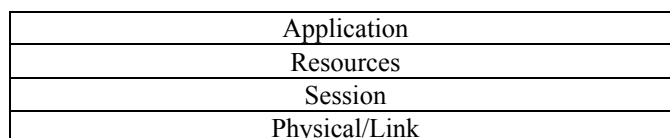
## 6 COMMAND CHANNEL OPERATION (NORMATIVE)

The Command Channel (a.k.a. Data Channel) operation for the M-CARD is duplicated from the Single-Stream CableCARD device as much as possible. The Session, Resource, and Application layers are identical between the two implementations.

The Command Channel is the control interface between the CableCARD device and the Host. This specification only defines those aspects of the Host that are required to completely specify the interactions across the interface. This specification assumes nothing about the Host design except to define a set of services which are required of the Host in order to allow the CableCARD device to operate.

The specification does not define the operation or functionality of a conditional access application on the module. The applications which MAY be performed by a module communicating across the interface are not limited to conditional access or to those described in this specification.

The Command Channel protocols are defined in several layers. The following figure shows these layers.



**Figure 13 - Command Channel Protocol Layers.**

### 6.1 Link Layer

#### 6.1.1 Backward Compatible Mode

When operating in the backward compatible single-stream mode, the Link Layer is defined as in Section 7 of [SCTE28].

#### 6.1.2 Multi-Stream CableCARD Mode

The link layer is defined as part of the physical interface (see section 4.1.7.2).

### 6.2 Transport Layer

#### 6.2.1 Backward Compatible Mode

When operating in the backward compatible single-stream mode, the CPU Interface will use the transport layer defined in Section 7 of [NRSSB].

#### 6.2.2 Multi-Stream CableCARD Mode

There is no Transport Layer in the M-CARD mode.

### 6.3 Session Layer

The session layer provides a mechanism for establishing communications between applications on the CableCARD device and resources on the Host.

### 6.3.1 Backward Compatible Mode

When operating in the backward compatible single-stream mode, the CPU Interface will use the transport layer defined in Section 7 of [NRSSB].

### 6.3.2 Multi-Stream CableCARD Mode

All sessions SHALL be opened by the CableCARD device applications. All resources, by definition, SHALL be resident in the Host.

#### 6.3.2.1 Resources with Multiple Sessions

Some resources MAY have multiple sessions so a method of identifying these is required. The session layer allows for this.

#### 6.3.2.2 SPDU Structure

The session layer uses a Session Protocol Data Unit (SPDU) structure to exchange data at session level either from the Host to the CableCARD device or from the CableCARD device to the Host. The general form of the SPDU structure is:

**Table 16 - SPDU Structure Syntax**

Syntax	No. of Bits	Mnemonic
SPDU() { spdu_tag length_field() for (i=0; i<length_value; i++) { session_object_value_byte } for (i=0; i<N; i++) { data_byte } }	8   8  8	uimsbf   uimsbf  uimsbf

**spdu\_tag** One of the values listed in Table 24.

**N** Variable, depending on the specific spdu\_tag.

##### 6.3.2.2.1 Open Session Request (*open\_session\_request*)

This object is issued by the CableCARD device to the Host in order to request the opening of a session between the device and one resource provided by the Host. The resource\_identifier MUST match in both class and type, a resource that the Host has in its list of available resources. If the version of the supplied resource identifier is zero, the Host will use the current version in its list. If the version number in the request is less than or equal to the current version number in the Host's list, the current version is used. If the requested version number is higher than the version in the Host's list, the Host will refuse the request with the appropriate return code.

**Table 17 - open\_session\_request() Syntax**

Syntax	No. of Bits	Mnemonic
<pre>open_session_request() {   open_session_request_tag   length_field() /* always equal to 0x04 */   resource_identifier() }</pre>	8	uimsbf

**open\_session\_request\_tag**      0x91

**resource\_identifier**      See section 6.5.2.

### 6.3.2.2.2 Open Session Response (open\_session\_response)

This open\_session\_response SPDU is issued by the Host to the CableCARD device in order to allocate a session number or inform the CableCARD device that its request could not be met.

**Table 18 - open\_session\_response() Syntax**

Syntax	No. of Bits	Mnemonic
<pre>open_session_response() {   open_session_response_tag   length_field() /* always equal to 0x07 */   session_status   resource_identifier()   session_nb }</pre>	8	uimsbf
	8	uimsbf
	16	uimsbf

**open\_session\_response\_tag**      0x92

**session\_status**      Status of the open session request.  
                          0x00 Session is opened  
                          0xF0 Session not opened – resource non-existent or not supported  
                          0xF1 Session not opened – resource exists but unavailable  
                          0xF2 Session not opened – resource exists but version lower than requested  
                          0xF3 Session not opened – resource busy  
                          0x01-0xEF Reserved  
                          0xF4-0xFF Reserved

**resource\_identifier**      See section 6.5.2 for description. The Host returns the actual resource identifier of the resource requested with the current version number. If the session\_status response is 0xF0, ‘resource non-existent’ then the resource identifier field SHALL be identical to that supplied in the open\_request SPDU.

**session\_nb**      A 16-bit integer number allocated by the Host for the requested session. A value of 0x00 is reserved and SHALL NOT be used. The session\_nb SHALL be used for all subsequent exchanges of APDUs between the CableCARD device and the Host until the session is closed. When the session could not be opened (session\_status ≠ 0x00), this value has no meaning.

### 6.3.2.2.3 Create Session (create\_session)

The create\_session APDU is issued by the Host to a CableCARD device to request the opening of a session.

**Table 19 - create\_session() Syntax**

Syntax	No. of Bits	Mnemonic
create_session () { create_session_tag	8	uimsbf
length_field() /* always equal to 0x06 */ resource_identifier() session_nb }	16	uimsbf

**create\_session\_tag**           0x93

**resource\_identifier**        See section 6.5.2.

**session\_nb**                 A 16-bit integer number allocated by the Host for the requested session. A value of 0x00 is reserved and SHALL NOT be used. The session\_nb SHALL be used for all subsequent exchanges of APDUs between the CableCARD device and the Host until the session is closed. When the session could not be opened (session\_status ≠ 0x00), this value has no meaning.

#### 6.3.2.2.4 Create Session Response (create\_session\_response)

The CableCARD device SHALL always issue a create\_session\_response SPDU to the Host when a create\_session SPDU is received.

**Table 20 - create\_session\_response() Syntax**

Syntax	No. of Bits	Mnemonic
create_session_response() { create_session_response_tag	8	uimsbf
length_field() /* always equal to 0x07 */ session_status	8	uimsbf
resource_identifier() session_nb }	16	uimsbf

**create\_session\_response\_tag**   0x94

**session\_status**             Status of the open session request.  
  0x00 Session is opened  
  0xF0 Session not opened – resource non-existent or not supported  
  0xF1 Session not opened – resource exists but unavailable  
  0xF2 Session not opened – resource exists but version lower than requested  
  0xF3 Session not opened – resource busy  
  0x01-0xEF Reserved  
  0xF4-0xFF Reserved

**resource\_identifier**        The M-CARD inserts in the create\_session\_response the class and type of the resource requested but with the version number than the M-CARD currently supports.

**session\_nb**                 This has the same value as the session\_nb field in the create\_session object to which this is a reply.

### 6.3.2.2.5 Close Session Request (*close\_session\_request*)

The *close\_session\_request* APDU MAY be issued by the Host or the CableCARD device to close a session.

**Table 21 - *close\_session\_request()* Syntax**

Syntax	No. of Bits	Mnemonic
<code>close_session_request() {</code>		
<code>close_session_request_tag</code>	8	uimsbf
<code>length_field() /* always equal to 0x02 */</code>		
<code>session_nb</code>	16	uimsbf
<code>}</code>		

**close\_session\_request\_tag**      0x95

**session\_nb**                      The 16-bit integer value assigned to the session.

### 6.3.2.2.6 Close Session Response (*close\_session\_response*)

The Host or CableCARD device SHALL issue the *close\_session\_response* SPDU after receiving a *close\_session\_request* SPDU.

**Table 22 - *close\_session\_response()* Syntax**

Syntax	No. of Bits	Mnemonic
<code>close_session_response() {</code>		
<code>close_session_response_tag</code>	8	uimsbf
<code>length_field() /* always equal to 0x03 */</code>		
<code>session_status</code>	8	uimsbf
<code>session_nb</code>	16	uimsbf
<code>}</code>		

**close\_session\_response\_tag**      0x96

**session\_status**                  Status of the close session request.  
     0x00 Session is closed as required  
     0xF0 *session\_nb* in the request is not allocated  
     0x01-0xEF Reserved  
     0xF1-0xFF Reserved

**session\_nb**                      The 16-bit integer value assigned to the session.

### 6.3.2.2.7 Session Number (*session\_number*)

The *session\_number* SPDU SHALL always precede a body of the SPDU containing an APDU.

**Table 23 - *session\_number()* Syntax**

Syntax	No. of Bits	Mnemonic
<code>session_number() {</code>		
<code>session_number_tag</code>	8	uimsbf
<code>length_field() /* always equal to 0x02 */</code>		
<code>session_nb</code>	16	uimsbf
<code>}</code>		

**session\_number\_tag**              0x90





## 6.5.2 Multi-Stream CableCARD Mode

The following table defines the resources for both the S-CARD and M-CARD devices.

**Table 29 - Comparison of Resources Between S-CARD and M-CARD**

Resource	S-CARD		M-CARD	
	Resource ID	Reference	Resource ID	Reference
Resource Manager	0x00010041	[SCTE28] Section 8.2	0x00010041	Section 6.6 of this document.
Application Information	0x00020081	[SCTE28] Section 8.4	0x00020081	[OC-CC] Section 7.4
Conditional Access Support	0x00030042	[SCTE28] Section 8.6	0x00030081	Section 6.9 of this document
Host Control	0x00200043	[SCTE28] Section.8	0x00200081	Section 6.10 of this document
System Time	0x00240041	[NRSSB]	0x00240041	[NRSSB]
MMI	0x00400081	[SCTE28] Section 8.3	0x004000C1	Section 6.12 of this document
Low Speed Communication <sup>1</sup> (Cable Return)	0x00605043	[OC-CC] Section 7.5	0x0060xxx2	Section 6.8 of this document.
Low Speed Communication <sup>1</sup> (Host Modem)	0x00608043	[OC-CC] Section 7.5	0x0060xxx2	Section 6.8 of this document.
Homing	0x00110042	[SCTE28] Section 8.13.3	0x00110042	[SCTE28] Section 8.13.3
Copy Protection	0x00B000C1	[SCTE28] Section 8.7	0x00B00101	[OC-MCCP] spec.
Specific Application	0x00900041	[OC-CC]	0x00900041	[OC-CC]
Generic Feature	0x002A0041	[SCTE28] Section 8.12	0x002A0041	[SCTE28] Section 8.12
Extended Channel	0x00A00041	[OC-CC]	0x00A00042	[OC-CC]
Generic Diagnostic	0x01040041	[OC-CC]	0x01040081	Section 6.17 of this document
System Control	0x002B0041	[OC-CD]	0x002B0041	[OC-CD]
CARD Capability Discovery	N/A	N/A	0x002600C1	Section 6.13 of this document.

<sup>1</sup> The Resource identifier delivered by a Host SHALL be either 0x00605043 for a Host device with Cable Return Channel, or 0x00608043 for a Host with a Host Modem (e.g. DOCSIS). If no Low Speed Communication Resource Identifier is reported by the Host then the Host device is assumed to be a FDC only. The CableCARD device MAY utilize the presence of this resource identifier as a means to identify what type of Cable Return Channel is supported by the Host.

## 6.6 Resource Manager

The material in this section and sub-sections corresponds to EIA-679-B [NRSSB] and is presented here for convenience.

The Resource Manager is a resource provided by the Host. There is only one type in the class. A maximum of 32 sessions SHALL be supported. It controls the acquisition and provision of resources to all applications. A discovery mechanism exists for the CableCARD device to determine which resources as well as the versions are supported on the Host.

It SHALL be mandatory that the first session opened be to the Resource Manager. After the session has been opened, the CableCARD device SHALL transmit a `profile_inq()` APDU to the Host. The Host SHALL respond with the `profile_reply()` APDU listing all available resources. The CableCARD device SHALL be able to issue a `profile_inq()` APDU at any time to which the Host SHALL respond.

In the rare event that a resource is modified on the Host, the Host SHALL issue a `profile_changed()` APDU to the CableCARD device. The CableCARD device SHALL then issue a `profile_inq()` APDU to the Host. The Host SHALL then issue the `profile_reply()` APDU. It SHALL be the CableCARD device's responsibility to determine if a resource has been modified such that the session should be closed and reopened.

**Table 30 - Resource Manager Resource Identifier**

Resource	Class	Type	Version	Identifier
Resource Manager	1	1	1	0x00010041

The Resource Manager includes three APDUs as described in the following table:

**Table 31 - Resource Manager APDU List**

APDU Name	Tag Value	Resource	Direction Host ↔ CableCARD
<code>profile_inq()</code>	0x9F8010	Resource Manager	←
<code>profile_reply()</code>	0x9F8011	Resource Manager	→
<code>profile_changed()</code>	0x9F8012	Resource Manager	→

Reference: [NRSSB]

### 6.6.1 `profile_inq()`

The `profile_inq()` APDU is issued by the CableCARD device to request the Host to transmit its available resources in the `profile_reply()` APDU.

**Table 32 - `profile_inq()` APDU Syntax**

Syntax	No. of Bits	Mnemonic
<pre>profile_inq() {   profile_inq_tag   length_field() /* always = 0x00 */ }</pre>	24	uimsbf

**profile\_inq\_tag**            0x9F8010

### 6.6.2 profile\_reply()

The profile\_reply() APDU is issued by the Host in response to the profile\_inq() APDU from the CableCARD device.

**Table 33 - profile\_reply() APDU Syntax**

Syntax	No. of Bits	Mnemonic
<pre>profile_reply() {   profile_reply_tag   length_field()   for (i=0; i&lt;N; i++) {     resource_identifier()   } }</pre>	24	uimsbf

**profile\_reply\_tag**      0x9F8011

**N**                      length divided by 4

### 6.6.3 profile\_changed()

The profile\_changed() APDU is transmitted to the Host if the value of any resource identifier has changed.

**Table 34 - profile\_changed() APDU Syntax**

Syntax	No. of Bits	Mnemonic
<pre>profile_changed() {   profile_changed_tag   length_field() /* always = 0x00 */ }</pre>	24	uimsbf

**profile\_changed\_tag**      0x9F8012

## 6.7 Application Information

*The material in this section is identical to the corresponding section in [SCTE28] and is presented here for convenience.*

The Application Information resource resides in the Host. The POD shall only open one session to it after it has completed the profile inquiry operation with the Resource Manager resource.

The Application Information resource provides:

- Support for the Host to expose its display characteristics to the POD
- Support for the POD to expose its applications to the Host
- Support for the POD to deliver HTML pages to the Host

The Application Information resource has been changed to type 2 to reflect the changes listed in this section as compared to [NRSSB]. (The CableCARD device is not required to support type 1.) During initialization, the POD module opens a session to the Application Information resource on the Host. This session SHALL remain open during normal operation.

**Table 35 - Application Information Resource Identifier**

Resource	Class	Type	Version	Identifier
Application Info	2	2	1	0x00020081

The Application Information resource includes four APDUs as described in Table 36:

**Table 36 - Application Information APDU List**

APDU Name	Tag Value	Resource	Direction Host ↔ CableCARD
application_info_req()	0x9F8020	Application Info	→
application_info_cnf()	0x9F8021	Application Info	←
server_query()	0x9F8022	Application Info	→
server_reply()	0x9F8023	Application Info	←

### 6.7.1 application\_info\_req()

After the CableCARD device has opened the Application Information resource, the Host SHALL send an `application_info_req()` APDU to the CableCARD device. This SHALL include the display capabilities of the Host. The CableCARD device SHALL reply with an `application_info_cnf()` APDU to describe its support applications. The Host MAY send this APDU anytime later to which the CableCARD device SHALL reply.

**Table 37 - applicatio\_info\_req() APDU Syntax**

Syntax	No. of Bits	Mnemonic
<code>application_info_req() {</code>		
<code>application_info_req_tag</code>	24	uimsbf
<code>length_field()</code>		
<code>display_rows</code>	16	uimsbf
<code>display_columns</code>	16	uimsbf
<code>vertical_scrolling</code>	8	uimsbf
<code>horizontal_scrolling</code>	8	uimsbf
<code>multi_window_support</code>	8	uimsbf
<code>data_entry_support</code>	8	uimsbf
<code>HTML_support</code>	8	uimsbf
<code>if (HTML_support == 1) {</code>		
<code>link_support</code>	8	uimsbf
<code>form_support</code>	8	uimsbf
<code>table_support</code>	8	uimsbf
<code>list_support</code>	8	uimsbf
<code>image_support</code>	8	uimsbf
<code>}</code>		
<code>}</code>		

**application\_info\_req\_tag** 0x9F8020

**display\_rows** Defines the number of rows the Host device can support. If the Host supports more than 255, set this value equal to 0xFF.

**display\_columns** Defines the number of columns the Host device can support. If the Host supports more than 255, set this value equal to 0xFF.

<b>vertical_scrolling</b>	Defines if the Host supports vertical scrolling. 0x00 – Vertical scrolling not supported 0x01 – Vertical scrolling supported 0x02-0xFF - Reserved
<b>horizontal_scrolling</b>	Defines if the Host supports horizontal scrolling. 0x00 – Horizontal scrolling not supported 0x01 – Horizontal scrolling supported 0x02-0xFF - Reserved
<b>multi_window_support</b>	Defines the window support capability of the Host. 0x00 – Full screen 0x01 – Overlay 0x02 – Multiple windows 0x03-0xFF - Reserved
<b>data_entry_support</b>	Defines the preferred data entry capability of the Host. 0x00 – None 0x01 – Last/Next 0x02 – Numeric Pad 0x03 – Alpha keyboard with mouse 0x04-0xFF - Reserved
<b>HTML_support</b>	Defines the HTML support capability of the Host. Baseline HTML support is described in Appendix B. 0x00 – Baseline 0x01 – Custom 0x02 – HTML 3.2 0x03 – XHTML 1.0 0x04-0xFF - Reserved
<b>link_support</b>	Defines whether the Host can support single or multiple links. 0x00 – One link 0x01 – Multiple link 0x02-0xFF - Reserved
<b>form_support</b>	Defines the Form support capability of the Host. 0x00 – None 0x01 – HTML 3.2 w/o POST method 0x02 – HTML 3.2 0x03-0xFF - Reserved
<b>table_support</b>	Defines the Table support capability of the Host. 0x00 – None 0x01 – HTML 3.2 0x02-0xFF - Reserved
<b>list_support</b>	Defines the List support capability of the Host 0x00 – None 0x01 – HTML 3.2 w/o Descriptive Lists 0x02 – HTML 3.2 0x03-0xFF - Reserved
<b>image_support</b>	Defines the Image capability of the Host

0x00 – None  
 0x01 – HTML 3.2 – PNG Picture under RGW w/o resizing  
 0x02 – HTML 3.2  
 0x03-0xFF - Reserved

### 6.7.2 application\_info\_cnf()

The CableCARD device SHALL transmit the application\_info\_cnf() APDU after receiving an application\_info\_req() from the Host.

**Table 38 - application\_info\_cnf() APDU Syntax**

Syntax	No. of Bits	Mnemonic
application_info_cnf() {		
application_info_cnf_tag	24	uimsbf
length_field()		
CableCARD_manufacturer_id	16	uimsbf
CableCARD_version_number	16	uimsbf
number_of_applications	8	uimsbf
for (i=0; i<number_of_applications; i++) {		
application_type	8	uimsbf
application_version_number	16	uimsbf
application_name_length	8	uimsbf
for (j=0; j<application_name_length; j++) {		
application_name_byte	8	uimsbf
}		
application_url_length	8	uimsbf
for (j=0; j<application_url_length; j++) {		
application_url_byte	8	uimsbf
}		
}		

**application\_info\_cnf\_tag** 0x9F8021

**CableCARD\_manufacturer\_id** The first byte specifies the CableCARD device manufacturer while the second byte is defined by the CableCARD device manufacturer to privately identify product generation and derivatives.

0x00XX – Motorola  
 0x01XX – Scientific-Atlanta  
 0x02XX – SCM Microsystems  
 0x0300-0xFFFF – Reserved for future manufacturers

**manufacturer\_version\_number** Privately defined by the CableCARD device manufacturer.

**number\_of\_applications** Number of applications in the following for loop.

**application\_type** Type of application  
 0x00 – Conditional access  
 0x01 – Copy protection  
 0x02 – IP service  
 0x03 – Network interface (SCTE 55-2)  
 0x04 – Network interface (SCTE 55-1)  
 0x05 – Network interface (DSG)  
 0x06 - Diagnostic  
 0x07 - Undesignated

0x08-0xFF – Reserved for future applications

**application\_version\_number** Defined by the CableCARD application supplier.

**application\_name\_length** Length of the application name in the following for loop. The maximum value is 32. This value SHALL be equal to 0 for applications which do not have an MMI interface.

**application\_name\_byte** The commercial name of the application specified as a text string in ASCII format. The Host SHALL replace the default generic identifier of the CableCARD device's application with the application name. The application name, when selected by the user, triggers a Host initialized MMI dialog.

The Host SHALL be capable of displaying at least eight different CableCARD device application name strings in its top menu. The application name length SHALL be limited to 32 characters.

**application\_url\_length** Length of the application url in the following for loop.

**application\_url\_byte** Defines the URL of the CableCARD device application's top level HTML page in the CableCARD device memory. The application URL MAY or MAY not be displayed in the Host top menu. The Host SHALL use the application URL in a server\_query() APDU to initialize an MMI dialog with the CableCARD device application, when an object identified by either the application name or the application URL is selected in the Host menu.

### 6.7.3 server\_query()

The Host SHALL send a server\_query() APDU to the CableCARD device to request the information in the CableCARD device file server system pointed by a specific URL. The URL defines the location of the data that the Host is requesting. Upon receipt of the URL, the CableCARD device locates the requested data and provides it back to the Host in the server\_reply APDU. The Host SHALL process and display the data returned in the server\_reply() APDU in a timely manner.

**Table 39 - server\_query() APDU Syntax**

Syntax	No. of Bits	Mnemonic
server_query() {		
server_query_tag	24	uimsbf
length_field()		
transaction_number	8	uimsbf
header_length	16	uimsbf
for (i=0; i<header_length; i++) {		
header_byte	8	uimsbf
}		
url_length	16	uimsbf
for (i=0; i<url_length; i++) {		
url_byte	8	uimsbf
}		
}		

**server\_query\_tag** 0x9F8022

**transaction\_number** A number supplied by the Host issued from an 8-bit cyclic counter that identifies each server\_query() APDU and allows the Host to route the server\_reply to the correct MMI dialog

<b>header_length</b>	The number of header bytes in the following for loop.
<b>header_byte</b>	Each header_byte is an octet of an optional parameter that uses the same format as the HTTP/1.1 request header to pass additional parameters related to the request, like browser version, accepted mime types, etc. A Host not supporting headers SHALL set header_length to 0x00. The CableCARD device MAY also ignore this parameter.
<b>url_length</b>	The number of URL bytes in the following for loop.
<b>url_byte</b>	Each url_byte is an octet of a parameter that defines a protocol, domain, and location for the transfer of data. For the purposes of an application running on a CableCARD device, the URL MUST allow the transfer of a file of data from the CableCARD device to the Host.

The access indicator is "CableCARD".

The second part of the URL is the Host. The convention for "current server" (i.e., the server that generated the current page) can be used and is indicated by an empty Host.

The third part of the URL is the file location. This is indicated by a hierarchical directory/file path.

For example, in order to request the file menu.html from the directory /apps/user/program\_guide on the CableCARD device, the properly constructed URL would be:

```
CableCARD///apps/user/program_guide/menu.html
```

If, after receiving a server\_reply from the CableCARD device, the Host has data that it wants to send to the CableCARD device, the Host can do so through a server\_query. In this case, the last part of the URL contains a list of name-value pairs separated by "&". This list is preceded by "?". A properly constructed URL would be:

```
CableCARD///path/file?name1=value1&name2=value2&...
```

Such a URL sent to an application on the CableCARD device as a response to a server\_reply would cause the name-value pairs to be processed by the application. Data entered and selected by the Host MAY be sent to the CableCARD device through the use of these name-value pairs as part of the URL in a server\_query APDU.

#### **6.7.4 server\_reply()**

The server\_reply() is issued by the CableCARD device in response to a server\_query() from the Host.

**Table 40 - server\_reply() APDU Syntax**

Syntax	No. of Bits	Mnemonic
server_reply() { server_reply_tag length_field() transaction_number file_status header_length for (i=0; i<header_length; i++) { header_byte } file_length for (i=0; i<url_length; i++) { file_byte } }	24  8 8 16  8  16 8	uimsbf  uimsbf uimsbf uimsbf  uimsbf  uimsbf uimsbf

**server\_reply\_tag**           0x9F8023

**transaction\_number**       A number supplied by the Host issued from an 8-bit cyclic counter that identifies each server\_query() APDU and allows the Host to route the server\_reply to the correct MMI dialog.

**file\_status**               Notifies the Host of the status of the requested file.  
                   0x00 – OK  
                   0x01 – URL not found  
                   0x02 – URL access not granted  
                   0x03-0xFF – Reserved

**header\_length**            The number of header bytes in the following for loop.

**header\_byte**              Each header\_byte is an octet of an optional parameter that uses the same format as the HTTP/1.1 request header to pass additional parameters related to the request, like browser version, accepted mime types, etc. A Host not supporting headers SHALL set header\_length to 0x00. The CableCARD device MAY also ignore this parameter.

**file\_length**               The number of bytes in the following for loop.

**file\_byte**                 The requested URL file.

## 6.8 Low Speed Communication

The Low Speed Communication resource is used to support the identification of the Forward Data Channel (FDC), the Reverse Data Channel (RDC), and any type of Host modem implementations. The Low Speed Communication resource is not a means for passing upstream/downstream OOB data to/from CableCARD device via the CableCARD-Host interface. All downstream OOB data SHALL be passed directly to/from the CableCARD device via the CableCARD-Host OOB Interface. Support for Section 8.7 of EIA-679B Part B is not permitted.

**Table 41 - A Low Speed Communication Resource**

Resource	Class	Type	Version	Identifier (hex)
Low_Speed_Communication (Cable Return)	96	50	3	00605043
Low_Speed_Communication (Host Modem)	96	80	3	00608043

The Low\_Speed\_Communication Identifier can be either 0x00605043 for a Host device with Cable Return Channel, or 0x00608043 for a Host with a Host Modem (e.g. DOCSIS). If no Low Speed Communication Resource Identifier reported by the Host then the Host device is assumed to be a FDC only.

## 6.9 CA Support

*The material in this section corresponds to EIA-679-B [NRSSB] and is presented here for convenience.*

This resource provides a set of objects to support Conditional Access applications. The CableCARD device will open a single session after the Application Information session initialization has completed. In the case of multiple transport streams, there is a transport ID (LTSID) included in the APDU. The Host will then send a CA Info Inquiry APDU to the CableCARD device, which will respond by returning a CA Info APDU with the appropriate information. The session SHALL be kept open. This resource is modified from the version in [SCTE28]. The Resource Type has been changed to indicate that the resource has been modified.

This APDU is modified from the PCMCIA CableCARD device to support the local transport stream ID (LTSID).

**Table 42 - CA Support Resource**

Resource	Class	Type	Version	Identifier
CA Support	3	2	1	00030081

The CA support resource consists of 4 APDUs.

**Table 43 - CA Support APDUs**

APDU Name	Tag Value	Resource	Direction Host ↔ CableCARD
ca_info_inquiry()	0x9F8030	CA Support	→
ca_info()	0x9F8031	CA Support	←
ca_pmt()	0x9F8032	CA Support	→
ca_pmt_reply()	0x9F8033	CA Support	←
ca_update()	0x9F8034	CA Support	←

### 6.9.1 ca\_info\_inquiry

The ca\_info\_inquiry APDU is transmitted by the Host to the CableCARD device after the CA session is opened.

**Table 44 - ca\_info\_inquiry() APDU Syntax**

Syntax	No. of Bits	Mnemonic
ca_info_inquiry() { ca_info_inquiry_tag	24	uimsbf
length_field() /* always = 1 */ ltsid	8	uimsbf
}		

**ca\_info\_inquiry\_tag**           0x9F8020

**ltsid**                        Local Transport Stream ID

### 6.9.2 ca\_info

After the CableCARD device receives a ca\_info\_inquiry APDU, it SHALL respond with the ca\_info APDU with the appropriate CA\_system\_id.

**Table 45 - ca\_info() APDU Syntax**

Syntax	No. of Bits	Mnemonic
ca_info() { ca_info_tag	24	uimsbf
length_field() ltsid	8	uimsbf
for (i=0; i<N ;i++) { CA_system_id	16	uimsbf
}		
}		

**ca\_info\_tag**                0x9F8020

**ltsid**                        Local Transport Stream

**CA\_system\_id**              CA system ID's supported by the CableCARD device

### 6.9.3 ca\_pmt

**Table 46 - ca\_pmt() APDU Syntax**

Syntax	No. of Bits	Mnemonic
ca_pmt() {		
ca_pmt_tag	24	uimsbf
length_field()		
program_index	8	uimsbf
transaction_id	8	uimsbf
ltsid	8	uimsbf
program_number	16	uimsbf
source_id	16	uimsbf
ca_pmt_cmd_id	8	uimsbf
reserved	4	bslbf
program_info_length	12	uimsbf
if (program_info_length != 0) {		
CA_descriptor() /* program level */		
}		
for (i=0; i<N1; i++) {		
stream_type	8	uimsbf
reserved	3	bslbf
elementary_PID /* elementary stream PID*/	13	uimsbf
reserved	4	bslbf
ES_info_length	12	uimsbf
if (ES_info_length != 0) {		
CA_descriptor() /* ES level */		
}		
}		
}		

**ca\_pmt\_tag** 0x9F8022

**program\_index** Program index values range from 0 to max\_programs-1, where max\_programs is the value returned by M-CARD in the program\_profile().

**transaction\_id** An 8-bit value, generated by the Host, that will be returned in the corresponding ca\_pmt\_reply() and/or ca\_update() from the M-CARD. The transaction\_id allows the Host to match the M-CARD's replies with the corresponding requests. The Host should increment the value, modulo 255, with every message it sends. A separate transaction\_id counter shall be maintained for each program index, so that the transaction\_ids increment independently for each index.

**ltsid** Local Transport Stream ID

**program\_number** The MPEG program number as defined in [ISO/IEC13818-1]

**source\_id** The source ID as defined in [SCTE65].

**ca\_pmt\_cmd\_id** This parameter indicates what response is required from the application to a ca\_pmt APDU.

00 Reserved

01 ok\_descrambling – The Host does not expect an answer to the ca\_pmt APDU and the CableCARD device can start descrambling the program or start an MMI dialog immediately.

- 02 ok\_mmi – The application can start a MMI dialog but SHALL NOT start descrambling before reception of a new ca\_pmt APDU with ca\_pmt\_cmd\_id set to “ok\_descrambling”. In this case, the Host SHALL guarantee that a MMI session can be opened by the CableCARD device.
- 03 query – The Host expects to receive a ca\_pmt\_reply APDU. The CableCARD device SHALL NOT start descrambling or start an MMI dialog before reception of a new ca\_pmt APDU with ca\_pmt\_cmd\_id set to either “ok\_descrambling” or “ok\_mmi”.
- 04 not selected – Indicates to the CableCARD device that the Host no longer requires that the CableCARD device attempt to descramble the service. The CableCARD device SHALL close any MMI dialog it has opened.
- 05-FF Reserved

<b>program_info_length</b>	Number of bytes in the program_info field.
<b>CA_descriptor</b>	The MPEG CA descriptor as defined in [ISO/IEC13818-1].
<b>stream_type</b>	The MPEG stream type as defined in [ISO/IEC13818-1].
<b>elementary_PID</b>	The MPEG elementary PID as defined in [ISO/IEC13818-1].
<b>ES_info_length</b>	Number of bytes in the elementary stream information section.

### Processing Rules for ca\_pmt

The Host SHALL send a new ca\_pmt APDU when:

- the user selects a different program
- the version number changes
- the current next indicator changes

The ca\_pmt APDU consists of entitlement control information extracted from the Program Map Table (PMT) by the Host and sent to the M-CARD device. This control information allows the M-CARD to locate and filter Entitlement Control Message (ECM) streams, and assign the correct ECM stream to each scrambled component.

The ca\_pmt APDU contains CA\_descriptors from the PMT of a selected program. If several programs are selected, then the Host shall send a ca\_pmt APDU for each program to the M-CARD device. The Host shall only include CA descriptors in the ca\_pmt, and shall not include any other descriptors from the PMT.

CA descriptors may be included in the ca\_pmt at the program level and at the elementary stream level. The program level CA descriptor shall apply to all elementary streams that have no elementary stream level CA descriptor. When CA descriptors are present at the elementary stream level, they shall apply to their own elementary streams.

The CA descriptors in the PMT are provided by the conditional access system to inform the M-CARD of which PID stream carries the Entitlement Control Messages associated with each program or elementary stream. The Host may elect to use some or all of the CA descriptors included in an individual program’s PMT to query or request descrambling of that program. However, the CA descriptors in the ca\_pmt must always appear at the same program and elementary stream levels as they originally appeared in the program’s PMT. Therefore, program level CA descriptors from a program’s PMT must continue to be provided at the program level in the ca\_pmt, and cannot be replicated at the elementary stream level, even if the Host elects to only request decryption of individual streams from that program.

The `program_index` tracks the program number and `ltsid` assigned to a particular CA resource in the M-CARD. The `program_index` shall be carried in the `ca_pmt` to allow the M-CARD to maintain a similar index table to track the assignments that it receives from the Host. The assignments to each `program_index` should be updated as old programs are replaced by new programs, thereby maintaining the total number of active programs within the M-CARD's limitations.

### **Example of Program Index Tables**

The figure below shows a Program Index Table for a Host and M-CARD with the ability to decrypt only two programs. The Program Index Tables track the Host's assignments of the programs that the M-CARD is to decrypt.

**Host Program Index Table**

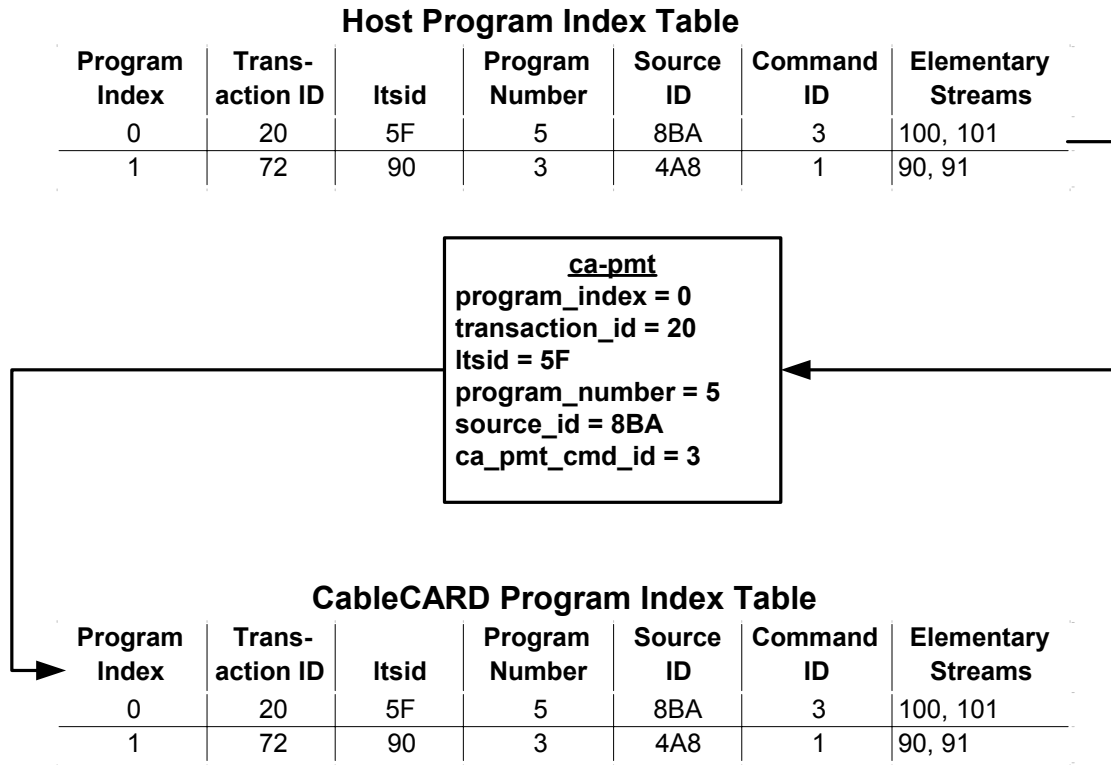
Program Index	Transaction ID	ltsid	Program Number	Source ID	Command ID	Elementary Streams
0	19	15	2	1B1	1	51, 52, 53
1	72	90	3	4A8	1	90, 91

**CableCARD Program Index Table**

Program Index	Transaction ID	ltsid	Program Number	Source ID	Command ID	Elementary Streams
0	19	15	2	1B1	1	51, 52, 53
1	72	90	3	4A8	1	90, 91

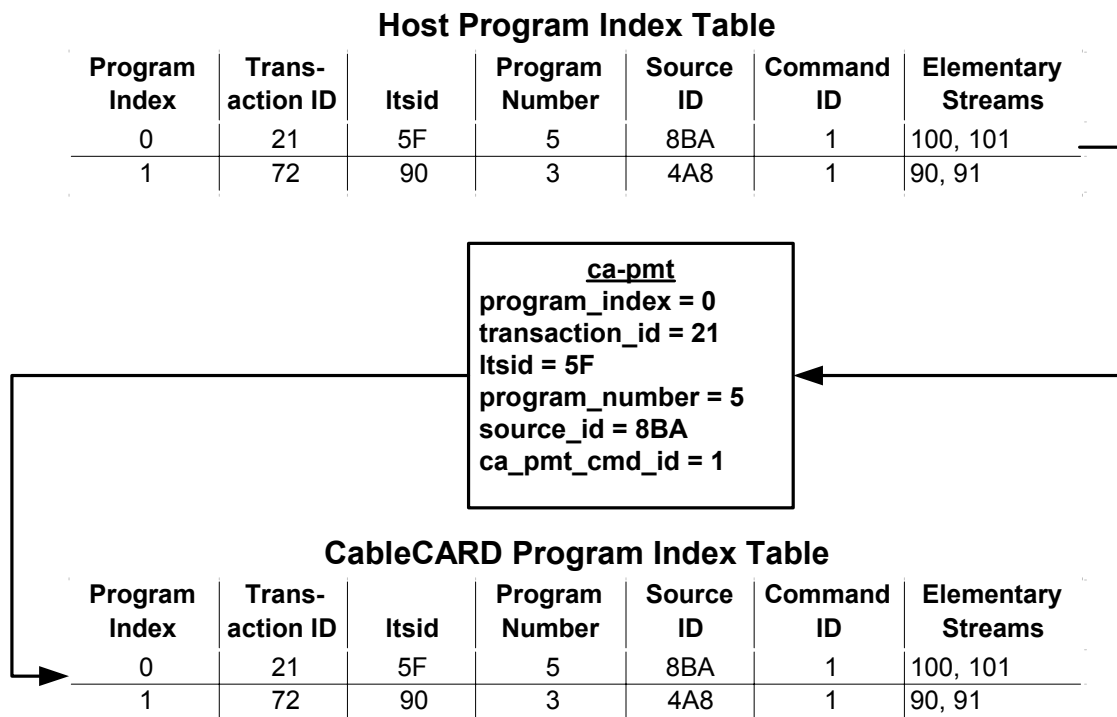
**Figure 14 - Program Index Table 1**

The next figure shows that the Host has made a change to the Program Index 0, to query the M-CARD about a new program. The `transaction_ID` is incremented, and the query command is passed through a new `ca_pmt`, which updates the M-CARD's Program Index 0. The M-CARD would then reassign its CA resource to evaluate the Host's query, and prepare a `ca_pmt_reply`. The `ca_pmt_reply` would include the same `transaction_id` that the Host assigned to the `ca_pmt`, and the program index, to uniquely identify the reply.



**Figure 15 - Program Index Table 2**

The M-CARD may not begin descrambling the new program on the basis of a query, even if it replies that descrambling is possible. The final figure shows the Host following up with another ca\_pmt, focused on the same program\_index, with an ok\_descrambling command and new transaction\_ID. The M-CARD will update its index accordingly and begin to descramble the program.



**Figure 16 - Program Index Table 3**

The M-CARD shall treat the arrival of each `ca_pmt` as a new program request, which is either an additional program request, or a replacement for an existing request.

When a `ca_pmt` arrives, the M-CARD shall allocate the necessary CA resources and assign them to the program index designated in the `ca_pmt`. If resources have previously been assigned to this program index, then the former allocations shall be released and new allocations made to reflect the contents of the new `ca_pmt`. Therefore, only a single `ca_pmt` will be needed to initiate a complete transition between program assignments and maintain synchronization of resource allocations between the Host and M-CARD.

The Host shall assign a `transaction_id` to each `ca_pmt`. The Host shall increment the `transaction_id` with every `ca_pmt` it sends. The Host shall maintain a separate `transaction_id` counter for each program index, so that the `transaction_ids` increment independently for each index.

The M-CARD shall use the `transaction_id` from the `ca_pmt` in any replies or updates it sends regarding that `ca_pmt`. This allows the host to match the M-CARD's messages with the corresponding `ca_pmts`. Obsolete replies or updates may then be discarded.

#### 6.9.4 `ca_pmt_reply`

The `ca_pmt_reply` is sent by the M-CARD device to the Host after receiving a `ca_pmt` APDU with the `command_id` set to query, and providing that the M-CARD has time to reply before the Host sends another `ca_pmt()` with the same `program_index`. The M-CARD shall include the `transaction_id` and program index of the `ca_pmt` that initiated the reply. This ensures that the Host can correlate `ca_pmt_reply` APDUs to a specific `ca_pmt` APDU.

**Table 47 - ca\_pmt\_reply() APDU Syntax**

Syntax	No. of Bits	Mnemonic
ca_pmt_reply() {		
ca_pmt_reply_tag	24	uimsbf
length_field()		
program_index	8	uimsbf
transaction_id	8	uimsbf
ltsid	8	uimsbf
program_number	16	uimsbf
source_id	16	uimsbf
CA_enable_flag	1	bslbf
if (CA_enable_flag == 1) {		
CA_enable /* program level */	7	uimsbf
}		
else {		
reserved	7	uimsbf
}		
for (i=0; i<N1; i++) {		
reserved	3	bslbf
elementary_PID /* elementary stream PID*/	13	uimsbf
CA_enable_flag	1	bslbf
if (CA_enable_flag == 1) {		
CA_enable /* elementary stream level*/	7	uimsbf
}		
else {		
reserved	7	uimsbf
}		
}		
}		

**ca\_pmt\_reply\_tag**           0x9F8023

**program\_index**           The same program index that was used in the original ca\_pmt. The combination of transaction\_id and program\_index uniquely identifies this update.

**transaction\_id**           The same 8-bit transaction\_id that was used in the original ca\_pmt.

**ltsid**                    Local Transport Stream ID

**program\_number**         The MPEG program number as defined in [ISO/IEC13818-1]

**source\_id**                The source ID as defined in [SCTE65].

**elementary\_PID**         The MPEG elementary PID as defined in [ISO/IEC13818-1].

**ca\_enable**                Indicates whether the CableCARD device is able to perform the descrambling operation requested in the ca\_pmt APDU.

  00 Reserved

  01 Descrambling possible with no extra conditions.

  02 Descrambling possible under conditions (purchase dialog). The CableCARD device has to enter a purchase dialog with the user before being able to descramble.

  03 Descrambling possible under conditions (technical dialog). The CableCARD device has to enter a technical dialog with the user before

being able to descramble (e.g., request fewer elementary streams because the descrambling capabilities are limited).

04-70 Reserved

71 Descrambling not possible (no entitlement). The selected program is not entitled and is not available for purchase.

72 Reserved

73 Descrambling not possible (technical reasons), for example if all the elementary streams capable are being used.

74-FF Reserved

### 6.9.5 ca\_update

The M-CARD device shall use the ca\_update APDU to inform the Host when the CA\_enable status of a ca\_pmt\_reply has changed. The M-CARD shall include the same transaction\_id and program index in the ca\_update as was used in the original ca\_pmt\_reply.

**Table 48 - ca\_update() APDU Syntax**

Syntax	No. of Bits	Mnemonic
ca_update() {		
ca_update_tag	24	uimsbf
length_field()		
program_index	8	uimsbf
transaction_id	8	uimsbf
ltsid	8	uimsbf
program_number	16	uimsbf
source_id	16	uimsbf
CA_enable_flag	1	bslbf
if (CA_enable_flag == 1) {		
CA_enable /* program level */	7	uimsbf
}		
else {		
reserved	7	uimsbf
}		
for (i=0; i<N1; i++) {		
reserved	3	bslbf
elementary_PID /* elementary stream PID*/	13	uimsbf
CA_enable_flag	1	bslbf
if (CA_enable_flag == 1) {		
CA_enable /* elementary stream level*/	7	uimsbf
}		
else {		
reserved	7	uimsbf
}		
}		
}		

**ca\_update\_tag** 0x9F8024

**program\_index** The same program index that was used in the original ca\_pmt. The combination of transaction\_id and program\_index uniquely identifies this update.

**transaction\_id** The same 8-bit transaction\_id that was used in the original ca\_pmt.

**ltsid** Local Transport Stream ID

**program\_number** The MPEG program number as defined in [ISO/IEC13818-1]

<b>source_id</b>	The source ID as defined in [SCTE65].
<b>elementary_PID</b>	The MPEG elementary PID as defined in [ISO/IEC13818-1].
<b>ca_enable</b>	Indicates whether the CableCARD device is able to perform the descrambling operation requested in the ca_pmt APDU. <ul style="list-style-type: none"> <li>00 Reserved</li> <li>01 Descrambling possible with no extra conditions.</li> <li>02 Descrambling possible under conditions (purchase dialog). The CableCARD device has to enter a purchase dialog with the user before being able to descramble.</li> <li>03 Descrambling possible under conditions (technical dialog). The CableCARD device has to enter a technical dialog with the user before being able to descramble (e.g., request fewer elementary streams because the descrambling capabilities are limited).</li> <li>04-70 Reserved</li> <li>71 Descrambling not possible (no entitlement). The selected program is not entitled and is not available for purchase.</li> <li>72 Reserved</li> <li>73 Descrambling not possible (technical reasons), for example if all the elementary streams capable are being used.</li> <li>74-FF Reserved</li> </ul>

## 6.10 Host Control

This resource allows the CableCARD device to set up the Host OOB RF receiver and transmitter, if available, and to allow the CableCARD device the capability to tune the Host's FAT tuner under certain conditions defined by the Homing resource. Unless a session to the Homing resource has been opened and the CableCARD device granted access to the FAT tuner, or if the Host is in the standby state and allows the CableCARD device access to the tuner (the Host MAY or MAY NOT allow this), the Host SHOULD NOT grant any request by the CableCARD device to tune the FAT tuner. Only one session SHALL be opened. This Resource has been modified from the version in [SCTE28]. The resource Type value has been changed to indicate the modified resource. Support of type 1 is optional.

**Table 49 - Host Control Support Resource**

Resource	Class	Type	Version	Identifier
Host Control	2	2	3	0x00020083

The Host Control resource consists of 6 APDUs.

**Table 50 - Host Control Support APDUs**

APDU Name	Tag Value	Resource	Direction Host ↔ CableCARD
OOB_TX_tune_req()	0x9F8404	Host Control	←
OOB_TX_tune_cnf()	0x9F8405	Host Control	→
OOB_RX_tune_req()	0x9F8406	Host Control	←
OOB_RX_tune_cnf()	0x9F8407	Host Control	→
inband_tune_req()	0x9F8408	Host Control	←
inband_tune_cnf()	0x9F8409	Host Control	→

### 6.10.1 OOB\_TX\_tune\_req

If the DSG option is not selected, the CableCARD device SHALL use the OOB\_TX\_tune\_req APDU to set up the Host's RF transmitter.

**Table 51 - OO\_TX\_tune\_req() APDU Syntax**

Syntax	No. of Bits	Mnemonic
OOB_TX_tune_req() { OOB_TX_tune_req_tag length_field() RF_TX_frequency_value RF_TX_coding_value RF_TX_rate_value }	24	uimsbf
	16	uimsbf
	8	uimsbf
	8	uimsbf

**OOB\_TX\_tune\_req\_tag**      0x9F8404

**RF\_TX\_frequency\_value**      This field defines the frequency of the RF transmitter, in KHz.

**RF\_TX\_coding\_value**      This value defines the power level of the RF transmitter, in units of 0.5 dBmV. The value 0x00 SHALL correspond to an output level of 0 dBmV.

**RF\_TX\_rate\_value**      This value defines the bit rate of the RF transmitter. The format is defined in Table 52.

0b00    256 kbps  
0b01    2,048 kbps  
0b10    1,544 kbps  
0b11    3,088 kbps

The LSB is used to denote whether the spectrum is inverted. The format is defined in Table 52.

0    Spectrum is non-inverted.  
1    Spectrum is inverted.

**Table 52 - OOB Transmit Rate Format**

Bit								
7	6	5	4	3	2	1	0	
Rate		Reserved						

### 6.10.2 OOB\_TX\_tune\_cnf

Upon reception of an OOB\_TX\_tune\_req APDU and tuning the transmitter, the Host SHALL send the OOB\_TX\_tune\_cnf APDU to the CableCARD device.

**Table 53 - OOB\_TX\_tune\_cnf() APDU Syntax**

Syntax	No. of Bits	Mnemonic
OOB_TX_tune_cnf() { OOB_TX_tune_cnf_tag	24	uimsbf
length_field() status_field }	8	uimsbf

**OOB\_TX\_tune\_cnf\_tag**      0x9F8405

**status\_field**

This field returns the status of the *OOB\_TX\_tune\_req()*. If the request was granted and the RF Transmitter set to the desired configuration, *Status\_field* will be set to 0x00. If the Host is a unidirectional Host, *Status\_field* SHALL be set to 0x01; the CableCARD device SHALL NOT attempt to perform RF transmit operations after receiving an *OOB\_TX\_tune\_cnf()* with *Status\_field* set to 0x01. If any of the parameters passed to the Host are outside of the Host's tuning rate, then the Host SHALL transmit the *OOB\_TX\_tune\_cnf()* with *Status\_field* set to 0x03. Otherwise *Status\_field* will be set to one of the following values:

- 00 Tuning granted
- 01 Tuning denied – RF transmitter no physically available
- 02 Tuning denied – RF transmitter busy
- 03 Tuning denied – Invalid parameters
- 04 Tuning denied – Other reasons
- 05-0xFF Reserved

### 6.10.3 OOB\_RX\_tune\_req

If the DSG option is not selected, the CableCARD device SHALL use the OOB\_RX\_tune\_req APDU to set up the Host's RF receiver.

**Table 54 - OO\_RX\_tune\_req() APDU Syntax**

Syntax	No. of Bits	Mnemonic
OOB_RX_tune_req() { OOB_RX_tune_req_tag	24	uimsbf
length_field() RF_RX_frequency_value	16	uimsbf
RF_RX_data_rate }	8	uimsbf

**OOB\_RX\_tune\_req\_tag**      0x9F8406

**RF\_RX\_frequency\_value**

This field defines the frequency of the RF receiver.  
Frequency = value \* 0.05 + 50 MHz.

**RF\_RX\_data\_rate**

This value defines the bit rate and spectral inversion of the RF transmitter. The format is defined in Table 55.

The following is the definition of the bit rate.

- 0b00 256 kbps
- 0b01 2,048 kbps
- 0b10 1,544 kbps
- 0b11 3,088 kbps

The following is the definition of the Spectral Inversion (SPEC).

- 0 Spectrum is non-inverted
- 1 Spectrum is inverted

**Table 55 - OOB Transmit Rate Format**

Bit							
7	6	5	4	3	2	1	0
Rate							SPEC

**6.10.4 OOB\_RX\_tune\_cnf**

Upon reception of an OOB\_RX\_tune\_req APDU and tuning the RF receiver, the Host SHALL send the OOB\_RX\_tune\_cnf APDU to the CableCARD device. The OOB\_RX\_tune\_cnf APDU SHOULD only be transmitted after either the requested frequency has been tuned and acquired ("tune time"), or 500 msec has elapsed since receiving the request, whichever comes first.

**Table 56 - OO\_RX\_tune\_cnf() APDU Syntax**

Syntax	No. of Bits	Mnemonic
OOB_RX_tune_cnf() {		
OOB_RX_tune_cnf_tag	24	uimsbf
length_field()		
status_field	8	uimsbf
}		

**OOB\_RX\_tune\_cnf\_tag** 0x9F8407

**status\_field**

Returns the status of the RF receiver. The following values are to be used:

- 00 Tuning granted
- 01 Tuning denied – RF receiver not physically available
- 02 Tuning denied – RF receiver busy
- 03 Tuning denied – Invalid parameters
- 04 Tuning denied – Other reasons
- 05-0xFF Reserved

**6.10.5 inband\_tune\_req**

The inband\_tune\_req APDU allows for the CableCARD device to request the Host to tune the inband QAM tuner. The APDU will allow support for tuning to a source\_id or a frequency with the modulation type.

**Table 57 - inband\_tune\_req() APDU Syntax**

Syntax	No. of Bits	Mnemonic
inband_tune_req() { inband_tune_req_tag length_field() ltsid tune_type if (tune_type == 0x00) { source_id } else if (tune_type == 0x01) { tune_frequency_value modulation_value } }	24  8 8 16  16 8	uimsbf  uimsbf uimsbf uimsbf  uimsbf uimsbf

**inband\_tune\_req\_tag**      0x9F8408

**ltsid**                      Local Transport Stream ID

**tune\_type**                Determines whether to use the source ID value or the frequency and modulation values.  
                               00 Source ID  
                               01 Frequency  
                               02-0xFF Reserved

**source\_id**                When tune\_type = 0x00, the source\_id is a 16 bit unsigned integer as defined in Section 8.8.3 of [SCTE28].

**tune\_frequency\_value**    When tune\_type = 0x01, tune\_frequency\_value contains the frequency for the Host to tune. The frequency is calculated by multiplying tune\_frequency\_value by 0x05 MHz (50 KHz resolution).

**modulation\_value**        When tune\_type = 0x01, modulation value sets the type of modulation for the inband tuner.  
                               00 QAM-64  
                               01 QAM-256  
                               02-0xFF Reserved

### 6.10.6 inband\_tune\_cnf

When the Host receives an inband\_tune\_req APDU, it SHALL respond with the following APDU.

**Table 58 - inband\_tune\_cnf() APDU Syntax**

Syntax	No. of Bits	Mnemonic
inband_tuning_cnf() { inband_tuning_cnf_tag length_field() ltsid tune_status }	24  8 8	uimsbf  uimsbf uimsbf

**inband\_tuning\_cnf\_tag**      0x9F8409

<b>itsid</b>	Local Transport Stream ID
<b>tune_status</b>	The Host's response to the inband_tuning_req APDU. 00 Tuning accepted 01 Invalid frequency (Host does not support this frequency) 02 Invalid modulation (Host does not support this modulation type) 03 Hardware failure (Host has hardware failure) 04 Tuner busy (Host is not relinquishing control of inband tuner) 05-0xFF Reserved

## 6.11 System Time

The system time resource is supplied by the Host and only one session is supported. The CableCARD device creates a session to the resource and then inquires the current time with a System Time Inquiry APDU. If response\_interval is zero, the response is a single system\_time APDU immediately. If response\_interval is non-zero, the response is a system\_time APDU, immediately followed by further system\_time APDUs, every response\_interval seconds. This resource has been modified from the one in [NRSSB]. The Type value has been changed to indicate the modified resource.

**Table 59 - System Time Support Resource**

Resource	Class	Type	Version	Identifier
System Time	36	2	1	00240041

The System Time resource consists of 2 APDUs.

**Table 60 - System Time Support APDUs**

APDU Name	Tag Value	Resource	Direction Host ↔ CableCARD
system_time_inq	0x9F8442	System Time	←
system_time	0x9F8443	System Time	→

### 6.11.1 system\_time\_inq

The CableCARD device transmits the system\_time\_inq to the Host.

**Table 61 - Transmission of system\_time\_inq**

Syntax	No. of Bits	Mnemonic
system_time_inq () { system_time_inq_tag length_field() response_interval }	24	uimsbf
	8	uimsbf

**system\_time\_inq\_tag** 0x9F8442

**response\_interval** How often, in seconds, the Host SHOULD transmit the system\_time APDU to the CableCARD device, starting immediately. A value of 0x00 means that only a single system\_time APDU is to be transmitted, immediately.

### 6.11.2 system\_time

The system\_time APDU is transmitted from the Host to the CableCARD device in response to the system\_time\_inq APDU and then, for non-zero response\_interval values, every response\_interval seconds.

**Table 62 - system\_time APDU**

Syntax	No. of Bits	Mnemonic
system_time() {		
system_time_tag	24	uimsbf
length_field()		
system_time_value	32	uimsbf
GPS.UTC_offset	8	uimsbf
DST_Info	2	Uimsbf
DS_Status	1	Uimsbf
DS_day	5	Uimsbf
DS_hour	8	uimsbf
}		

**system\_time\_tag** 0x9F8443

**system\_time\_value** A 32-bit unsigned integer quantity representing the current system time as the number of seconds since 12 AM, January 6, 1980.

**GPS.UTC\_offset** An 8-bit unsigned integer that defines the current offset in whole seconds between GPS and UTC time standards. To convert GPS time to UTC, the GPS.UTC\_offset is subtracted from GPS time. Whenever the International Bureau of Weights and Measures decides that the current offset is too far in error, an additional leap second MAY be added (or subtracted), and the GPS.UTC\_offset will reflect that change.

**DST\_Info** Indicates the status of the following Daylight Savings information:  
 '00' – None of the following DS information is valid  
 '01' – The DS\_Status flag is valid but the DS\_day and DS\_hour is not.  
 '11' – The DS\_Status, DS\_day and DS\_hour information are all valid.

**DS\_Status** Indicates the status of Daylight Savings:

'0' – Not in daylight savings time

'1' – In daylight savings time

**DS\_day** The local day of the month on which the transition to/from DST occurs (1-31)

**DS\_hour** The local hour at which transition to/from DST occurs (0-23)

## 6.12 Man-Machine Interface (MMI)

The Man-Machine Interface (MMI) resource resides in the Host. The CableCARD device SHALL only open one session to this resource if it wants to initialize one or more MMI dialogs. This session SHALL remain open during normal operation.

The MMI resource provides the following:

- Support to the CableCARD device to open an MMI dialog
- Support to the Host to confirm that the MMI dialog has been opened
- Support to the CableCARD device to close the MMI dialog it opened

- Support to the Host to confirm that the MMI dialog has been closed either upon Host or CableCARD device request

The MMI resource has been changed to type 3 to reflect the changes listed into this section compared to [SCTE28].

The MMI resource has been updated to allow for implementing on Multi-Stream Hosts, using the transport stream ID. This allows for the CableCARD device to determine which transport stream SHOULD display the desired MMI dialog. With the addition of this new functionality, there are circumstances in which the CableCARD device MAY choose to display the MMI screen on 1) all of the Host outputs, 2) only non-recorded Host outputs, 3) the display (for Host with displays, i.e., televisions), and 4) specific transport streams.

**Table 63 - MMI Support Resource**

Resource	Class	Type	Version	Identifier
MMI	64	3	1	0x000400C1

The MMI resource consists of 4 APDUs.

**Table 64 - MMI Support APDUs**

APDU Name	Tag Value	Resource	Direction Host ↔ CableCARD
open_mmi_req()	0x9F8820	MMI	←
open_mmi_cnf()	0x9F8821	MMI	→
close_mmi_req()	0x9F8822	MMI	←
close_mmi_cnf()	0x9F8823	MMI	→

### 6.12.1 open\_mmi\_req

The CableCARD device SHALL send an open\_mmi\_req() APDU to the Host when it wants to initialize an MMI dialog. For a Host that supports more than one MMI dialog at the same time (multiple windows), the CableCARD device MAY send another open\_mmi\_req APDU before it closes the previous one.

**Table 65 - open\_mmi\_req()**

Syntax	No. of Bits	Mnemonic
open_mmi_req() { open_mmi_req_tag length_field() display_stream_type if (stream_type == 0x03) { ltsid } display_type url_length for (i=0; i<url_length; i++) { url_byte } }	24  8  8  8 16  8	uimsbf  uimsbf  uimsbf  uimsbf uimsbf  uimsbf

**open\_mmi\_req\_tag**                   0x9F8820

<b>display_stream_type</b>	Type of stream display 0x00 – All transport streams 0x01 – Only on non-recording transport streams 0x02 – Only on display (for Hosts with display capability) 0x03 – Only on a specific transport stream 0x04-0xFF – Reserved
<b>ltsid</b>	Local Transport Stream ID
<b>display_type</b>	Describes how the MMI dialog SHOULD take place. For a Host that supports more than one MMI dialog at the same time, the new MMI dialog can be in the current window or in a new one. Display type implementation is up to the Host manufacturer. One resource class is provided. It supports display and keypad instructions to the user. 00 Full screen 01 Overlay 02 New window 03-0xFF Reserved
<b>url_length</b>	Number of bytes in the following loop.
<b>url_byte</b>	Each url_byte is one octet of a parameter that points to a HTML page in the CableCARD device and that needs to be queried by the Host using the server_query APDU (Application Info resource) when the MMI dialog is opened.

### 6.12.2 open\_mmi\_cnf

After receiving an open\_mmi\_req APDU from the CableCARD device, the Host SHALL reply with an open\_mmi\_cnf APDU to confirm the status of the request.

**Table 66 - open\_mmi\_cnf**

Syntax	No. of Bits	Mnemonic
open_mmi_cnf() { open_mmi_cnf_tag length_field() display_stream_type if (stream_type == 0x03) { ltsid } dialog_number open_status }	24  8  8  8 8	uimsbf  uimsbf  uimsbf  uimsbf uimsbf

<b>open_mmi_cnf_tag</b>	0x9F8821
<b>display_stream_type</b>	Type of stream display 0x00 – All transport streams 0x01 – Only on non-recording transport streams 0x02 – Only on display (for Hosts with display capability) 0x03 – Only on a specific transport stream 0x04-0xFF - Reserved
<b>ltsid</b>	Local Transport Stream ID

**dialog\_number** A number supplied by the Host issued from an 8-bit cyclic counter that uniquely identifies each open\_mmi\_cnf APDU and allows the CableCARD device to close the associated MMI dialog.

**open\_status** The status of the requested MMI dialog defined as follows:

- 00 OK- Dialog opened
- 01 Request denied – Host busy
- 02 Request denied – Display type not supported
- 03 Request denied – No video signal
- 04 Request denied – No more windows available
- 05-0xFF Reserved

### 6.12.3 close\_mmi\_req

The CableCARD device SHALL send a close\_mmi\_req APDU to the Host to close a MMI dialog previously opened with an open\_mmi\_req APDU.

**Table 67 - close\_mmi\_req**

Syntax	No. of Bits	Mnemonic
close_mmi_req() { close_mmi_req_tag	24	uimsbf
length_field() display_stream_type	8	uimsbf
if (stream_type == 0x03) { ltsid	8	uimsbf
} dialog_number	8	uimsbf
}		

**close\_mmi\_req\_tag** 0x9F8822

**display\_stream\_type** Type of stream display

- 0x00 – All transport streams
- 0x01 – Only on non-recording transport streams
- 0x02 – Only on display (for Hosts with display capability)
- 0x03 – Only on a specific transport stream
- 0x04-0xFF - Reserved

**ltsid** Local Transport Stream ID

**dialog\_number** The number of the MMI dialog assigned by the Host in the open\_mmi\_cnf APDU.

### 6.12.4 close\_mmi\_cnf

After receiving a close\_mmi\_req APDU from the CableCARD device, the Host SHALL reply with a close\_mmi\_cnf APDU to confirm the status of the close operation. The Host MAY send a close\_mmi\_cnf APDU without the CableCARD device having sent a close\_mmi\_req APDU to inform the CableCARD device about a close operation performed by the Host (e.g., the subscriber closes the window).

**Table 68 - close\_mmi\_cnf**

Syntax	No. of Bits	Mnemonic
close_mmi_cnf() { close_mmi_cnf_tag length_field() display_stream_type if (stream_type == 0x03) { ltsid } dialog_number }	24  8  8  8	uimsbf  uimsbf  uimsbf  uimsbf

**close\_mmi\_cnf\_tag**           0x9F8823

**display\_stream\_type**       Type of stream display  
                               0x00 – All transport streams  
                               0x01 – Only on non-recording transport streams  
                               0x02 – Only on display (for Hosts with display capability)  
                               0x03 – Only on a specific transport stream  
                               0x04-0xFF - Reserved

**ltsid**                        Local Transport Stream ID

**dialog\_number**            The number of the MMI dialog received in the close\_mmi\_req APDU.

### 6.13 CableCARD Device Capability Discovery

The M-CARD MUST indicate to the Host what its multi-stream capabilities are. It communicates the maximum number of transport streams it supports via the stream\_profile APDU, how many programs via program\_profile APDU, and how many elementary streams via es\_profile APDU. The Host replies in each case with an appropriate confirmation APDU.

The maximum number of elementary streams does not include those PIDs which are consumed internally by the CableCARD Device for internal CableCARD Device applications. This number refers only to PIDs of programs that are consumed by the Host for viewing or storage. If the CableCARD Device requires additional PIDs for internal consumption, they are not to be included in this number.

The CableCARD device SHALL inform the Host which PIDs it requires through the CableCARD Capability Discovery resource, and the Host MUST not remove those PIDs from each transport stream. Since the PIDs that the CableCARD device requires MAY change, the Host MUST be ready to receive this APDU at any time. The CableCARD device MAY request a maximum of 8 PIDs be transmitted to it.

In the event that the CableCARD device is unable to meet the requirements that a user has requested, it SHALL be the Host’s responsibility to notify the user.

The Host is responsible for controlling the data through the M-CARD interface such that the data rate does not exceed the interface maximum. The Host may perform this by removing packets with PID’s that are not used.

**Table 69 - CableCARD Device Resources Resource**

Resource	Class	Type	Version	Identifier
CARD RES	38	3	1	0x002600C1

The CableCARD Resources resource consists of 8 APDUs.

**Table 70 - CableCARD Resources Support APDUs**

APDU Name	Tag Value	Resource	Direction Host ↔ CARD
stream_profile()	0x9FA010	CARD RES	←
stream_profile_cnf()	0x9FA011	CARD RES	→
program_profile()	0x9FA012	CARD RES	←
program_profile_cnf()	0x9FA013	CARD RES	→
es_profile()	0x9FA014	CARD RES	←
es_profile_cnf()	0x9FA015	CARD RES	→
pids_required()	0x9FA016	CARD RES	←
pids_required_cnf()	0x9FA017	CARD RES	→

### 6.13.1 stream\_profile APDU

After the Host Control session is established, the CableCARD device SHALL transmit the stream\_profile APDU to the Host. This includes the maximum number of streams that the CableCARD device can support.

**Table 71 - stream\_profile APDU Syntax**

Syntax	# of bits	Mnemonic
stream_profile() { stream_profile_tag length_field() max_number_of_streams }	24   8	uimsbf   uimsbf

**stream\_profile\_tag** Value = 0x9FA010

**max\_number\_of\_streams** The maximum number of unique MPEG transport streams input into the CableCARD device from the Host that the M-CARD can manage. The value MUST be greater than three.

### 6.13.2 stream\_profile\_cnf APDU

When the Host receives the stream\_profile APDU from the CableCARD device, it will respond with the following APDU.

**Table 72 - stream\_profile\_cnf APDU**

Syntax	# of bits	Mnemonic
stream_profile_cnf() { stream_profile_cnf_tag length_field() number_of_streams_used }	24	uimsbf
	8	uimsbf

**stream\_profile\_cnf\_tag** Value = 0x9FA011

**number\_of\_streams\_used** The number of unique MPEG transport streams that the Host will be sending to the CableCARD device simultaneously.

### 6.13.2.1 program\_profile APDU

After the Host Control session is established, the CableCARD device SHALL transmit the program\_profile APDU to the Host. This includes the maximum number of simultaneous programs, summed across all transport streams, that the CableCARD device's CA system can simultaneously decrypt.

**Table 73 - program\_profile APDU**

Syntax	# of bits	Mnemonic
program_profile() { program_profile_tag length_field() max_number_of_programs }	24	uimsbf
	8	uimsbf

**program\_profile\_tag** Value = 0x9FA012

**max\_number\_of\_programs** The maximum number of programs that the M-CARD's CA system can simultaneously decrypt. Must be greater than or equal to **four**.

### 6.13.2.2 programs\_profile\_cnf APDU

When the Host receives the program\_profile APDU from the CableCARD device, it will respond with the following APDU.

**Table 74 - programs\_profile\_cnf APDU**

Syntax	# of bits	Mnemonic
programs_profile_cnf() { programs_profile_cnf_tag length_field() }	24	uimsbf

**programs\_profile\_cnf\_tag** Value = 0x9FA013

### 6.13.2.3 es\_profile APDU

After the Host Control session is established, the CableCARD device SHALL transmit the es\_profile APDU to the Host. This includes the maximum number of simultaneous elementary streams, summed across all transport streams, that the CableCARD device can support.

This maximum number does not include those PIDs which are consumed internally by the CableCARD device for internal CableCARD device applications. This number refers only to PIDs of programs that are consumed by the Host for viewing or storage. If the CableCARD device requires additional PIDs for internal consumption, they are not to be included in this number.

**Table 75 - es\_profile APDU Syntax**

Syntax	# of bits	Mnemonic
es_profile() { es_profile_tag length_field() max_number_of_es }	24	uimsbf
	8	uimsbf

**es\_profile\_tag** Value = 0x9FA014

**max\_number\_of\_es** The maximum number of elementary streams that the M-CARD can manage. Must be greater than or equal to sixteen (16).

### 6.13.2.4 es\_profile\_cnf APDU

When the Host receives the es\_profile APDU from the CableCARD device, it will respond with the following APDU.

**Table 76 - es\_profile\_cnf APDU Syntax**

Syntax	# of bits	Mnemonic
es_profile_cnf() { es_profile_cnf_tag length_field() }	24	uimsbf

**es\_profile\_cnf\_tag** Value = 0x9FA015

### 6.13.2.5 request\_pids APDU

If the Host has to perform filtering (removing transport packets corresponding to PIDs that it does not need) to keep the bandwidth below the maximum of the interface, the Host SHALL ask the M-CARD for the PID's that it MUST receive and therefore the Host SHOULD NOT filter. The request\_pids() APDU is issued by the CableCARD device to request a list of PIDs that SHOULD NOT be filtered.

**Table 77 - request\_pids APDU**

Syntax	# of bits	Mnemonic
request_pids() { request_pids_tag length_field() ltsid pid_filtering_status }	24   8 8	uimsbf   uimsbf uimsbf

**request\_pids\_tag** Value = 0x9FA016

**ltsid** Local Transport Stream ID

**pid\_filtering\_status**  
00 = Host not filtering PID's  
01 = Host filtering PID's  
02-FF = Reserved

### 6.13.3 request\_pids\_cnf APDU

The CableCARD device SHALL respond to the request PIDs command when the Host is filtering PIDs with the PIDs it requires. The CableCARD device MAY send this APDU at any time. This list is absolute and SHOULD always include all PIDs that the CableCARD device requires for a given transport stream.

**Table 78 - request\_pids\_cnf APDU**

Syntax	# of bits	Mnemonic
request_pids_cnf() { request_pids_cnf_tag length_field() ltsid number_of_pids for (i=0; i<number_of_pids; i++) { zero pid } }	24   8 8  3 13	uimsbf   uimsbf uimsbf  uimsbf uimsbf

**request\_pids\_cnf\_tag** Value = 0x9FA017

**ltsid** Local Transport Stream ID

**number\_of\_pids** Number of PIDs required, maximum is 8

**zero** 3 bits of zero

**pid** PID value

## 6.14 Copy Protection

A Copy Protection Resource SHALL be required. This resource is defined in [OC-MCCP].

## 6.15 Extended Channel Support

For purposes of the Extended Channel, the CableCARD device or the Host that provides the physical communications link to the headend is referred to as the "link device". The CableCARD device is the link device for the QPSK modem, and the Host is the link device for the High Speed Host Modem.

The Extended Channel Support resource SHALL be created to register the applications that expect to send and receive data to and from the Extended Channel.

All Hosts are required to provide the hardware necessary to support a QPSK downstream out-of-band (OOB) channel for the CableCARD device, or to support the DOCSIS Set-top Gateway (DSG) function. The CableCARD device SHALL forward data received on this channel to the Host as appropriate through one or more data flows requested by the Host. In some cases, the CableCARD device will terminate data received on the QPSK downstream OOB channel or the DSG Ethernet tunnel for its own use (e.g. EMMs). In other cases, it MAY perform a filtering function and discard data known to be of no interest to the Host.

Additionally, a Host MAY contain multiple CableCARD devices. When the cable plant uses a legacy OOB method, a single CableCARD device will control the reverse channel transmitter and is called the primary CableCARD device. All other CableCARD devices SHALL NOT control the reverse channel transmitter and are called secondary CableCARD devices. Each CableCARD device will be assigned a CableCARD ID (CableCARD\_id) by the Host with the primary CableCARD device being assigned the value of 0x01. The value of 0x00 SHALL NOT be used. Each secondary CableCARD device will open an extended channel flow to the primary CableCARD device, via the Host. Anytime a secondary CableCARD device needs to transmit data to the headend, it SHALL send this data over the extended channel with the assigned flow ID to the Host. The Host SHALL then send this data to the primary CableCARD device using the same flow\_id. There SHALL be data transfer capability between the primary and secondary CableCARD devices using this mechanism with the Host routing the data. The actual data format is not defined in this specification. Only secondary to primary CableCARD-to-CableCARD data will be supported.

Supported system architectures imply three different ways of using the Extended Channel Support resource:

- The application is in the Host and the data are transferred to/from the headend via the QPSK modem.
- The application is in the CableCARD device and the data are transferred to/from the headend via the Host's High Speed Host Modem.
- The application is in a secondary CableCARD device and the data needs to be transferred to the headend through the primary CableCARD device.

This resource has three versions: version 1 of this resource is required for Hosts that do not have an embedded High Speed Host (DOCSIS) Modem; version 2 of this resource is required for Hosts that do have an embedded High Speed Host (DOCSIS) Modem and support the DSG interface; and version 3 of this resource is required for Hosts that support multiple CableCARD devices. Version 2 of this resource includes support for all of the objects defined by version 1. Version 3 of this resource includes support for all of the objects defined by version 2.

**Table 79 - Extended Channel Support Resource**

Resource	Class	Type	Version	Identifier
Extended Channel Support	160	1	1	0x00A00041
Extended Channel Support	160	1	2	0x00A00042
Extended Channel Support	160	1	3	0x00A00043

Version 1 contains the following APDU's:

**Table 80 - Extended Channel Support APDUs (Version 1)**

APDU Name	Tag Value	Resource	Direction Host ↔ CableCARD	
			Host modem	CableCARD modem
new_flow_req()	0x9F8E00	Extended Channel Support	↔	→
new_flow_cnf()	0x9F8E01	Extended Channel Support	↔	←
delete_flow_req()	0x9F8E02	Extended Channel Support	←	→
delete_flow_cnf()	0x9F8E03	Extended Channel Support	→	←
lost_flow_ind()	0x9F8E04	Extended Channel Support	→	←
lost_flow_cnf()	0x9F8E05	Extended Channel Support	←	→

Version 2 contains the following additional APDUs:

**Table 81 - Extended Channel Support APDUs (Version 2)**

APDU Name	Tag Value	Resource	Direction Host ↔ CableCARD	
			Host modem	CableCARD modem
inquire_DSG_mode()	0x9F8E06	Extended Channel Support	→	→
set_DSG_mode()	0x9F8E07	Extended Channel Support	←	←
DSG_packet_error	0x9F8E08	Extended Channel Support	←	←

Version 3 contains the following additional APDUs:

**Table 82 - Extended Channel Support APDUs (Version 3)**

APDU Name	Tag Value	Resource	Direction Host ↔ CableCARD	
			Host modem	CableCARD modem
inquire_multi_mode()	0x9F8E09	Extended Channel Support	←	←
set_multi_mode()	0x9F8E0A	Extended Channel Support	→	→

### 6.15.1 new\_flow\_req APDU

The application SHALL use the new\_flow\_req APDU to register a new flow with the link device.

For the case of when the DSG mode is selected, even though the Host has the modem, after the CableCARD device has opened the DSG flow, the Host SHOULD open an MPEG section flow to the CableCARD device for reception of the SI data.

The service types available are MPEG section, IP unicast, IP multicast, and DSG.

The CableCARD device is required to support only one outstanding new\_flow\_req APDU at a time. The CableCARD device SHALL send a new\_flow\_cnf APDU with a status field of 0x04 (Network Busy) when additional new\_flow\_req APDUs are received and one is pending.

**MPEG section** - This service type is applicable only for flows between the CableCARD device and the Host. The requested flow SHALL be in the form of MPEG table sections (both long and short form). This type of flow is unidirectional, from CableCARD device to Host only. The value of the section length field in these sections SHALL NOT exceed 4,093 bytes.

When the table section is in long form (as indicated by the section\_syntax\_indicator flag set to "1"), a 32-bit CRC is present. The 32-bit CRC is also present in short-form sections (as indicated by the section\_syntax\_indicator flag set to "0") carried in the SI\_base\_PID (0x1FFC). For these table sections in which an MPEG-2 CRC is known to be present, the CableCARD device SHALL verify the integrity of the table section using the 32-bit CRC at the table section level, or a 32-bit CRC at another protocol layer. Only messages that pass the CRC check SHALL be forwarded to the Host. The CableCARD device SHALL discard table sections that are incomplete or fail the CRC check.

The 32-bit CRC MAY or MAY NOT be present in short-form sections associated with PID values other than the SI\_base\_PID (0x1FFC) and the CableCARD device MAY send these sections to the Host without any checks. In this case, the Host is responsible for validation of these sections.

**IP Unicast** - This service type is applicable both for flows between the CableCARD device, and a modem in the Host (DSG mode), and for the Host, and a modem in the CableCARD device. The requested flow SHALL be in the form of IP packets addressed to or from the modem's IP address. This type of flow MAY be bidirectional. The maximum total length of any IP packet SHALL be 1,500 bytes.

**IP Multicast** - This service type is applicable both for flows between the CableCARD device and a modem in the Host, and for the Host and a modem in the CableCARD device. The requested flow SHALL be in the form of multicast IP packets addressed to the modem's IP address. This type of flow is unidirectional, from network to application only. The maximum total length of any IP packet SHALL be 1,500 bytes.

**DSG** - This service type is only applicable between the Host and the CableCARD device.

Conformance to this specification requires the Host and the CableCARD device to comply with the following requirements:

- The CableCARD device interface SHALL support at least six concurrent MPEG section service\_type flows.
- The CableCARD device interface SHALL support at least one IP unicast service\_type flow.
- The CableCARD device interface SHALL support one DSG service\_type flow.
- If the Service Information Virtual Channel Table indicates that one or more services are defined as being transported out-of-band, the CableCARD device SHALL provide one or more additional flows of the MPEG section service\_type.

- The Host SHALL support one IP unicast service\_flow to the CableCARD device, if requested, when a High Speed Host Modem is available.
- When the Host supports a unicast IP flow, it SHALL use DHCP per RFC 2131 to obtain an IP address for CableCARD device use. The Host SHALL provide the options parameters supplied by the CableCARD device in the new\_flow\_req APDU to build the DHCP message, and add any other options as necessary or desired.

**Table 83 - new\_flow\_req APDU Syntax**

Syntax	No. of Bits	Mnemonic
new_flow_req() {		
new_flow_req_tag	24	uimsbf
length_field()		
service_type	8	uimsbf
if (service_type == 00) { /* MPEG section */		
Reserved	3	bslbf
PID	13	uimsbf
}		
if (service_type == 01) { /* IP unicast */		
MAC_address	48	uimsbf
option_field_length	8	uimsbf
for (i=0; i<option_field_length; i++) {		
option_byte	8	uimsbf
}		
}		
if (service_type == 02) { /* IP multicast */		
Reserved	4	bslbf
multicast_group_ID	28	uimsbf
}		
if (service_type == 04) { /* (CableCARD- CableCARD) */		
CableCARD_id	8	uimsbf
}		

**new\_flow\_req\_tag**           0x9F8E00

**service\_type**            Defines the type of requested service.

- 00 MPEG section
- 01 IP unicast (IP\_U)
- 02 IP multicast (IP\_M)
- 03 DSG
- 04 CableCARD to CableCARD flow
- 05-0xFF Reserved

**PID**                    The 13-bit MPEG-2 Packet Identifier associated with the flow request. The CableCARD device SHALL be responsible for filtering the MPEG-2 transport stream and delivering only MPEG table sections delivered on transport packets with the given value of PID.

**MAC\_address**           The 48-bit MAC address of the entity requesting the unicast IP flow.

**option\_field\_length**    The number of bytes in the following for loop.

**option\_byte**            These bytes correspond to the options field of a DHCP message. One or more DHCP options per [RFC2132] MAY be included. The "end option" (code 255) SHALL NOT be used, so that the entity granting the IP flow request MAY append zero or more additional option fields before delivering the request to the server.

**multicast\_group\_ID** The multicast group ID associated with the flow request. The modem function SHALL be responsible for filtering arriving multicast IP packets and delivering only packets matching the given multicast\_group\_ID address.

**CableCARD\_id** The ID of the secondary CableCARD device requesting a flow to the primary CableCARD device.

Note: If the Host does not support multiple CableCARD devices or the primary CableCARD device requests this, then the Host SHALL respond with the status field = 0x02 request denied, service type not available.

### 6.15.2 new\_flow\_cnf APDU

The link device SHALL return a new\_flow\_cnf APDU after receiving a new\_flow\_req APDU.

The CableCARD device is required to support only one outstanding new\_flow\_req transaction at a time. The CableCARD device SHALL send a new\_flow\_cnf with a status field of 0x04 (Network Busy) when additional new\_flow\_req messages are received and one is pending.

**Table 84 - new\_flow\_cnf APDU Syntax**

Syntax	No. of Bits	Mnemonic
new_flow_cnf() {		
new_flow_cnf_tag	24	uimsbf
length_field()		
status_field	8	uimsbf
flows_remaining	8	uimsbf
if (status_field == 0x00) {		
flow_id	24	uimsbf
service_type	8	uimsbf
if (service_type == IP_U) {		
IP_address	32	uimsbf
Flow_type	8	uimsbf
flags	3	uimsbf
Max_pdu_size	13	uimsbf
option_field_length	8	uimsbf
For (i=0; i<option_field_length; i++) {		
option_byte	8	uimsbf
}		
}		
}		
}		

**new\_flow\_cnf\_tag** 0x9F8E01

**status\_field** Returns the status of the new\_flow\_req.

- 00 Request granted, new flow created
- 01 Request denied, number of flows exceeded
- 02 Request denied, service\_type not available
- 03 Request denied, network unavailable or not responding
- 04 Request denied, network busy
- 05-0xFF Reserved

**flows\_remaining** The number of additional flows of the same service\_type that can be supported. The value 0x00 indicates that no additional flows beyond the one currently requested can be supported.

<b>flow_id</b>	The unique flow identifier for this application's data flow. The flow_id value of 0x000000 is reserved and is not to be assigned.
<b>service_type</b>	The requested service_type received in the new_flow_req APDU.
<b>IP_address</b>	the 32-bit IP address associated with the requested flow.
<b>Flow_type</b>	an 8-bit unsigned integer number that represents the protocol(s) supported by the CableCARD device to establish the IP-U flow. The field has the following values: 0x00 UDP and TCP supported 0x01 UDP only supported 0x02-FF Reserved.
<b>flags</b>	a 3-bit field that contains information, as defined below, pertaining to limitations associated with the interactive network. bit 0 – no_frag bits 2:1 – reserved
<b>no_frag</b>	a 1-bit boolean that designates if the network supports fragmentation. A value of 02 indicates that fragmentation is supported. A value of 12 indicated that fragmentation is not supported.
<b>max_pdu_size</b>	a 13-bit unsigned integer number that designates the maximum PDU length that MAY be transmitted across the interface.
<b>option_field_length</b>	an 8-bit unsigned integer number that represents the number of bytes of option field data to follow.
<b>option_byte</b>	these bytes correspond to the options requested in the new_flow_req() message. The format of the field is as defined in [RFC2132]. The end option (code 255) SHALL NOT be used.

### 6.15.3 delete\_flow\_req APDU

The application SHALL use the delete\_flow\_req APDU to delete a registered data flow.

**Table 85 - delete\_flow\_req APDU Syntax**

Syntax	No. of Bits	Mnemonic
delete_flow_req() { delete_flow_req_tag length_field() flow_id }	24	uimsbf
	24	uimsbf

**delete\_flow\_req\_tag** 0x9F8E02

**flow\_id** The flow identifier for the flow to be deleted.

### 6.15.4 delete\_flow\_cnf APDU

When the link device receives a delete\_flow\_req APDU, it SHALL respond with the delete\_flow\_cnf APDU.

**Table 86 - delete\_flow\_cnf APDU Syntax**

Syntax	No. of Bits	Mnemonic
delete_flow_cnf() { delete_flow_cnf_tag length_field() flow_id }	24	uimsbf
	24	uimsbf

**delete\_flow\_cnf\_tag** 0x9F8E03

**flow\_id** The flow identifier for the flow to be deleted.

**status\_field** Returns the status of the delete\_flow\_req APDU.

- 00 Request granted, flow deleted
- 01 Reserved
- 02 Reserved
- 03 Request denied, network unavailable or not responding
- 04 Request denied, network busy
- 05 Request denied, flow\_id does not exist
- 06 Request denied, not authorized
- 07-0xFF Reserved

### 6.15.5 lost\_flow\_ind APDU

A link device SHALL indicate that a registered data flow has been lost by issuing the lost\_flow\_ind APDU.

**Table 87 - lost\_flow\_ind APDU Syntax**

Syntax	No. of Bits	Mnemonic
lost_flow_ind() { lost_flow_ind_tag length_field() flow_id reason_field }	24	uimsbf
	24	uimsbf
	8	uimsbf

**lost\_flow\_ind\_tag** 0x9F8E04

**flow\_id** The flow identifier for the flow that has been lost.

**reason\_field** Returns the reason the flow was lost.

- 00 Unknown or unspecified reason
- 01 IP address expiration
- 02 Network down or busy
- 03 Lost or revoked authorization
- 04-0xFF Reserved

### 6.15.6 lost\_flow\_cnf APDU

The application SHALL respond with the lost\_flow\_cnf APDU when a lost\_flow\_ind APDU is received.

**Table 88 - lost\_flow\_cnf APDU Syntax**

Syntax	No. of Bits	Mnemonic
lost_flow_cnf() { lost_flow_cnf_tag length_field() flow_id status_field }	24  24 8	uimsbf  uimsbf uimsbf

**lost\_flow\_cnf\_tag**           0x9F8E05

**flow\_id**                    The flow identifier for the flow that has been lost.

**status\_field**               Returns the status of the lost\_flow\_ind APDU.  
00       Indication acknowledged  
01-0xFF Reserved

### 6.15.7 inquire\_DSG\_mode APDU

The Host MAY inquire of the CableCARD device the preferred operational mode for the network, either OOB mode or DSG mode by sending the inquire\_DSG\_mode APDU.

**Table 89 - 01-0xFF Reserved inquire\_DSG\_mode APDU Syntax**

Syntax	No. of Bits	Mnemonic
inquire_DSG_mode() { inquire_DSG_mode_tag length_field() }	24	uimsbf

**inquire\_DSG\_mode\_tag**      0x9F8E06

### 6.15.8 set\_DSG\_mode APDU

The CableCARD device SHALL use the set\_DSG\_mode APDU to inform the Host of the preferred operational mode for the network. This message is sent in response to the inquire\_DSG\_mode APDU, or it MAY be sent as an unsolicited message to the Host after the resource session has been established. The method by which the CableCARD device determines the preferred operational mode is proprietary to the CA/CableCARD system vendor.

A default operational mode MUST be utilized when the Host and/or CableCARD device is unable to obtain the preferred operational mode. There are two potential default conditions that MUST be addressed.

- Either the Host or the CableCARD device MAY NOT support version 2 of the Extended Channel Support resource (inquire\_DSG\_mode and set\_DSG\_mode APDUs).
- The CableCARD device MAY NOT have acquired the preferred operational mode from the network due to possible network errors.

To insure backward compatibility in the first case above, an Multi-Stream Host MUST initialize in the default operational mode of OOB\_mode. In the second case, the CableCARD device SHOULD instruct the Host that the preferred operational mode is OOB\_mode.

If the operational mode is DSG\_mode or DSG\_one-way\_mode, the CableCARD device MAY provide up to eight Ethernet MAC address and number of header bytes to be removed from the DSG tunnel packets. In DSG or one-

way mode, once the DSG extended channel flow has been opened, the Host MUST filter IP packets whose Ethernet destination address match any of the specified DSG\_MAC\_address values, remove the specified number of header bytes from these packets, and transmit them to the CableCARD device over the extended channel, using the flow\_id assigned to the DSG flow.

**Table 90 - set\_DSG\_mode APDU Syntax**

Syntax	No. of Bits	Mnemonic
set_DSG_mode() { set_DSG_mode_tag	24	uimsbf
length_field() operational_mode	8	uimsbf
if ((operation_mode == DSG_mode)    (operation_mode == DSG_one-way_mode)) { number_MAC_addresses	8	uimsbf
for (i=0; i<number_MAC_addresses; i++) { DSG_MAC_address	16	uimsbf
} remove_header_bytes } }		

**set\_DSG\_mode\_tag**           0x9F8E07

**operational\_mode**           Defines the preferred operational mode of the network.

- 00 OOB\_mode – In this mode, the reverse transmitter is under control of the CableCARD device through the use of the OOB\_TX\_tune\_req APDU in the Host Control resource. The Host MUST respond to these messages by tuning the reverse transmitter to the requested frequency and coding value (bit-rate and power level). The CableCARD device uses the OOB-RDC for returning data to the cable headend.
- 01 DSG\_mode – In this mode, the reverse transmitter is under the control of the Host for DOCSIS functionality. If the CableCARD device attempts to command the reverse transmitter with the OOB\_TX\_tune\_req APDU while the Host is operating in DSG mode, the Host will deny the tune request with a Tuning Denied – RF Transmitter Busy status. Also, in this mode, the receiver for the OOB FDC is not active. If the CableCARD device attempts to command this receiver with the OOB\_RX\_tune\_req() message while the Host is operating in the DSG mode, the Host SHALL deny the tune request with a “Tuning Denied – Other Reasons” status.
- 02 DSG\_one-way\_mode – In this mode, the reverse transmitter MUST be disabled for both the OOB channel and the DOCSIS return channel. Also, in this mode, the receiver for the OOB FDC is not active. If the CableCARD device attempts to command this receiver with the OOB\_RX\_tune\_req() message while the Host is operating in the DSG one-way mode, the Host SHALL deny the tune request with a “Tuning Denied – Other Reasons” status. This mode could be used in one-way cable systems and for network diagnosis in two-way cable systems.
- 03-0xFF Reserved

**number\_MAC\_addresses**       The number of DSG MAC addresses allocated by the CableCARD device provider to carry DSG tunnels. A maximum of eight DSG tunnels per PAO module provider are allowed.

**DSG\_MAC\_address**           The Ethernet MAC addresses allocated by the CableCARD device provider to carry the DSG tunnels.

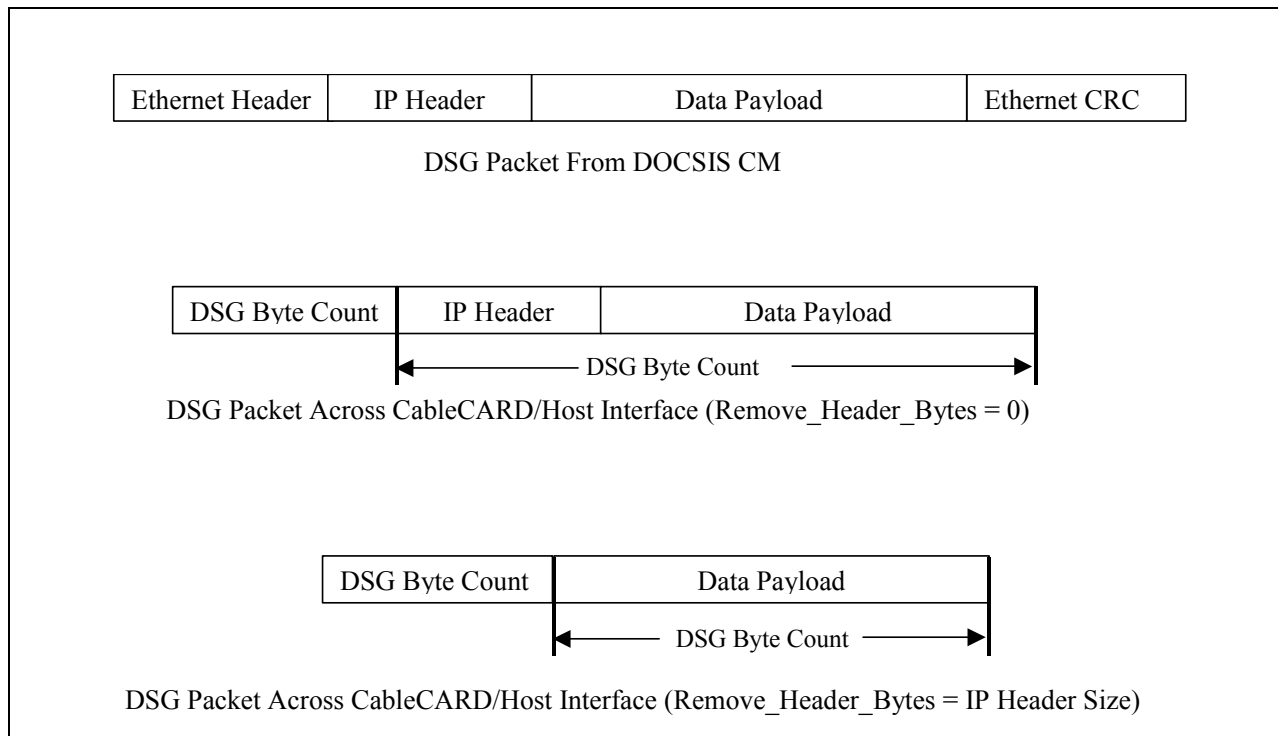
**remove\_header\_bytes** The number of bytes to be removed from the DSG tunnel packets before transmitting the data over the extended channel [see below].

**6.15.8.1 DSG Ethernet Frames**

In DSG mode the downstream communications are implemented in accordance with the DOCSIS Set-top Gateway (DSG) Specification. The upstream Conditional Access Messages and network management messages will be transmitted from the M-CARD Module via IP over the DOCSIS upstream channel using the Extended Channel.

The first two bytes of the downstream DSG frame are the total number of bytes following in the frame, i.e., they do not include this two-byte length field. There is no CRC check required on the frame, as the interface between the Host and CableCARD device is reliable. It is the responsibility of the CableCARD device vendor to implement error detection in the DSG encapsulated data. The CableCARD device should disregard any invalid packets received from the Host. The Host must provide buffer space for a minimum of two DSG IP packets, one for transmission to the CableCARD device and one for receiving from the DOCSIS channel. Informational note: The DSG rate limits the aggregate data rate to 2.048 Mbps to avoid buffer overflow.

Figure 17 below shows how the DSG packets are transported across the CableCARD-Host interface, with and without removal of the IP header bytes. The Ethernet header and CRC are removed, then optionally the IP header is removed, and a two byte field containing the byte count of the resulting data is transmitted across the CableCARD-Host interface.



**Figure 17 - DSG Packet Format Across CableCARD-Host Interface**

**6.15.9 inquire\_multi\_mode APDU**

The CableCARD device MAY inquire to the Host to find out if it is a primary or secondary CableCARD device, and if it is secondary, what is its assigned CableCARD\_id.

**Table 91 - inquire\_multi\_mode APDU Syntax**

Syntax	No. of Bits	Mnemonic
<pre>inquire_multi_mode() {   inquire_multi_mode_tag   length_field() }</pre>	24	uimsbf

**inquire\_multi\_mode\_tag**      0x9F8E09

#### 6.15.10 set\_multi\_mode APDU

In response to the inquire\_multi\_mode APDU, the Host SHALL respond with the following APDU.

**Table 92 - set\_multi\_mode APDU Syntax**

Syntax	No. of Bits	Mnemonic
<pre>set_multi_mode() {   set_multi_mode_tag   length_field()   multi_mode_support   CableCARD_type   CableCARD_id }</pre>	24	uimsbf
	8	uimsbf
	8	uimsbf
	8	uimsbf

**set\_multi\_mode\_tag**      0x9F8E0A

**multi\_mode\_support**      Host support for multiple CableCARD devices  
 0x00 Multiple CableCARD devices not supported, DSG not supported  
 0x01 Multiple CableCARD devices not supported, DSG supported  
 0x02 Multiple CableCARD devices supported, DSG not supported  
 0x03 Multiple CableCARD devices supported, DSG supported  
 0x04-0xff Reserved

**CableCARD\_type**      Type of CableCARD device (NOTE: if multiple CableCARD devices not supported, then this value SHALL be set to 0x00 (primary))  
 0x00 Primary  
 0x01 Secondary  
 0x02-0xff Reserved

**CableCARD\_id**      CableCARD ID value. The primary CableCARD device SHALL always be 0x01.  
 NOTE: 0x00 SHALL NOT be used.

#### 6.15.11 DSG\_packet\_error APDU

The CableCARD device MAY inform the Host of errors that occur in receiving DSG packets using the DSG\_packet\_error APDU.

**Table 93 - DSG\_packet\_error APDU Syntax**

Syntax	No. of Bits	Mnemonic
DSG_packet_error() { DSG_packet_error_tag length_field() error_status }	24	uimsbf
	8	uimsbf

**DSG\_packet\_error\_tag**      0x9F8E08

**error\_status**              Indicates the type of error that occurred  
                                   00 Byte count error – The CableCARD device did not receive the same  
   number of bytes in the DSG packet as was signaled by the Host.  
                                   01-0xFF Reserved

## 6.16 Generic Feature Control

The Generic Feature Control resource enables the Host device to receive control of features which are considered generic to Host devices. There are three aims to this resource: 1) to provide control of features that subscribers do not desire to set themselves, 2) to provide the ability to inhibit subscriber control and only allow headend control, and 3) to provide a mechanism in which a CableCARD device or Host device can be staged to a known value.

A resource is created which resides in the Host called the Generic Feature Control resource. If the Host reports this resource to the CableCARD device, the CableCARD device SHALL open only one session to the Host and SHOULD never close the session.

### 6.16.1 Parameter Storage

#### 6.16.1.1 Host

The Host MAY provide non-volatile storage for the parameters associated with generic features on a parameter-by-parameter basis. These parameters SHALL be stored in the Host.

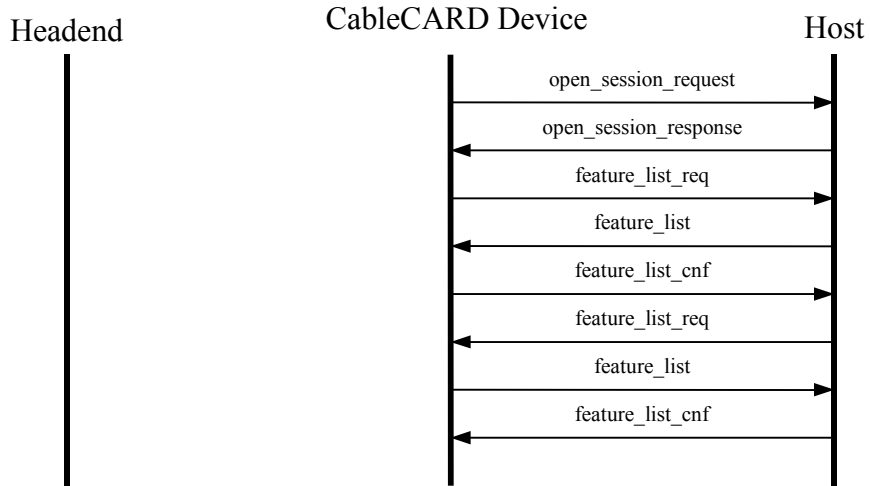
#### 6.16.1.2 CableCARD Device

There is no requirement for the CableCARD device to store the generic feature's parameters although there is no requirement that it cannot.

### 6.16.2 Parameter Operation

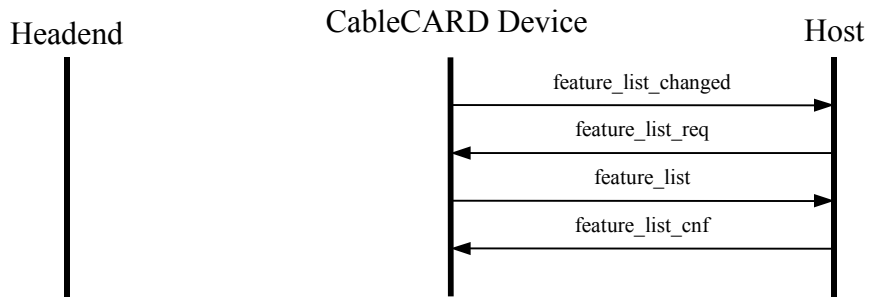
#### 6.16.2.1 Feature List Exchange

Immediately after the session to the Generic Feature Control resource has been established, the CableCARD device SHALL query the Host to determine which generic features are supported in the Host (*feature\_list\_req*). After the CableCARD device receives the generic feature list from the Host (*feature\_list*), the CableCARD device SHALL send its confirmation of the feature list to the Host (*feature\_list\_cnf*). The Host SHALL then query the CableCARD device to determine which generic features are supported in the CableCARD device and the headend (*feature\_list\_req*). The CableCARD device SHALL send its feature list to the Host (*feature\_list*) to which the Host SHALL send its confirmation (*feature\_list\_cnf*). This is called the generic feature list exchange.

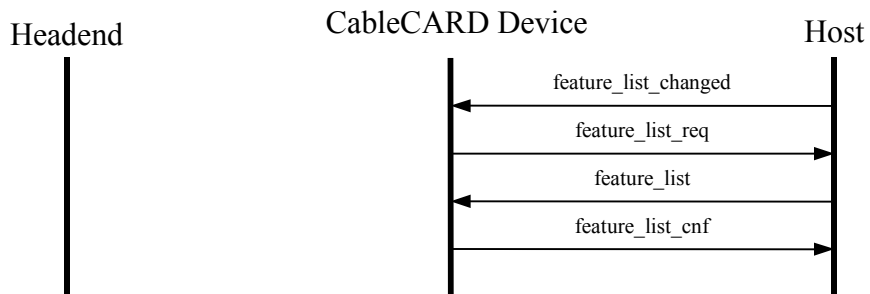


**Figure 18 - Generic Feature List Exchange**

If the generic feature list on the Host or the CableCARD device changes, then the changed device SHALL send a `feature_list_changed` APDU to the other device. The other device SHALL then perform the generic feature list exchange to obtain the new list.



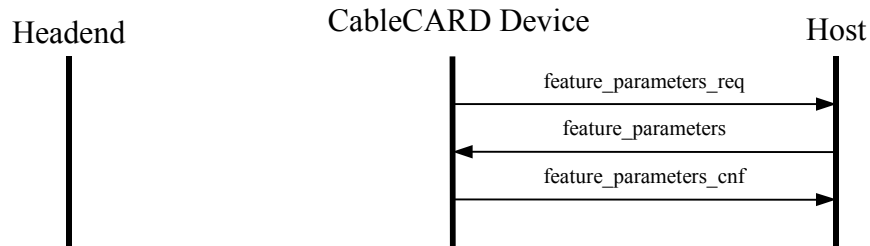
**Figure 19 - CableCARD Device Feature List Change**



**Figure 20 - Host Feature List Change**

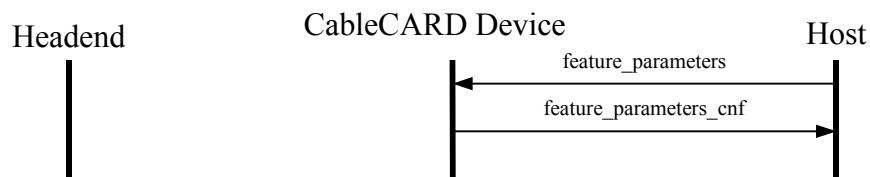
### 6.16.2.2 Host to CableCARD Device Transfer

After the feature exchange has occurred, the CableCARD device MAY request the Host to send its feature parameters (`feature_parameters_req`). After any request, the Host SHALL send to the CableCARD device, the parameters for all the generic features in the Host's generic feature list (`feature_parameters`). The CableCARD device SHALL reply with the confirmation (`feature_parameters_cnf`). The CableCARD device MAY utilize these generic feature parameters, transfer them to the headend, or ignore them.



**Figure 21 - Host to CableCARD Device Feature Parameters**

When any of the parameters of the generic features that are in the CableCARD device generic feature list are changed in the Host, for whatever reason, the Host SHALL transmit these new parameters to the CableCARD device (`feature_parameters`). The CableCARD device SHALL reply with the confirmation (`feature_parameters_cnf`).



**Figure 22 - Host Parameter Update**

The CableCARD device MAY request, at any time the session is open and the generic feature list exchange has occurred, the current parameters in the Host. The CableCARD device SHALL do this by sending a `feature_parameters_request` APDU.

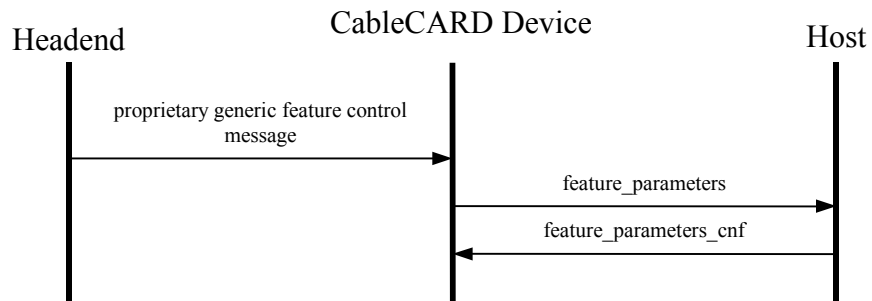
### 6.16.2.3 Headend to Host

It is not intended that the headend would transmit all the generic feature's parameters cyclically. Most of the parameters would only be transmitted once at the request of the user or for staging of the device. The generic feature's parameters which MAY need to be sent cyclically are the RF output channel, time zone, daylight savings, and rating region. The headend MAY send all or just some of the parameters.

The method in which the CableCARD device receives the generic feature's parameters is proprietary to the CableCARD device manufacturer.

After the session has been established, when the CableCARD device receives a message from the headend containing generic feature parameters, the CableCARD device SHALL transfer this information to the Host (`feature_parameters`). The Host SHALL replace its parameters with the values in the APDU. If the CableCARD device utilizes the parameters, it SHALL replace its internal parameters with the values in the message from the headend. The Host SHALL respond with the confirmation (`feature_parameters_cnf`). The Host MAY receive

parameters for generic features which it does not support. The Host SHALL ignore any generic feature parameters that it does not implement.



**Figure 23 - Headend to Host Feature Parameters**

### 6.16.3 Generic Feature Control Resource Identifier

The following resource identifier SHALL be utilized for Generic Feature Control.

**Table 94 - Generic Feature Control Resource**

Resource	Class	Type	Version	Identifier
Generic Feature Control	42	1	1	0x002A0041

### 6.16.4 Feature ID

Each generic feature SHALL have a unique ID assigned to it. This ID is the same for all APDUs. The following is a list of the features and their assigned feature ID.

**Table 95 - Feature IDs**

Feature ID	Feature
0x00	Reserved
0x01	RF Output Channel
0x02	Parental Control PIN
0x03	Parental Control Settings
0x04	IPPV PIN
0x05	Time Zone
0x06	Daylight Savings Control
0x07	AC Outlet
0x08	Language
0x09	Rating Region
0x0A	Reset PINS
0x0B	Cable URL
0x0C	EAS location code
0x0C-0x6F	Reserved for future use
0x70-0xFF	Reserved for proprietary use

### 6.16.5 Generic Feature Control APDUs

The Generic Feature Control resource consists of the following 7 APDUs

**Table 96 - Generic Feature Control APDUs**

APDU Name	Tag Value	Resource	Direction Host ↔ CableCARD
feature_list_req	0x9F9802	Generic Feature Control	↔
feature_list()	0x9F9803	Generic Feature Control	↔
feature_list_cnf()	0x9F9804	Generic Feature Control	↔
feature_list_changed()	0x9F9805	Generic Feature Control	↔
feature_parameters_req()	0x9F9806	Generic Feature Control	←
feature_parameters()	0x9F9807	Generic Feature Control	↔
feature_parameters_cnf()	0x9F9808	Generic Feature Control	↔

#### 6.16.5.1 feature\_list\_req APDU

The Host SHALL send this APDU to the CableCARD device and the CableCARD device SHALL send this APDU to the Host to query the generic features that are supported.

**Table 97 - feature\_list\_req APDU Syntax**

Syntax	No. of Bits	Mnemonic
<pre>feature_list_req() {   feature_list_req_tag   length_field() }</pre>	24	uimsbf

**feature\_list\_req\_tag**            0x9F9802

### 6.16.5.2 feature\_list APDU

After receiving the feature\_list\_req APDU, the Host or CableCARD device SHALL transmit the feature\_list APDU to the CableCARD device or Host, which lists the generic features that are supported by the CableCARD device or Host.

**Table 98 - feature\_list APDU Syntax**

Syntax	No. of Bits	Mnemonic
<pre>feature_list() {   feature_list_tag   length_field()   number_of_features   for (i=0; i&lt;number_of_features; i++) {     feature_id   } }</pre>	24	uimsbf
	8	uimsbf
	8	uimsbf

**feature\_list\_tag**            0x9F9803

**number\_of\_features**        Number of features to report

**feature\_id**                Assigned feature ID number (see Table 95).

### 6.16.5.3 feature\_list\_cnf APDU

After receiving the feature\_list APDU, the Host or CableCARD device SHALL respond with the feature\_list\_cnf APDU.

**Table 99 - feature\_list\_cnf APDU Syntax**

Syntax	No. of Bits	Mnemonic
<pre>feature_list_cnf() {   feature_list_cnf_tag   length_field() }</pre>	24	uimsbf

**feature\_list\_cnf\_tag**        0x9F9804

### 6.16.5.4 feature\_list\_changed APDU

The Host or the CableCARD device SHALL send the feature\_list\_changed APDU to inform the CableCARD device or Host that its feature list has changed.

**Table 100 - feature\_list\_changed APDU Syntax**

Syntax	No. of Bits	Mnemonic
<pre>feature_list_changed() {   feature_list_changed_tag   length_field() }</pre>	24	uimsbf

**feature\_list\_changed\_tag**     0x9F9805

#### 6.16.5.5 feature\_parameters\_req APDU

After the feature list exchange has occurred, the CableCARD device MAY, at any time, send the feature\_parameters\_req APDU to the Host. The Host SHALL NOT send this APDU to the CableCARD device.

**Table 101 - feature\_parameters\_req APDU Syntax**

Syntax	No. of Bits	Mnemonic
<pre>feature_paramters_req() {   feature_paramters_req_tag   length_field() }</pre>	24	uimsbf

**feature\_parameters\_req\_tag**     0x9F9806

#### 6.16.5.6 feature\_parameters APDU

The Host SHALL send the feature\_parameters APDU with its feature list to the CableCARD device after receiving a feature\_parameters\_req APDU, or when any of the parameters in the Host's generic feature list are modified, except if the change is the result of receiving a feature\_parameters APDU from the CableCARD device. The CableCARD device MAY ignore any feature parameters which it does not support.

The CableCARD device MAY send the feature\_parameters APDU at any time in response to a message it receives from the headend.

**Table 102 - feature\_parameters APDU Syntax**

Syntax	No. of Bits	Mnemonic
feature_parameters() { feature_parameters_tag length_field() number_of_features for (i=0; i<number_of_features; i++) { feature_id if (feature_id == 0x01) { rf_output_channel() } if (feature_id == 0x02) { p_c_pin() } if (feature_id == 0x03) { p_c_settings() } if (feature_id == 0x04) { purchase_pin() } if (feature_id == 0x05) { time_zone() } if (feature_id == 0x06) { daylight_savings() } if (feature_id == 0x07) { ac_outlet() } if (feature_id == 0x08) { language() } if (feature_id == 0x09) { rating_region() } if (feature_id == 0x0A) { reset_pin() } if (feature_id == 0x0B) { cable_urls() } if (feature_id == 0x0C) { EA_location_code() } } }	24  8  8	uimsbf  uimsbf  uimsbf

**feature\_parameters\_tag**      0x9F9807

**number\_of\_features**      Number of features to report.

**feature\_id**              Assigned feature ID number

6.16.5.6.1 *rf\_output\_channel*Table 103 - *rf\_output\_channel*

Syntax	No. of Bits	Mnemonic
<code>rf_output_channel() {</code>		
<code>output_channel</code>	8	uimsbf
<code>output_channel_ui</code>	8	uimsbf
<code>}</code>		

**output\_channel** RF output channel. The Host SHALL ignore any value that it cannot accommodate and will use its previous value.

**output\_channel\_ui** Enable RF output channel user interface. If disabled, the Host SHALL disable the user from changing the RF output channel.

- 00 Reserved
- 01 Enable RF output channel user interface
- 02 Disable RF output channel user interface
- 03-0xFF Reserved

6.16.5.6.2 *p\_c\_pin*Table 104 - *p\_c\_pin*

Syntax	No. of Bits	Mnemonic
<code>p_c_pin () {</code>		
<code>p_c_pin_length</code>	8	uimsbf
<code>for (i=0; i&lt;p_c_pin_length; i++) {</code>		
<code>p_c_pin_chr</code>	8	uimsbf
<code>}</code>		
<code>}</code>		

**p\_c\_pin\_length** Length of the parental control PIN. Maximum length is 255 bytes.

**p\_c\_pin\_chr** Parental control PIN character. The value is coded as defined in [ISO/IEC10646-1]. The first character received is the first character entered by the user.

6.16.5.6.3 *p\_c\_settings*Table 105 - *p\_c\_settings*

Syntax	No. of Bits	Mnemonic
<code>p_c_settings() {</code>		
<code>p_c_factory_reset</code>	8	uimsbf
<code>p_c_channel_count</code>	16	uimsbf
<code>for (i=0; i&lt;p_c_channel_count; i++) {</code>		
<code>reserved</code>	4	'1111'
<code>major_channel_number</code>	10	uimsbf
<code>minor_channel_number</code>	10	uimsbf
<code>}</code>		
<code>}</code>		

<b>p_c_factory_reset</b>	Perform factory reset on parental control feature. 00-0xA6 No factory reset. 0xA7 Perform factory reset. 0xA8-0xFF Reserved
<b>p_c_channel_count</b>	Number of virtual channels to place under parental control
<b>major_channel_number</b>	For two-part channel numbers, this is the major number for a virtual channel to place under parental control. For one-part channel numbers, this is the higher 10 bits of the channel number for a virtual channel to place under parental control. Both two-part and one-part channel numbers SHALL be as defined in [SCTE65].
<b>minor_channel_number</b>	For two-part channel numbers, this is the minor number for a virtual channel to place under parental control. For one-part channel numbers, this is the lower 10 bits of the channel number for a virtual channel to place under parental control. Both two-part and one-part channel numbers SHALL be as defined in [SCTE65].

#### 6.16.5.6.4 purchase\_pin

**Table 106 - purchase\_pin**

Syntax	No. of Bits	Mnemonic
<code>purchase_pin() {   purchase_pin_length   for (i=0; i&lt;purchase_pin_length; i++) {     purchase_pin_chr   } }</code>	8	uimsbf
	8	uimsbf

**purchase\_pin\_length** Length of the Purchase PIN. Maximum length is 255 bytes.

**purchase\_pin\_chr** Purchase PIN character. The value is coded as defined in [ISO/IEC10646-1]. The first character received is the first character entered by the user.

#### 6.16.5.6.5 time\_zone

**Table 107 - time\_zone**

Syntax	No. of Bits	Mnemonic
<code>time_zone() {   time_zone_offset }</code>	16	uimsbf

**time\_zone\_offset** Two's complement integer offset, in number of minutes, from UTC. The value represented SHALL be in the range of -12 to +12 hours. This is intended for systems which cross time zones.

6.16.5.6.6 *daylight\_savings***Table 108 - daylight\_savings**

Syntax	No. of Bits	Mnemonic
<pre>daylight_savings() {     daylight_savings_control }</pre>	8	uimsbf

**daylight\_savings\_control** Daylight savings time control

- 00 Ignore this field
- 01 Do not use daylight savings time
- 02 Use daylight savings
- 03-0xFF Reserved

6.16.5.6.7 *ac\_outlet***Table 109 - ac\_outlet**

Syntax	No. of Bits	Mnemonic
<pre>ac_outlet() {     ac_outlet_control }</pre>	8	uimsbf

**ac\_outlet\_control** AC outlet control

- 00 Use user setting
- 01 Switched AC outlet
- 02 Unswitched AC outlet (always on)
- 03 0x-0xFF Reserved
- 04

6.16.5.6.8 *language***Table 110 - language**

Syntax	No. of Bits	Mnemonic
<pre>language() {     language_control }</pre>	24	uimsbf

**language\_control** ISO 639-1: 2002 Codes for the representation of names of Languages - Part 1: Alpha-2 code [ISO696-1], and ISO 639-2: 2002 Codes for the representation of names of Languages - Part 1: Alpha-3 code [ISO639-2].

6.16.5.6.9 *rating\_region***Table 111 - rating\_region**

Syntax	No. of Bits	Mnemonic
<pre>rating_region() {     rating_region_setting }</pre>	8	uimsbf

**rating\_region\_setting** The 8-bit unsigned integer defined in [SCTE65] that defines the rating region in which the Host resides.

- 00 Forbidden
- 01 United States (50 states + possessions)
- 02 Canada
- 03-0xFF Reserved

#### 6.16.5.6.10 reset\_pin

**Table 112 - reset\_pin**

Syntax	No. of Bits	Mnemonic
<pre>reset_pin() {   reset_pin_control }</pre>	8	uimsbf

**reset\_pin\_control** Defines the control of resetting PIN(s). The reset value is defined by the manufacturer and is not covered in this document.

- 00 Do not reset any PIN
- 01 Reset parental control PIN
- 02 Reset purchase PIN
- 03 Reset both parental control and purchase PINs
- 04-0xFF Reserved

#### 6.16.5.6.11 cable\_urls

**Table 113 - cable\_urls**

Syntax	No. of Bits	Mnemonic
<pre>cable_urls() {   number_of_urls   for (i=0; i&lt;number_of_urls; i++) {     url_type     url_length     for (j=0; j&lt;url_length; j++) {       url_char     }   } }</pre>	8	uimsbf
	8	uimsbf
	8	uimsbf
	8	uimsbf

**number\_of\_urls** Number of URLs defined. Used in the following for loop:

**url\_type** Type of URL.

- 00 Undefined
- 01 Web portal URL
- 02 EPG URL
- 03 VOD URL
- 04-0xFF Reserved

**url\_length** Length of the URL. Used in the following for loop. The maximum length is 255 bytes.

**url\_char** A URL character. The restricted set of characters and generic syntax defined in RFC 2396, August 1998, T. Berners-Lee, R. Fielding, and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax" [RFC2396] SHALL be used.

#### 6.16.5.6.12 EA\_location\_code

**Table 114 - EA\_location\_code**

Syntax	No. of Bits	Mnemonic
EA_location_code() { state_code county_subdivision reserved county_code }	8 4 2 10	uimsbf uimsbf '11' uimsbf

**state\_code** As defined in [J042].

**county\_subdivision** As defined in [J042].

**county\_code** As defined in [J042].

#### 6.16.5.7 feature\_parameters\_cnf APDU

When the Host or the CableCARD device receives a feature\_parameter APDU, it SHALL respond with the feature parameters confirmation APDU.

**Table 115 - feature\_parameters\_cnf APDU Syntax**

Syntax	No. of Bits	Mnemonic
feature_paramters_cnf() { feature_paramters_cnf_tag length_field() number_of_features for (i=0; i<number_of_features; i++) { feature_id status } }	24 8 8 8	uimsbf uimsbf uimsbf uimsbf

**feature\_parameters\_cnf\_tag** 0x9F9808

**number\_of\_features** Number of features to report

**feature\_id** Assigned feature ID number

**status** Status of feature parameter.

- 00 Accepted
- 01 Denied - feature not supported
- 02 Denied - invalid parameter
- 03 Denied - other reason
- 04-0xFF Reserved

## 6.17 Generic Diagnostic Support

The Generic Diagnostic Support resource enables the CableCARD device to request that the Host perform a diagnostic and report the status/result of the request to the CableCARD device. The CableCARD device MAY then use the diagnostic information to report diagnostics to the headend or the OSD diagnostic application. If the CableCARD device attempts to open a diagnostic support session, and the Host replies that the Generic Diagnostic Support is unavailable, the CableCARD device SHALL NOT request any diagnostic information from the Host.

The CableCARD device MAY request that the Host perform a diagnostic and report the status/result in response to a headend OOB message, or a SNMP message request to perform a diagnostic that is supported exclusively on the Host.

This resource has been modified from the version in [SCTE28]. Version 2 of this Resource allows for receiving transport stream information from a specific transport stream identified by the Local Transport Stream Identifier (LTSID).

If a Host does support Version 1 of this APDU, the information returned SHALL only be for the primary transport stream (LTSID = 0x01).

**Table 116 - Generic Diagnostic Support Resource**

Resource	Class	Type	Version	Identifier
Generic Diagnostic Support	260	1	2	0x01040042

The Generic Diagnostic Support resource is made up of the following 2 APDUs.

**Table 117 - Generic Diagnostic Support APDUs**

APDU Name	Tag Value	Resource	Direction Host ↔ CableCARD
diagnostic_req()	0x9FDF00	Generic Diagnostic Support	←
diagnostic_cnf()	0x9FDF01	Generic Diagnostic Support	→

The following values SHALL be used as the diagnostic\_id.

**Table 118 - Diagnostic IDs**

Diagnostic	Value
Host memory allocation	0x00
Application version number	0x01
Firmware version	0x02
MAC address	0x03
FAT status	0x04
FDC status	0x05
Current Channel Report	0x06
1394 Port	0x07
DVI_status	0x08

Diagnostic	Value
Reserved	0x09-0xFF

### 6.17.1 diagnostic\_req APDU

The CableCARD device's diagnostic application SHALL use the diagnostic\_req APDU to request the Host to perform a specific set of diagnostic functions and report the result/status of the diagnostics to the CableCARD device's diagnostic application.

**Table 119 - diagnostic\_req APDU Syntax**

Syntax	No. of Bits	Mnemonic
diagnostic_req() { diagnostic_req_tag length_field() number_of_diag for (i=0; i<number_of_diag; i++) { diagnostic_id LTSID } }	24  8 8 8	uimsbf  uimsbf uimsbf uimsbf

**diagnostic\_req\_tag**           0x9FDF00

**number\_of\_diag**           This field indicates the total number of self-diagnostics being requested.

**diagnostic\_id**           This field is a unique ID assigned to a particular diagnostic. These values are defined in Table 96.

**LTSID**                   Local Transport Stream ID.

### 6.17.2 diagnostic\_cnf APDU

The Host SHALL transmit the diagnostic\_cnf APDU after reception of the diagnostic\_req APDU and the Host has completed any tests required to report the result/status.

**Table 120 - diagnostic\_cnf APDU Syntax**

Syntax	No. of Bits	Mnemonic
diagnostic_cnf() { diagnostic_cnf_tag length_field() number_of_diag for (i=0; i<number_of_diag; i++) { diagnostic_id LTSID status_field if (status_field == 0x00) { if (diagnostic_id == 0x00) { memory_report() } if (diagnostic_id == 0x01) { software_ver_report() } if (diagnostic_id == 0x02) { firmware_ver_report() } if (diagnostic_id == 0x03) { MAC_address_report() } if (diagnostic_id == 0x04) { FAT_status_report() } if (diagnostic_id == 0x05) { FDC_status_report() } if (diagnostic_id == 0x06) { current_channel_report() } if (diagnostic_id == 0x07) { 1394_port_report() } if (diagnostic_id == 0x08) { DVI_status() } } } }	24  8  8 8 8	uimsbf  uimsbf uimsbf uimsbf

**diagnostic\_cnf\_tag**           0x9FDF01

**number\_of\_diag**           This field indicates the total number of self-diagnostics being requested.

**diagnostic\_id**            This field is a unique ID assigned to a particular diagnostic. These values are defined in Table 96.

**LTSID**                    Local Transport Stream ID.

**status\_field**            Status of the requested diagnostic.

**Table 121 - Table Status Field Values**

Bit Value (Hex)	Status_field
00	Diagnostic granted
01	Diagnostic Denied - Feature not Implemented.
02	Diagnostic Denied - Device Busy
03	Diagnostic Denied - Other reasons
04-FF	Reserved for future use.

For Diagnostic\_id values from 09 to FF, a Status\_field value of 01 SHALL be returned.

### 6.17.3 Diagnostic Report Definition

Each applicable diagnostic SHALL consist of a set of diagnostic reports that SHALL contain a specific set of parameters applicable to the requested diagnostic. The following sections define these reports and their associated parameters.

#### 6.17.3.1 memory\_report

Memory reports SHALL contain the memory parameters associated with the Host.

**Table 122 - memory\_report**

Syntax	No. of Bits	Mnemonic
memory_report() { number_of_memory if (i=0; i<number_of_memory; i++) { memory_type memory_size } }	8	uimsbf
	8	uimsbf
	32	uimsbf

**number\_of\_memory**     The number of memory types being reported in this message.

**memory\_type**         Designates the type of memory that is being reported.

- 00 ROM
- 01 DRAM
- 02 SRAM
- 03 Flash
- 04 NVM
- 05 Hard drive
- 06 Video memory
- 07 Other memory
- 0x-0xFF Reserved

**memory\_size**         Designates the physical size of the specified memory type. The units are kilobytes, defined to be 1,024 bytes.

**6.17.3.2 software\_ver\_report**

Software version reports SHALL contain the software version parameters associated with the Host.

**Table 123 - software\_ver\_report**

Syntax	No. of Bits	Mnemonic
software_ver_report() {		
number_of_applications	8	uimsbf
for (i=0; i<number_of_applications; i++) {		
application_version_number	16	uimsbf
application_status_flag	8	uimsbf
application_name_length	8	uimsbf
for (j=0; j<application_name_length; j++) {		
application_name_byte	8	uimsbf
}		
application_sign_length	8	uimsbf
for (j=0; j<application_sign_length) j++) {		
application_sign_byte	8	uimsbf
}		
}		
}		

<b>number_of_applications</b>	Total number of applications contained with the report.
<b>application_version_number</b>	16-bit version number of the application.
<b>application_status_flag</b>	Status of the software, either active, inactive or downloading. 00 Active 01 Inactive 02 Downloading 03-0xFF Reserved
<b>application_name_length</b>	Designates the number of characters required to define the applications name.
<b>application_name_byte</b>	ASCII character, 8-bits per character, a string that identifies the application.
<b>application_sign_length</b>	Designates the number of characters required to define the application signature.
<b>application_sign_byte</b>	ASCII character, 8-bits per character, a string that identifies the application signature.

### 6.17.3.3 *firmware\_ver\_report*

Firmware version reports SHALL contain the firmware version parameters associated with the Host.

**Table 124 - *firmware\_ver\_report***

Syntax	No. of Bits	Mnemonic
<code>firmware_ver_report() {</code>		
<code>firmware_version</code>	16	uimsbf
<code>firmware_date{</code>	32	uimsbf
<code>firmware_year</code>	16	uimsbf
<code>firmware_month</code>	8	uimsbf
<code>firmware_day</code>	8	uimsbf
<code>}</code>		

**firmware\_version** 16-bit version number of the firmware.

**firmware\_date** 32-bit numerical representation, in the form of YYYYMMDD, which identifies the date of the firmware.

**firmware\_year** 16-bit designation of the firmware's year.

**firmware\_month** 8-bit numerical representation of the firmware's month.

**firmware\_day** 8-bit numerical representation of the firmware's day.

### 6.17.3.4 *MAC\_address\_report*

The MAC address report SHALL contain the MAC address parameters associated with the Host.

**Table 125 - *MAC\_address\_report***

Syntax	No. of Bits	Mnemonic
<code>MAC_address_report() {</code>		
<code>number_of_addresses</code>	8	uimsbf
<code>for (i=0; i&lt;number_of_addresses; i++) {</code>		
<code>MAC_address_type</code>	8	uimsbf
<code>number_of_bytes</code>	8	uimsbf
<code>for (j=0; j&lt;number_of_bytes; j++) {</code>		
<code>MAC_address_byte</code>	8	uimsbf
<code>}</code>		
<code>}</code>		
<code>}</code>		

**number\_of\_addresses** Total number of MAC addresses contained in the report.

**MAC\_address\_type** Type of device associated with reported MAC address.

00 No addressable device available

01 Host

02 1394 port

03 USB port

04 DOCSIS

05 Ethernet port

06-0xFF Reserved

**number\_of\_bytes** The total number of bytes required for the MAC address.

**MAC\_address\_byte** One of a number of bytes that constitute the Media Access Control (MAC) address of the Host device. Each byte represents 2 hexadecimal values (xx) in the range of 0x00 to 0xFF.

### 6.17.3.5 FAT\_status\_report

In response to a FAT status report request, the Host SHALL reply with a FAT\_status\_report, unless an error has occurred.

**Table 126 - FAT\_status\_report**

Syntax	No. of Bits	Mnemonic
FAT_status_report() {		
reserved	4	'1111'
PCR_lock	1	bslbf
modulation_mode	2	bslbf
carrier_lock_status	1	bslbf
SNR	16	simsbf
signal_level	16	simsbf
}		

**PCR\_lock** Indicates if the FAT channel receiver is locked to the currently tuned channel  
 0 Not locked  
 1 Locked

**modulation\_mode** Indicates if the current forward transport is analog, QAM-64, or QAM-256  
 00 Analog  
 01 QAM-64  
 10 QAM-256  
 11 Reserved

**carrier\_lock\_status** Indicates if the current carrier is locked or not locked.  
 0 Not locked  
 1 Locked

**SNR** Numerical representation of the signal to noise ration in tenths of a dB.

**signal\_level** Numerical representation of the signal level in tenths of a dBmV.

### 6.17.3.6 FDC\_status\_report

In response to a FDC status report request, the Host SHALL reply with a FDC status report, unless an error has occurred.

**Table 127 - FDC\_status\_report**

Syntax	No. of Bits	Mnemonic
FDC_report() { FDC_center_freq reserved carrier_lock_status carrier_lock_status }	16 6 1 1	uimsbf '111111' bslbf bslbf

**FDC\_center\_freq** Indicates the frequency of the FDC center frequency, in MHz. (Frequency = value \* 0.05 + 50 MHz).

**carrier\_lock\_status** Indicates if the current carrier is locked or not locked.  
0 Not locked  
1 Locked

**packet\_sync\_status** Indicates if the current FDC packets are in sync.  
0 Not in sync  
1 In sync

### 6.17.3.7 current\_channel\_report

In response to a Current Channel report request, the Host SHALL reply with a current\_channel report, unless an error has occurred.

**Table 128 - current\_channel\_report**

Syntax	No. of Bits	Mnemonic
current_channel() { reserved channel_type authorization_flag purchasable_flag purchased_flag preview_flag parental_control_flag current_channel }	2 1 1 1 1 1 1 16	'11' bslbf bslbf bslbf bslbf bslbf bslbf uimsbf

**channel\_type** Indicates if the channel is analog or digital.  
0 Analog  
1 Digital

**authorization\_flag** Indicates if the Host is authorized for the currently tuned channel.  
0 Not authorized  
1 Authorized

**purchasable\_flag** Indicates if the currently tuned channel MAY be purchased.  
0 Not purchasable  
1 Purchasable

**purchased\_flag** Indicates if the currently tuned channel has been purchased.  
0 Not purchased  
1 Purchased

- preview\_flag** Indicates if the currently tuned channel is in preview mode.  
 0 Not in preview mode  
 1 In preview mode
- parental\_control\_flag** Indicates if the currently tuned channel is under parental control.  
 0 Channel is not blocked  
 1 Channel is blocked
- current\_channel** Indicates the numerical representation of the currently tuned channel.

### 6.17.3.8 1394\_port\_report

In response to a 1394 Port report request, the Host SHALL reply with a 1394\_port\_report, unless an error has occurred.

**Table 129 - 1394\_port\_report**

Syntax	No. of Bits	Mnemonic
1394_port_report() {		
reserved	2	\111'
loop_status	1	bslbf
root_status	1	bslbf
cycle_master_status	1	bslbf
port_1_connection_status	1	bslbf
port_2_connection_status	1	bslbf
total_number_of_nodes	16	uimsbf
}		

- loop\_status** Indicates if a loop exists on the 1394 bus.  
 0 No loop exists  
 1 Loop exists
- root\_status** Indicates if the set-top terminal is the root node on the 1394 bus.  
 0 Not root  
 1 Is root
- cycle\_master\_status** Indicates if the set-top terminal is the cycle master node on the 1394 bus.  
 0 Not cycle master  
 1 Is cycle master
- port\_1\_connection\_status** Indicates if port 1 of the 1394 PHY is connected to a 1394 bus.  
 0 Not connected  
 1 Connected
- port\_2\_connection\_status** Indicates if port 2 of the 1394 PHY is connected to a 1394 bus.  
 0 Not connected  
 1 Connected
- total\_number\_of\_nodes** Indicates the total number of nodes connected to the 1394 bus. A maximum of 65,535 nodes MAY exist, excluding the Host (a maximum of 64 notes with a maximum of 1,024).

### 6.17.3.9 DVI Status Report

In response to a DVI Status Report request, the Host SHALL reply with a DVI Status Report, unless an error has occurred.

**Table 130 - DVI Status Report Syntax**

Syntax	# of bits	Mnemonic
DVI_status_report() {		
reserved	3	Bslbf
connection_status	2	Bslbf
host_HDCP_status	1	Bslbf
device_HDCP_status	2	Bslbf
video_format		
{		
horizontal_lines	16	Uimsbf
vertical_lines	16	Uimsbf
scan_rate	8	Uimsbf
aspect_ratio	2	Bslbf
prog_inter_type	1	Bslbf
reserved	5	Bslbf
}		
}		

- reserved** All reserved bits SHALL be set to 1<sub>2</sub>
- connection\_status** Indicates if a connection exists on the DVI port,  
00<sub>2</sub> = no connection exists, 01<sub>2</sub> = device connected – not repeater,  
10<sub>2</sub> = device connected - repeater , 11<sub>2</sub> = reserved
- host\_HDCP\_status** Indicates if HDCP is enabled on the DVI link, 0<sub>2</sub> = not enabled, 1<sub>2</sub> = enabled.
- device\_HDCP\_status** Indicates the connected device's HDCP status (valid only when connection\_status is not equal to 00<sub>2</sub>).  
00<sub>2</sub> = non HDCP device, 01<sub>2</sub> = compliant HDCP device,  
10<sub>2</sub> = revoked HDCP device, 11<sub>2</sub> = reserved
- video\_format** Indicates the current video format utilized on the DVI port as defined in the following fields:
- horizontal\_lines** Indicates the number of horizontal lines associated with the video format on the DVI link.
- vertical\_lines** Indicates the number of vertical lines associated with the video format on the DVI link.
- scan\_rate** Indicates the scan rate associated with the video format on the DVI link.
- aspect\_ratio** Indicates the aspect ratio associated with the video format on the DVI link as defined in the following table:

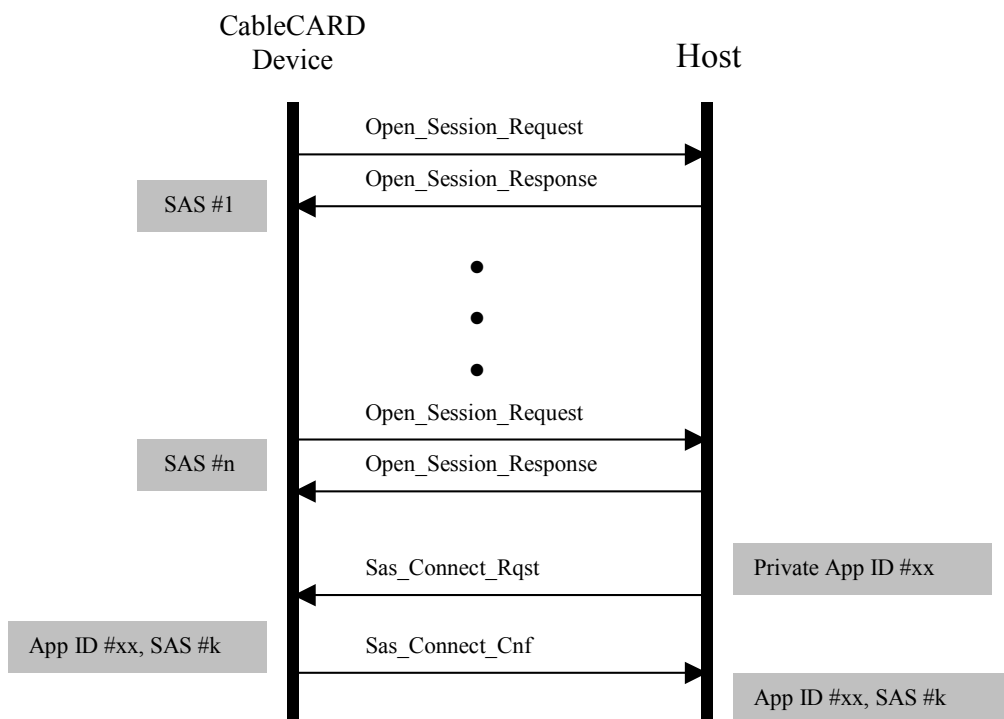
**Table 131 - Aspect Ratio Associated With the Video Format On the DVI Link**

Bit Value (hex)	Video Format
00	4:3
01	16:9
02-FF	Reserved

**prog\_inter\_type** Indicates if the video is progressive or interlaced on the DVI link,  
 0<sub>2</sub> = Interlaced, 1<sub>2</sub> = Progressive

## 6.18 Specific Application Support

The CableCARD device MAY open one or more Specific Application Support (SAS) sessions for private communications between vendor-specific CableCARD device applications and private Host applications. The CableCARD device, as the initiator of the sessions, is responsible for associating each session (by session number) with the appropriate vendor-specific CableCARD device application. When a private Host application is ready to establish a connection with the CableCARD device, a SAS\_connect\_rqst APDU is sent to the CableCARD device over any opened SAS session. The CableCARD device uses the private Host application ID to identify the specific SAS session that SHOULD be used for communication between the identified private Host application and the appropriate vendor-specific CableCARD device application. This session number, along with the private Host application ID, is returned to the Host via the SAS\_connect\_cnf APDU. This operation establishes the communication path between a specific pair of applications (vendor-specific CableCARD device application, private Host application).

**Figure 24 - Specific Application Support Connection Sequence**

In some instances, the CableCARD device MAY receive a SAS\_connect\_rqst APDU before a session has been opened for the associated vendor-specific application. In this case, the CableCARD device SHALL establish the necessary SAS session and then respond with the SAS\_connect\_cnf APDU.

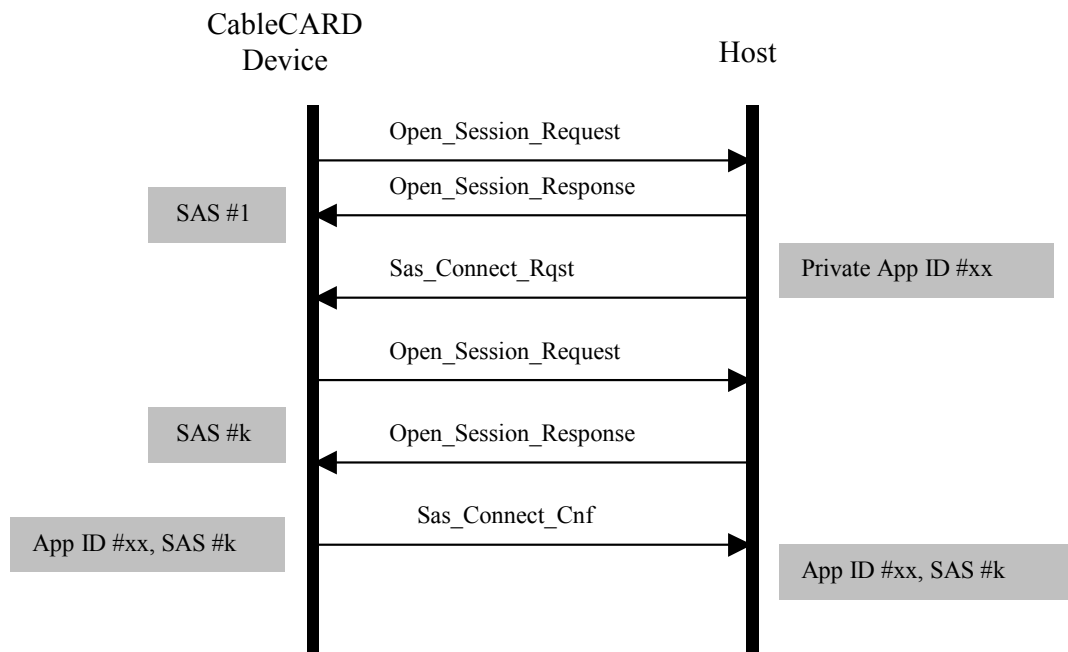


Figure 25 - Specific Application Support Alternate Connection Sequence

Table 132 - Specific Application Support Resource

Resource	Class	Type	Version	Identifier
Specific Application Support	144	1	1	0x00900041

The Specific Application Support resource includes seven APDUs as described in the following table:

Table 133 - Specific Application Support APDUs

APDU Name	Tag Value	Resource	Direction Host ↔ CableCARD
SAS_connect_rqst()	0x9F9A00	Specific Application Support	→
SAS_connect()	0x9F9A01	Specific Application Support	←
SAS_data_rqst()	0x9F9A02	Specific Application Support	↔
SAS_data_av()	0x9F9A03	Specific Application Support	↔
SAS_data_cnf()	0x9F9A04	Specific Application Support	↔
SAS_data_query()	0x9F9A05	Specific Application Support	↔
SAS_data_reply()	0x9F9A06	Specific Application Support	↔

### 6.18.1 SAS\_connect\_rqst APDU

If required by a private Host application, the Host SHALL send a SAS\_connect\_rqst APDU to the CableCARD device to establish a connection between that application and the corresponding CableCARD device vendor-specific application.

**Table 134 - SAS\_connect\_rqst APDU Syntax**

Syntax	No. of Bits	Mnemonic
SAS_connect_rqst () { SAS_connect_rqst_tag length_field() private_host_application_ID }	24	uimsbf
	64	uimsbf

**SAS\_connect\_rqst\_tag**           0x9F9A00

**private\_host\_application\_ID**   This is a unique identifier of the private Host application.  
**NOTE (Informative):** There is no need to register private\_host\_application\_id used by different manufacturers. Applications that make use of this resource are downloaded into the Host by the cable operator, and thus the application has knowledge of valid ID values that are expected from operator-supplied CableCARD devices.

**SAS\_session\_status**           The status of the requested connection.  
 00 Connection established  
 01 Connection denied – no associated vendor-specific CableCARD device application found  
 02 Connection denied – no more connections available  
 03-0xFF Reserved

**SAS\_session\_nb**               The session number to be used for the designated Specific Application communications.

### 6.18.2 SAS\_connect APDU

After receiving the SAS\_connect\_rqst APDU, the CableCARD device SHALL reply with a SAS\_connect\_cnf APDU to inform the Host of which SAS session is to be used for this connection.

**Table 135 - SAS\_connect APDU Syntax**

Syntax	No. of Bits	Mnemonic
SAS_connect_cnf() { SAS_connect_cnf_tag length_field() private_host_application_ID SAS_session_status }	24	uimsbf
	64	uimsbf
	8	uimsbf

**SAS\_connect\_cnf\_tag**           0x9F9A01

**private\_host\_application\_ID**   This is a unique identifier of the private Host application.

Note: There is no need to register private\_host\_application\_id used by different manufacturers. Applications that make use of this resource are downloaded into the Host by the cable operator, and thus the application has knowledge of valid ID values that are expected from operator-supplied CableCARD devices.

**SAS\_session\_status**                      The status of the requested connection.

- 00 Connection established
- 01 Connection denied – no associated vendor-specific CableCARD device application found
- 02 Connection denied – no more connections available
- 03-0xFF Reserved

### 6.18.3 SAS\_data\_rqst APDU

Once a communication path has been established between the application pair (vendor-specific CableCARD device application and private Host application), via a SAS session, each of the applications can utilize the SAS\_data\_rqst APDU to request data from the other. This APDU is bidirectional.

**Table 136 - SAS\_data\_rqst APDU Syntax**

Syntax	No. of Bits	Mnemonic
SAS_data_rqst() { SAS_data_rqst_tag length_field() }	24	uimsbf

**SAS\_data\_rqst\_tag**                      0x9F9A02

### 6.18.4 SAS\_data\_av APDU

Once a communication path has been established between the application pair (vendor-specific CableCARD device application and private Host application) via a SAS session, each of the applications can utilize the SAS\_data\_av APDU to being transmission of data in response to the SAS\_data\_rqst APDU. This APDU MAY be sent independent of receiving a SAS\_data\_rqst APDU.

Note: The data itself is transmitted in the SAS\_query and SAS\_reply APDUs.

**Table 137 - SAS\_data\_av APDU Syntax**

Syntax	No. of Bits	Mnemonic
SAS_data_av() { SAS_data_av_tag length_field() SAS_data_status transaction_nb }	24	uimsbf
	8	uimsbf
	8	uimsbf

**SAS\_data\_av\_tag**                      0x9F9A03

**SAS\_data\_status**                      Status of the available data.

- 00 Data available
- 01 Data not available
- 02-0xFF Reserved

**transaction\_nb** The transaction number is issued from an 8-bit cyclic counter (1-255) and is used to identify each data transaction and to gain access to the available data. When data is not available, the transaction\_nb SHALL be set to 0x00.

### 6.18.5 SAS\_data\_cnf APDU

Once a communication path has been established between the application pair (vendor-specific CableCARD device application and private Host application), via a SAS session, after an application receives an SAS\_data\_av APDU, the application SHALL transmit the SAS\_data\_cnf APDU to acknowledge that it is ready to receive the available data.

**Table 138 - SAS\_data\_cnf APDU Syntax**

Syntax	No. of Bits	Mnemonic
SAS_data_av_cnf() { SAS_data_av_cnf_tag length_field() transaction_nb }	24	uimsbf
	8	uimsbf

**SAS\_data\_av\_cnf\_tag** 0x9F9A04

**transaction\_nb** The transaction number is issued from an 8-bit cyclic counter (1-255) and is used to identify each data transaction and to gain access to the available data. When data is not available, the transaction\_nb SHALL be set to 0x00.

### 6.18.6 SAS\_server\_query APDU

When data availability has been confirmed, a SAS\_server\_query APDU is sent to initiate the transfer of application specific data.

**Table 139 - SAS\_server\_query APDU Syntax**

Syntax	No. of Bits	Mnemonic
SAS_server_query () { SAS_server_query_tag length_field() transaction_nb }	24	uimsbf
	8	uimsbf

**SAS\_server\_query\_tag** 0x9F9A05

**transaction\_nb** The transaction\_nb assigned in the SAS\_data\_av APDU.

### 6.18.7 SAS\_server\_reply APDU

After receiving the SAS\_server\_query APDU, the application SHALL respond with the SAS\_server\_reply APDU with the data to transfer.

**Table 140 - SAS\_server\_reply APDU Syntax**

Syntax	No. of Bits	Mnemonic
SAS_server_reply() {		
SAS_server_reply_tag	24	uimsbf
length_field()		
transaction_nb	8	uimsbf
message_length	16	uimsbf
for (i=0; i<message_length; i++) {		
message_byte	8	uimsbf
}		
}		

**SAS\_server\_reply\_tag**                   0x9F9A06

**transaction\_nb**                        The transaction\_nb assigned in the SAS\_data\_av APDU.

**message\_length**                        The length of the message in the following for loop.

**message\_byte**                         The data to transfer.

## 6.19 CableCARD Device Firmware Upgrade

The M-CARD SHALL support firmware upgrades as defined by section 8.13 of [SCTE28].

## 6.20 Support for Common Download Specification

The M-CARD SHALL support common download as defined in section 8.15 of [SCTE28].

## 7 EXTENDED CHANNEL OPERATION (NORMATIVE)

The extended channel provides a data path between the CableCARD device and the Host. It is implemented as a separate endpoint from the command channel, and in the same manner as described in [SCTE28].

### 7.1 Link Layer

The link layer of the Extended Channel fragments the datagram PDU, if necessary, over the limited buffer size of the physical layer, and reassembles the received fragments.

The link header includes two control bits and the flow\_id value that has been negotiated by the link device for the application (see section 6.15), to identify the end-to-end communication flow.

**Table 141 - Extended Channel Link Layer Packet**

Bit							
7	6	5	4	3	2	1	0
L	F	0x00					
flow_id (MSB)							
flow_id							
flow_id (LSB)							
datagram PDU fragment							

**L** Last indicator: if this bit is set to '0', then at least one more datagram fragment follows. If this bit is set to '1', this fragment is the last in the datagram.

**F** First fragment indicator: if this bit is set to '1', then this fragment is the first of the datagram. If this bit is set to '0', this fragment is not the first.

**flow\_id** The 3-byte flow identifier associates the data with a registered flow. The flow\_id is assigned as defined in section 6.15. The flow\_id value of zero is reserved and is not to be assigned.

For data flows made available to the Host by the CableCARD device, the CableCARD device is responsible for link layer processing of messages to be transferred across the Extended Channel. It is the Host's responsibility to reassemble the received datagram PDU fragments, and to segment PDUs for delivery across the interface. For data flows made available to the CableCARD device by the Host, the roles are reversed.

Received datagram PDU fragments SHALL be reassembled into IP packets, or MPEG-2 table sections, or DSG messages, depending upon the service\_type associated with the flow given by flow\_id. The maximum size of the reassembled PDU (IP packet or MPEG-2 table section or DSG message) SHALL be 4,096 for any Service Type.

#### 7.1.1 Maximum PDUs

The maximum size of any PDU before fragmentation SHALL be 4,096 bytes for downstream data for any Service Type. The maximum size of any PDU before fragmentation SHALL be 1,500 bytes for upstream data for any Service Type.

## 7.2 Modem Models

There are 3 different network connection models that a Host MAY have:

- Unidirectional (no modem)
- Bidirectional, modem function in the CableCARD device
- Bidirectional, modem function in the Host

### 7.2.1 Unidirectional Host Model

For the unidirectional Host model, there is no IP connectivity. The extended channel will be utilized solely for receiving the OOB SI data.

For this model, the CableCARD device will be the link device for the OOB SI MPEG data flow.

### 7.2.2 Bidirectional With Modem in CableCARD Device

For the bidirectional Host model with the modem functionality in the CableCARD device, the SCTE 55-2 and the SCTE 55-1 PHY (RF processing, QPSK demodulation and modulation) layer is implemented in the Host, and the Data-link and MAC protocols are implemented in the CableCARD device. The details of the OOB hardware implementation are covered in Section 4.1.6 of this specification.

For this model, the CableCARD device will be the link device for all flows.

### 7.2.3 Bidirectional With Modem in Host

When the Host implements the DSG functionality, all of the DOCSIS cable modem functionality are implemented in the Host. The OOB data flows are transmitted, utilizing Ethernet tunneling [DSG]. The Host MUST be capable of receiving the DSG OOB data flows even if it is unable to connect in two-way mode.

For this model, the CableCARD device will be the link device for the OOB MPEG SI data flow, and the Host will be link device for IP data flows and for the DSG flow, to the CableCARD device.

## 7.3 SI Requirements

OOB SI data is transmitted from the CableCARD device to the Host using the protocols defined in [SCTE65]. The CableCARD device SHALL always be the source of all OOB SI data to the Host. The Host SHALL NOT use OOB SI data received by any other method. This is due to the fact that the CableCARD device MAY reformat the OOB SI data to meet the requirements of [SCTE65], as there is no requirement for the cable system to transmit the OOB SI data in that format.

A Host MAY use data received on OOB data streams that is outside of the [SCTE65] protocols.

## 7.4 EAS Requirements

EAS messages are transmitted from the CableCARD device to the Host using the protocol defined in [J042]. When a CableCARD device is installed in the Host, the CableCARD device SHALL always be the source of these messages. The Host SHALL NOT use OOB EAS messages received by any other method. The CableCARD device MAY reformat the OOB EAS message to meet the requirements of [J042], as there is no requirement for the cable system to transmit the OOB EAS message in that format.

Note: EAS operation for when a Host does not have a CableCARD device installed, is outside the scope of this specification.

## APPENDIX A REFERENCE TABLES

### A.1 Multi-Stream CableCARD Device Resource Identifiers

Resource	Class	Type	Version	Resource Identifier
Resource Manager	1	1	1	0x00010041
Application Information	2	2	1	0x00020081
Conditional Access Support	3	2	2	0x00030082
Host Control	32	2	1	0x00200081
System Time	36	1	1	0x00240041
MMI	64	3	1	0x000400C1
Homing	17	1	2	0x00110042
Copy Protection	176	4	1	0x00B00101
Specific Application Support	144	1	1	0x00900041
Generic Feature Control	42	1	1	0x002A0041
Extended Channel Support	160	1	2	0x00A00042
Generic Diagnostic Support	260	2	1	0x010400081
System Control	43	1	1	0x002B0041
CARD Capability Discovery	38	3	1	0x002600C1

### A.2 Application Object Tag Values

apdu_tag	tag value	Resource	Direction Host ↔ CableCARD
profile_inq	0x9F8010	Resource Manager	←
profile_reply	0x9F8011	Resource Manager	→
profile_changed	0x9F8012	Resource Manager	→
application_info_req	0x9F8020	Application Info	→
application_info_cnf	0x9F8021	Application Info	←
server_query	0x9F8022	Application Info	→
server_reply	0x9F8023	Application Info	←
ca_info_inq	0x9F8020	CA Support	→
ca_info	0x9F8021	CA Support	←
ca_pmt	0x9F8022	CA Support	→
ca_pmt_reply	0x9F8023	CA Support	←
ca_update	0x9F8024	CA Support	←
oob_tx_tune_req	0x9F8404	Host Control	←
oob_tx_tune_cnf	0x9F8405	Host Control	→
oob_rx_tune_req	0x9F8406	Host Control	←
oob_rx_tune_cnf	0x9F8407	Host Control	→
inband_tune_req	0x9F8408	Host Control	←
inband_tune_cnf	0x9F8409	Host Control	→
profile_req	0x9F840A	Host Control	→
profile_cnf	0x9F840B	Host Control	←
request_pids	0x9F840C	Host Control	→
pids_required	0x9F840D	Host Control	←
pids_required_cnf	0x9F840E	Host Control	→

apdu_tag	tag value	Resource	Direction Host ↔ CableCARD	
			Host modem	CableCARD Modem
system_time_inq	0x9F8442	System Time	←	
system_time	0x9F8443	System Time	→	
open_mmi_req	0x9F8820	MMI	←	
open_mmi_cnf	0x9F8821	MMI	→	
close_mmi_req	0x9F8822	MMI	←	
close_mmi_cnf	0x9F8823	MMI	→	
			Host modem	CableCARD Modem
new_flow_req	0x9F8E00	Extended Channel Support	↔	→
new_flow_cnf	0x9F8E01	Extended Channel Support	↔	←
delete_flow_req	0x9F8E02	Extended Channel Support	←	→
delete_flow_cnf	0x9F8E03	Extended Channel Support	→	←
lost_flow_ind	0x9F8E04	Extended Channel Support	→	←
lost_flow_cnf	0x9F8E05	Extended Channel Support	←	→
inquire_DSG_mode	0x9F8E06	Extended Channel Support	→	→
set_DSG_mode	0x9F8E07	Extended Channel Support	←	←
DSG_packet_error	0x9F8E08	Extended Channel Support	←	←
feature_list_req	0x9F9802	Generic Feature Control	↔	
feature_list	0x9F9803	Generic Feature Control	↔	
feature_list_cnf	0x9F9804	Generic Feature Control	↔	
feature_list_changed	0x9F9805	Generic Feature Control	↔	
feature_parameters_req	0x9F9806	Generic Feature Control	←	
feature_parameters	0x9F9807	Generic Feature Control	↔	
features_parameters_cnf	0x9F9808	Generic Feature Control	↔	
open_homing	0x9F9990	Homing	→	
homing_cancelled	0x9F9991	Homing	→	
open_homing_reply	0x9F9992	Homing	←	
homing_active	0x9F9993	Homing	→	
homing_complete	0x9F9994	Homing	←	
firmware_upgrade	0x9F9995	Homing	←	
firmware_upgrade_reply	0x9F9996	Homing	→	
firmware_upgrade_complete	0x9F9997	Homing	←	
diagnostic_req	0x9FDF00	Generic Diagnostic Support	←	
diagnostic_cnf	0x9FDF01	Generic Diagnostic Support	→	
SAS_connect_rqst	0x9F9A00	Specific Application Support	→	
SAS_connect_cnf	0x9F9A01	Specific Application Support	←	
SAS_data_rqst	0x9F9A02	Specific Application Support	↔	
SAS_data_av	0x9F9A03	Specific Application Support	↔	
SAS_data_cnf	0x9F9A04	Specific Application Support	↔	
SAS_data_query	0x9F9A05	Specific Application Support	↔	
SAS_data_reply	0x9F9A06	Specific Application Support	↔	
host_info_request	0x9F9C00	System Control	←	
host_info_response	0x9F9C01	System Control	→	
code_version_table	0x9F9C02	System Control	←	
code_version_table_reply	0x9F9C03	System Control	→	
host_download_control	0x9F9C04	System Control	→	
host_download_command	0x9F9C05	System Control	←	

<b>apdu_tag</b>	<b>tag value</b>	<b>Resource</b>	<b>Direction Host ↔ CableCARD</b>
stream_profile()	0x9FA010	CARD RES	←
stream_profile_cnf()	0x9FA011	CARD RES	→
program_profile()	0x9FA012	CARD RES	←
program_profile_cnf()	0x9FA013	CARD RES	→
es_profile()	0x9FA014	CARD RES	←
es_profile_cnf()	0x9FA015	CARD RES	→
pids_required()	0x9FA016	CARD RES	←
pids_required_cnf()	0x9FA017	CARD RES	→

## APPENDIX B BASELINE HTML SUPPORT

This appendix describes HTML keywords that SHALL be supported by the CableCARD HTML Baseline and gives requirements for each keyword foreseen on the Host.

The CableCARD HTML Baseline only supports formatted text messages, in the form of HTML pages, with one hyperlink.

The Application Information resource MAY identify Hosts that support more elaborate HTML pages with multiple hyperlinks and multiple levels of text rendering and graphic support. In such a case, the CableCARD device can supply HTML pages that take advantage of these enhanced features.

Note: This extended mode of operation is not described in this appendix.

### B.1 Format

#### B.1.1 Display

The CableCARD HTML Baseline pages SHALL be designed to fit in a 4/3 and 16/9 NTSC display size using the smallest common screen (640 x 480) without vertical and horizontal scrolling.

The CableCARD HTML Baseline specification requires the HTML page to be displayed on a full screen.

#### B.1.2 Font

The CableCARD HTML Baseline font SHALL support a minimum of 32 characters per line, and a minimum of 16 lines of characters.

#### B.1.3 Background Color

The CableCARD HTML Baseline background color is light gray (#C0C0C0)

#### B.1.4 Paragraph

The CableCARD HTML Baseline paragraph alignment is LEFT.

#### B.1.5 Image

The CableCARD HTML Baseline specification does not include support for images.

#### B.1.6 Table

The CableCARD HTML Baseline specification does not include support for tables.

### B.2 Supported User Interactions

#### B.2.1 Navigation and Links

The CableCARD HTML Baseline specification does not define how a hyperlink is navigated and selected. It is up to the Host manufacturer to provide some navigation/selection mechanism to identify the user intention and forward the selected link to the CableCARD device using the server\_query APDU. It is up to the CableCARD device

manufacturer to determine how results are returned to the CableCARD device through the URL of the server\_query APDU.

### B.2.2 Forms

The CableCARD HTML Baseline specification does not include support for forms.

### B.2.3 HTML Keywords

Table 142 lists HTML keywords required for the CableCARD HTML Baseline (R=Required, O=Optional).

A keyword or a parameter marked as optional MAY be inserted in an HTML page, but MAY not be used by the Host. It SHALL NOT change what is displayed on the screen but only the way of displaying it (basically, it applies to the style).

**Table 142 - HTML Keyword List**

	Required or Optional
<b>Structure</b>	
<HTML>...</HTML>	R
Begin and end HTML document.	
<BODY>...</BODY>	R
Begin and end of the body of the document, optional attributes of the document	
• bgcolor: background color, default = light gray (#C0C0C0)	O
• text: color of text, default = black (#000000)	O
• link: color of unvisited links, default = blue (#0000FF)	O
<A href= ...> ... </A>	R
Begin and end an anchor.	
• href: URL targeted by this anchor.	R
<b>Style Element</b>	
<P>	R
Change of paragraph	
• align: CENTER, LEFT, or RIGHT (default = LEFT)	O
 	R
Force new line.	
<B>...</B> <I> ... </I> <U> ... </U>	O
Character style: bold, italic, and underlined	

### B.3 Characters

An HTML page can refer to all Latin-1 characters by their numeric value by enclosing them between the & and ; symbols. For example, the quotation mark “ can be expressed as &#34; in an HTML page. The characters specified in the Added Latin-1 entity set also have mnemonic names. Thus, the following 3 expressions are interpreted as the character “.

&quot;  
&#34;  
“

*NOTE: Mnemonic expressions are case sensitive.*

Table 143 defines characters, their numeric and mnemonic expressions that the OpenCable Baseline HTML viewer SHALL support. Any OpenCable baseline HTML page SHALL NOT use the characters, numeric or mnemonic

expressions, which are not defined in Table 143. The OpenCable host device MAY ignore the characters which are not defined in Table 143.

This list is taken from the HTML 4 Character entity references found at:

<http://www.w3.org/TR/1999/REC-html1401-19991224/sgml/entities.html>

**Table 143 - Characters**

Character	Name	Numeric Expression	Mnemonic Expression
	Horizontal tab	&#9;	
	Line feed	&#10;	
	Space	&#32	
!	Exclamation mark	&#33;	
"	Quotation mark	&#34;	&quot;
#	Number sign	&#35;	
\$	Dollar sign	&#36;	
%	Percent sign	&#37;	
&	Ampersand	&#38;	&amp;
'	Apostrophe	&#39;	
(	Left parenthesis	&#40;	
)	Right parenthesis	&#41;	
*	Asterisk	&#42;	
+	Plus sign	&#43;	
,	Comma	&#44;	
-	Hyphen	&#45;	
.	Period	&#46;	
/	Solidus (slash)	&#47;	
0		&#48;	
1		&#49;	
2		&#50;	
3		&#51;	
4		&#52;	
5		&#53;	
6		&#54;	
7		&#55;	
8		&#56;	
9		&#57;	
:	Colon	&#58;	
;	Semicolon	&#59;	
<	Less than	&#60;	&lt;
=	Equals sign	&#61;	
>	Greater than	&#62;	&gt;
?	Question mark	&#63;	
@	Commercial at	&#64;	
A		&#65;	
B		&#66;	
C		&#67;	
D		&#68;	
E		&#69;	
F		&#70;	
G		&#71;	

Character	Name	Numeric Expression	Mnemonic Expression
H		&#72;	
I		&#73;	
J		&#74;	
K		&#75;	
L		&#76;	
M		&#77;	
N		&#78;	
O		&#79;	
P		&#80;	
Q		&#81;	
R		&#82;	
S		&#83;	
T		&#84;	
U		&#85;	
V		&#86;	
W		&#87;	
X		&#88;	
Y		&#89;	
Z		&#90;	
[	Left square bracket	&#91;	
\	Reverse solidus	&#92;	
]	Right square bracket	&#93;	
^	Circumflex	&#94;	
_	Horizontal bar	&#95;	
`	Grave accent	&#96;	
a		&#97;	
b		&#98;	
c		&#99;	
d		&#100;	
e		&#101;	
f		&#102;	
g		&#103;	
h		&#104;	
I		&#105;	
j		&#106;	
k		&#107;	
l		&#108;	
m		&#109;	
n		&#110;	
o		&#111;	
p		&#112;	
q		&#113;	
r		&#114;	
s		&#115;	
t		&#116;	
u		&#117;	
v		&#118;	
w		&#119;	
x		&#120;	
y		&#121;	
z		&#122;	
{	Left curly brace	&#123;	

Character	Name	Numeric Expression	Mnemonic Expression
	Vertical bar	&#124;	
}	Right curly brace	&#125;	
~	Tilde	&#126;	
	Non-breaking space	&#160;	&nbsp;
¡	Inverted exclamation	&#161;	&iexcl;
¢	Cent	&#162;	&cent;
£	Pound	&#163;	&pound;
¤	Currency	&#164;	&curren;
¥	Yen	&#165;	&yen;
¦	Broken vertical	&#166;	&brvbar;
§	Section sign	&#167;	&sect;
¨	Umlaut/diaeresis	&#168;	&uml;
©	Copyright	&#169;	&copy;
ª	Feminine	&#170;	&ordf;
«	Left angle quote	&#171;	&laquo;
¬	No sign	&#172;	&not;
-	Hyphen	&#173;	&shy;
®	Reg. trade mark	&#174;	&reg;
¯	Macron	&#175;	&macr;
°	Degrees	&#176;	&deg;
±	Plus/Minus	&#177;	&plusmn;
²	Superscript 2	&#178;	&sup2;
³	Superscript 3	&#179;	&sup3;
´	Acute accent	&#180;	&acute;
µ	Micron	&#181;	&micro;
¶	Paragraph sign	&#182;	&para;
·	Middle dot	&#183;	&middot;
¸	Cedilla	&#184;	&cedil;
¹	Superscript 1	&#185;	&sup1;
º	Masculine	&#186;	&ordm;
»	Right angle quote	&#187;	&raquo;
¼	One quarter	&#188;	&frac14;
½	One half	&#189;	&frac12;
¾	Three quarters	&#190;	&frac34;
¿	Inverted question mark	&#191;	&iquest;
À	A Acute	&#192;	&Aacute;
Á	A Grave	&#193;	&Agrave;
Â	A Circumflex	&#194;	&Acirc;
Ã	A Tilde	&#195;	&Atilde;
Ä	A Diaeresis	&#196;	&Auml;
Å	A Ring	&#197;	&Aring;
Æ	AE Diphthong	&#198;	&AElig;
Ç	C Cedilla	&#199;	&Ccedil;
È	E Acute	&#200;	&Eacute;
É	E Grave	&#201;	&Egrave;
Ê	E Circumflex	&#202;	&Ecirc;
Ë	E Diaeresis	&#203;	&Euml;
Ì	I Acute	&#204;	&Iacute;
Í	I Grave	&#205;	&Igrave;
Î	I Circumflex	&#206;	&Icirc;
Ï	I Diaeresis	&#207;	&Iuml;
Ð	Icelandic eth	&#208;	&ETH;

Character	Name	Numeric Expression	Mnemonic Expression
Ñ	N Tilde	&#209;	&Ntilde;
Ò	O Acute	&#210;	&Oacute;
Ó	O Grave	&#211;	&Ograve;
Ô	O Circumflex	&#212;	&Ocirc;
Õ	O Tilde	&#213;	&Otilde;
Ö	O Diaeresis	&#214;	&Ouml;
×	Multiplication	&#215;	&times;
Ø	O Slash	&#216;	&Oslash;
Ù	U Acute	&#217;	&Uacute;
Ú	U Grave	&#218;	&Ugrave;
Û	U Circumflex	&#219;	&Ucirc;
Ü	U Diaeresis	&#220;	&Uuml;
Ý	Y Acute	&#221;	&Yacute;
Þ	Icelandic Thorn	&#222;	&THORN;
ß	Small sharp S	&#223;	&szlig;
à	a Acute	&#224;	&aacute;
á	a Grave	&#225;	&agrave;
â	a Circumflex	&#226;	&acirc;
ã	a Tilde	&#227;	&atilde;
ä	a Diaeresis	&#228;	&auml;
å	a Ring	&#229;	&aring;
æ	ae Diphthong	&#230;	&aelig;
ç	c Cedilla	&#231;	&ccedil;
è	e Acute	&#232;	&eacute;
é	e Grave	&#233;	&egrave;
ê	e Circumflex	&#234;	&ecirc;
ë	e Diaeresis	&#235;	&euml;
ì	i Acute	&#236;	&iacute;
í	i Grave	&#237;	&igrave;
î	i Circumflex	&#238;	&icirc;
ï	i Diaeresis	&#239;	&iuml;
ð	Icelandic eth	&#240;	&eth;
ñ	n Tilde	&#241;	&ntilde;
ò	o Grave	&#242;	&ograve;
ó	o Acute	&#243;	&oacute;
ô	o Circumflex	&#244;	&ocirc;
õ	o Tilde	&#245;	&otilde;
ö	o Diaeresis	&#246;	&ouml;
÷	Division	&#247;	&divide;
ø	o Slash	&#248;	&oslash;
ù	u Acute	&#249;	&uacute;
ú	u Grave	&#250;	&ugrave;
û	u Circumflex	&#251;	&ucirc;
ü	u Diaeresis	&#252;	&uuml;
ý	y Acute	&#253;	&yacute;
þ	Icelandic thorn	&#254;	&thorn;
ÿ	y Diaeresis	&#255;	&yuml;

## APPENDIX C ERROR HANDLING

Interface errors SHALL be handled as described in Table E.1-A of Appendix E of [SCTE28] with the following exceptions to that table:

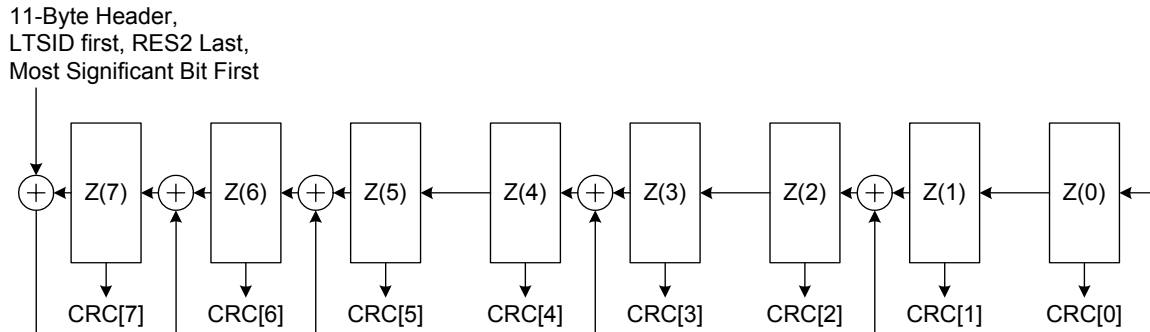
	<b>Error Condition</b>	<b>Failure Mechanism</b>	<b>Host Action</b>	<b>SCTE CableCARD device Action</b>	<b>Comments</b>
1	CableCARD READY signal does not go active	CableCARD	<p>Minimum – Perform 1 PCMCIA reset, Report Error if not successful</p> <p>Optional – Retry PCMCIA resets up to two times and report error.</p> <p>Preferred – Perform at least 1 PCMCIA reset. Report Error if not successful and continue to perform PCMCIA resets.</p>	None	Host reports error to user.
4	Host sets command channel RS bit but CableCARD device fails to set FR bit within 5 second timeout.	CableCARD	<p>Minimum – Perform 1 PCMCIA rest, Report Error if not successful,</p> <p>Optional – Retry PCMCIA resets up to two times and report error.</p> <p>Preferred – Perform at least 1 PCMCIA reset. Report Error if not successful and continue to perform PCMCIA resets.</p>	None	Host reports error to user.
5	Host sets extended channel RS bit but CableCARD device fails to set FR bit within 5 second timeout.	CableCARD	<p>Minimum – Perform 1 PCMCIA reset, Report Error if not successful</p> <p>Optional – Retry PCMCIA resets up to two times and report error.</p> <p>Preferred – Perform at least 1 PCMCIA reset. Report Error if not successful and continue to perform PCMCIA resets.</p>	None	Host reports error to user.

	<b>Error Condition</b>	<b>Failure Mechanism</b>	<b>Host Action</b>	<b>SCTE CableCARD device Action</b>	<b>Comments</b>
10	CableCARD device does not respond to Hosts open transport request within 5 seconds	CableCARD	<p>Minimum – Perform 1 PCMCIA reset, Report Error if not successful</p> <p>Optional – Retry PCMCIA resets up to two times and report error.</p> <p>Preferred – Perform at least 1 PCMCIA reset. Report Error if not successful and continue to perform PCMCIA resets.</p>	None	Host reports error to user.
17	CableCARD device fails to respond to profile_inq within 5 seconds.	CableCARD	<p>Minimum – Perform 1 PCMCIA reset, Report Error if not successful.</p> <p>Optional – Retry PCMCIA resets up to two times and report error.</p> <p>Preferred – Perform at least 1 PCMCIA reset. Report Error if not successful and continue to perform PCMCIA resets.</p>	None	Host reports error to user.
38	CableCARD device fails to respond to ca_info_inq within 5 seconds.	CableCARD	<p>Minimum – Perform 1 PCMCIA reset, Report Error if not successful.</p> <p>Optional – Retry PCMCIA resets up to two times and report error.</p> <p>Preferred – Perform at least 1 PCMCIA reset. Report Error if not successful and continue to perform PCMCIA resets.</p>	None	Host reports error to user.

	<b>Error Condition</b>	<b>Failure Mechanism</b>	<b>Host Action</b>	<b>SCTE CableCARD device Action</b>	<b>Comments</b>
53	CableCARD device fails to respond to any request within 5 seconds	CableCARD	<p>Minimum – Perform 1 PCMCIA reset, Report Error if not successful.</p> <p>Optional – Retry PCMCIA resets up to two times and report error.</p> <p>Preferred – Perform at least 1 PCMCIA reset. Report Error if not successful and continue to perform PCMCIA resets.</p>	None	User MAY see frozen picture on scrambled channels.

## APPENDIX D CRC-8 REFERENCE MODEL

The 8-bit CRC generator/checker for Multi-Stream CableCARD applications is specified in Figure 26.



**Figure 26 - 8 bit CRC generator/checker model**

The model shown above implements the CRC-8 value used in the MPEG Transport Stream Pre-Header, utilizing the generator Polynomial:

$$x^8 + x^7 + x^6 + x^4 + x^2 + 1$$

The CRC-8 generator/checker operates on the first 11 bytes of the MPEG Transport Stream Pre-Header, starting with the LTSID field and ending with the RES2 field. Each byte is operated on Most Significant Bit first, and the model is initialized with all ones before the first byte is sent through the model. After the 11 bytes are processed, the CRC-8 value (CRC[7:0]) is taken from the 8 delay elements of the model. This value is placed in the 12<sup>th</sup> byte of the MPEG Transport Stream Pre-Header (for the generator) or compared with the 12<sup>th</sup> byte of the MPEG Transport Stream Pre-Header (for the checker).

An example stream and associated CRC-8 is:

0x01, 0x00, 0x55, 0xAA, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00

Produces a CRC of 0x8A.

## APPENDIX E REVISION HISTORY

OC-SP-MC-IF-I02-040831 contains modifications from the following ECNs.

<b>ECN</b>	<b>Date Accepted</b>	<b>Summary</b>
MC-IF-N-04.0585-2	4/9/04	Add Support for Common Download Specification Requirement
MC-IF-N-04.0590-1	4/9/04	CPU Interface Packet Format - no data to send condition
MC-IF-N-04.0591-1	4/9/04	Correct Detection of S-CARD or M-CARD Operational Mode
MC-IF-N-04.0592-1	4/9/04	CableCARD MPEG Packet (CMP) Handling by M-CARD and Host
MC-IF-N-04.0593-1	4/9/04	Specify M-CARD Requirements on Host_reserved Transport Pre-Header field processing
MC-IF-N-04.0595-1	4/29/04	Clarify use of OOB FDC in DSG mode for Multistream CableCARD
MC-IF-N-04.0648-2	8/13/04	Revisions to the ca_pmt in the M-CARD environment
MC-IF-N-04.0653-1	8/13/04	Clarification of CRC-8