CERTIFICATE

By Authority Of
THE UNITED STATES OF AMERICA
Legally Binding Document

By the Authority Vested By Part 5 of the United States Code § 552(a) and Part 1 of the Code of Regulations § 51 the attached document has been duly INCORPORATED BY REFERENCE and shall be considered legally binding upon all citizens and residents of the United States of America. *HEED THIS NOTICE*: Criminal penalties may apply for noncompliance.

Document Name: SCTE 28: Host-POD Interface Standard

CFR Section(s): 47 CFR 15.123(b)(4)

Standards Body: Society of Cable Telecommunications Engineers

APPROVED

Official Incorporator:

THE EXECUTIVE DIRECTOR
OFFICE OF THE FEDERAL REGISTER
WASHINGTON, D.C.

Society of Cable Telecommunications Engineers

ENGINEERING COMMITTEE
Digital Video Subcommittee

AMERICAN NATIONAL STANDARD

ANSI/SCTE 28 2007

HOST-POD Interface Standard

NOTICE

The Society of Cable Telecommunications Engineers (SCTE) Standards are intended to serve the public interest by providing specifications, test methods and procedures that promote uniformity of product, interchangeability and ultimately the long term reliability of broadband communications facilities. These documents shall not in any way preclude any member or nonmember of SCTE from manufacturing or selling products not conforming to such documents, nor shall the existence of such standards preclude their voluntary use by those other than SCTE members, whether used domestically or internationally.

SCTE assumes no obligations or liability whatsoever to any party who may adopt the Standards. Such adopting party assumes all risks associated with adoption of these Standards or Recommended Practices, and accepts full responsibility for any damage and/or claims arising from the adoption of such Standards or Recommended Practices.

Attention is called to the possibility that implementation of this standard may require use of subject matter covered by patent rights. By publication of this standard, no position is taken with respect to the existence or validity of any patent rights in connection therewith. SCTE shall not be responsible for identifying patents for which a license may be required or for conducting inquires into the legal validity or scope of those patents that are brought to its attention.

Patent holders who believe that they hold patents which are essential to the implementation of this standard have been requested to provide information about those patents and any related licensing terms and conditions. Any such declarations made before or after publication of this document are available on the SCTE web site at http://www.scte.org.

140 Philips Road
Exton, PA 19341

Table of Contents

List of Tables

List of Figures

Host-POD
Interface Specification

1 SCOPE

This standard defines the characteristics and normative specifications for the interface between Point of Deployment (POD) security modules owned and distributed by cable operators, and commercially available consumer receivers and set-top terminals ("Host devices") that are used to access multi-channel television programming carried on North American cable systems. The Point-of-Deployment module is also known as a CableCARD™ device. These Host devices may also be supplied by the cable operators. The combination of a properly-authorized POD module and a Host device permits the unscrambled display of cable programming that is otherwise protected by a conditional access scrambling system.

This standard applies extensions, modifications, and constraints to the interface defined in CEA-679B Part B, the National Renewable Security Standard.

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 ANSI SCTE 55-1 2002 and ANSI/SCTE 55-2 2002. Other data transfer mechanisms such as the signaling methods of the DOCSIS version. 1.1 cable modem standard may be supported in the Host device. A cable operator is able to upgrade security in response to a breach by replacing the POD modules, without requiring any change in the host device.

The interface will support Emergency Alert messages transmitted over the out of band channel to the POD module and then delivered by the POD module over the interface to the host device using the format defined in SCTE 18 2002.

It may also support Interactive Program Guide services, Impulse Pay Per View services, Video on Demand, and other messaging and interactive services. It supports both one way and two way cable systems, as well as host devices that incorporate DOCSIS modems or telco modems.

This standard defines the physical interface, signal timing, the link interface, and the application interface. It includes the extended channel specification, power management specifications, initialization procedures and firmware upgrade methods.

2 OVERVIEW OF HOST-POD INTERFACE

2.1 Historical Perspective (INFORMATIVE)

This specification has its origins in CEA-679 (formerly EIA-679), the National Renewable Security Standard, which was initially adopted in September 1998. Part B of that standard has the physical size, shape and connector of the computer industry PCMCIA card, and also defines the interface protocols and stack. Part B of that standard was adopted by SCTE DVS as DVS/064.

Further extensions and modifications of EIA-679 led to the adoption of EIA-679-B in 2000. Independently, the cable industry prepared a modified version of DVS/064 which was submitted as DVS/131. Revision 7 of that document was adopted by SCTE DVS in early 1999 but never attained the status of a final standard because there were comments and objections that were never resolved.

Instead, the cable industry prepared a revised version that was submitted in January 2000 as DVS/295, incorporating many of the comments associated with DVS/131. Work on this document by the cable industry proceeded during the first half of 2000, leading to substantial changes that were embodied in DVS/295r1 (July 2000) and subsequent revisions in the open review process.

2.2 Advanced Cable Services (INFORMATIVE)

The POD Module interface specification is designed to support advanced digital cable services by a digital television receiver when a POD Module is inserted.

In this case, "Advanced Digital Cable Services" would include support of the following functions:

- Emergency Alert System
- Interactive Program Guide
- Impulse Pay-Per-View (IPPV)
- Video On Demand (VOD)
- General Messaging
- Interactive Services

2.2.1 Interactive Program Guide (IPG)

The Host may support an Interactive Program Guide (IPG) to enable the user to navigate to available services. The services supported by the IPG may include basic channel, premium channels, and Impulse Pay-Per-View (IPPV) events. Program

guide data may be delivered to the application by means of the in-band (QAM) channel and/or by means of the out-of-band (QPSK) channel:

- In-band transmission of program and system information typically describes only the digital multiplex in which it is sent. This means that a single-tuner Host must periodically scan through all channels to receive data for each channel and store this information in memory.
- Optionally, at the discretion of the cable operator, the out-of-band channel may be used to deliver guide data. The format of this information over the OOB channel will be defined by the cable operator and may be used to support specific IPG implementations. The Host receives data from the POD that is sent on the out-of-band channel and delivered over the ***Extended Channel*** described in Section 5.The guide data typically describes the entire range of services offered by the cable system.

2.2.2 Impulse Pay-Per-View (IPPV)

The Host may support the purchase of Impulse Pay-Per-View (IPPV) events. IPPV processing is split as follows:

- All security related and billing functions are in the POD Module.
- All user-interface functions are in the Host.

The IPPV API is specified by "Generic IPPV Support" in section 8.10 and covers all common functions related to (1) IPPV purchase (2) IPPV cancel (3) IPPV purchase review.

2.2.3 Video-on-Demand (VOD)

Video-on-Demand (VOD) may be modeled as an IPPV event where the program stream is dedicated to an individual subscriber. The VOD application executes in the Host and supports all of the User Interface (UI) functions.

The additional streaming media control functions (i.e. Pause, Play, Fast-Forward, Rewind) may be supported using DSM-CC User-to-User messages. The ***Extended Channel*** described in Section 5 may be used as the communication path for VOD signaling, and may also be used for VOD event purchases. After a VOD control session is established via the session creation interface, UDP messages may be exchanged transparently between the Host and the cable system. RFC 1831, 1832 & 1833 may be used as the underlying RPC mechanism for the exchange of DSM-CC UU.

2.2.4 Interactive services

Interactive Services may be supported by applications executing on the Host, for example, an email or game application. To advertise interactive services, a mechanism is required to deliver information about applications to the Host and the

protocols described in ANSI/SCTE 80 2002 may be used for this purpose. Typically, information about interactive services are not associated with a streaming media service, so information about them is delivered via the out-of-band channel. The service information is passed to the Host via the ***Extended Channel*** resource when the POD Module serves as the OOB modem.

The ***Extended Channel*** may also be used as the communication path for interactive service signaling when the POD Module is serving as the OOB modem. After an interactive service session is established via the session creation interface, UDP messages may be exchanged transparently between the Host and the cable system. RFC 1831, 1832 & 1833 may be used as the underlying RPC mechanism for the exchange of application level messages.

2.3 References

2.3.1 Normative references

The following standards contain provisions that, through reference in this text, constitute normative provisions of this Specification. At the time of publication, the editions indicated are current. All standards are subject to revision, and parties to agreements based on this Specification are encouraged to investigate the possibility of applying for the most recent editions of the standards listed in this section.

Normative reference list

1. CEA-679-C Part B: National Renewable Security Standard (July 2005)

2. ANSI/SCTE 55-2 2002 (formerly DVS 167), Digital Broadband Delivery System: Out Of Band Transport – Mode B

3. ANSI/SCTE 55-1 2002 (formerly DVS 178), Digital Broadband Delivery System: Out Of Band Transport – Mode A

4. SCTE 18 2002 (formerly DVS 208), Emergency Alert Message for Cable, approved as a joint standard with CEA as ANSI-J-STD-042-2002

5. ANSI/SCTE 65 2002 (formerly DVS 234), Service Information Delivered Out-of-Band for Digital Cable Television (28 March, 2000)

6. ANSI/SCTE 54 2004, Digital Video Service Multiplex and Transport System Standard for Cable Television

7. ISO/IEC 13818-6:1998 (E) Information Technology: Generic coding of moving pictures and associated audio information. Part 6: Extension for DSM-CC

8. ISO 8859-1: 8-Bit Single-Byte Coded Graphic Character Sets - Part 1: Latin Alphabet No. 1, revised 1987

9. PC Card Standard, Volume 2 Electrical Specification, March 1997, Personal Computer Memory Card International Association, Sunnyvale, CA.

10. PC Card Standard, Volume 4 Metaformat Specification, 2001, Personal Computer Memory Card International Association, Sunnyvale, CA

11. RFC 2131, Dynamic Host Configuration Protocol, March 1997.

12. RFC 2132, DHCP Options and BOOTP Vendor Extensions, March 1997.

13. PC Card Standard, Volume 3 Physical Specification, Release 7, February 1999, Personal Computer Memory Card International Association, Sunnyvale, CA.

14. ANSI/SCTE 41 2004, POD Copy Protection System

15. HTML 3.2 Reference Specification: http://www.w3.org/TR/REC-html32.html

16. Hypertext Transfer Protocol – HTTP/1.1: http://www.ietf.org/rfc/rfc2616.txt?number=2616

17. SCTE 23-2 2002, Data-Over-Cable Systems 1.1, Baseline Privacy Plus Interface Specification,

18. DOCSIS Set-top Gateway (DSG) Interface Specification, CM-SP-DSG-I08-060728

19. ANSI/SCTE 90-1 2004 SCTE Applications Platform Part 1: OCAP 1.0 Profile

Normative reference acquisition

ANSI/CEA Standards:

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

CEA 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.commailto:global@ihs.com ; URL: <http://global.ihs.com>

SCTE Standards: United States of America

- Society of Cable Telecommunications Engineers Inc., 140 Philips Road, Exton, PA 19341; Telephone 800-542-5040; Facsimile: 610-363-5898; E-mail: standards@scte.org ;URL: <http://www.scte.org>

ITU Standards:

- ITU Sales and Marketing Service, International Telecommunication Union, Place des Nations CH-1211, Geneva 20, Switzerland; Telephone: +41 22 730 6141; Facsimile: +41 22 730 5194; E-mail: sales@itu.int ; URL: <http://www.itu.org>

ISO/IEC Standards:

- Global Engineering Documents, World Headquarters, 15 Inverness Way East, Englewood, CO 80112-5776, USA; Telephone: 800-854-7179; Facsimile: 303-397-2740; E-mail: global@ihs.commailto: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; URL: http://www.ietf.org/rfc

PC Card Standards:

- Personal Computer Memory Card International Association, 2635 North First Street, Suite 209, San Jose, CA 95134, (Tel) +408-433-CARD (2273), (Fax) +408-433-9558, (Email) office@pcmcia.org

DOCSIS and OpenCable Specifications:

- Cable Television Laboratories, Inc., 858 Coal Creek Circle, Louisville, CO 80027-9750; URLs: http://cablelabs.com/, http://www.cablemodem.com/, http://www.opencable.com/

2.3.2 Informative references

The following documents contain information that is useful in understanding of this Specification. Some of these documents are drafts of standards or balloted standards with unresolved comments.

Informative document list

1. OC-SP-CD-IF-I08-040831 OpenCable Common Download Specification

2. Data-Over-Cable Service Interface Specifications, Radio Frequency Interface Specification, SP-RFIv1.1-I10-030730.

3. Data-Over-Cable Service Interface Specifications, Radio Frequency Interface Specification, SP-RFIv2.0-I07-041210

4. Data-Over-Cable Service Interface Specifications, Operations Support System Interface Specification, ANSI/SCTE 23-3 2005.

5. Data-Over-Cable Service Interface Specifications, Operations Support System Interface Specification, SP-OSSIv2.0-I07-041210

Informative document acquisition

Same as listed under Normative reference acquisition.

3 CEA 679 PART B COMPLIANCE

3.1 Exceptions to Compliance

In all aspects not covered in this document, the POD Module interface requires complete CEA-679-Part C compliance with the following exceptions:

Table 3.1-A CEA 679 Part B Compliance Exceptions

Item	Section	Issue	Comment
1	4.1	Second paragraph states "It also allows for multiple instance of CA processes to exist for the same Host."	SCTE 28 supports only one POD Module per POD/Host Interface.
2	4.1	Third paragraph is a description of handling multiple modules.	SCTE 28 supports only one POD Module per POD/Host Interface
3	5.3	First paragraph states "This functionality includes: the ability to support multiple modules on one host, ...".	SCTE 28 supports only one POD Module per POD/Host Interface
4	5.4.1	Item 5 of the requirements and limits list is "5) use of multiple modules".	SCTE 28 supports only one POD Module per POD/Host Interface
5	5.4.2	Item 3 of the transport stream interface restrictions regarding the use of the word contiguous.	Packets arrive synchronously with clock but not necessarily continuously
6	5.4.2	Item 5 of the transport stream interface specifies the maximum data rate of 58 Mbps..	Section 6.1.1 specifies rates around 27 and 39 Mbps.
7	5.4.2	Maximum jitter is not defined in the document.	Defined in Section 6.1.1
8	5.4.4	Entire paragraph is about multiple modules.	SCTE 28 supports only one POD Module per POD/Host Interface
9	5.5	First paragraph sentence contains "... and indicates to the host that it is a NRSS-conformant module."	"NRSS-conformant module" should be changed to a POD module.

Table 3.1-A CEA 679 Part B Compliance Exceptions			
Item	**Section**	**Issue**	**Comment**
10	7.2.4	Item 2 of the SPDU list states "a conditional body of variable length which contains an integer number of APDUs belonging to the same session (see application layer)."	Section 8.1 limits the number of APDUs in an SPDU to one.
11	7.2.6	"... which is always followed by a SPDU body containing one or several APDUs."	Section 8.1 limits the number of APDUs in an SPDU to one.
12	8.2.1	First paragraph contains "Resources with higher version number shall be backwards compatible with previous versions, so that applications requesting a previous version will have a resource with expected behavior.	All resources that are defined in CEA-679-C that are modified in SCTE 28 should have a different type value or version number.
13	8.4.1.1	First paragraph contains "When a module is plugged in or the host is powered up one or perhaps two transport connections are created to the module, ...".	Section 8.1 limits the number of transport connections to one.
14	8.4.2	Entire section.	Section 8.4 replaces this entire section.
15	8.5	**Entire section.**	Section 8.8 of SCTE 28 replaces Section 8.5 of CEA 679B Part B except for Section 8.5.2 which remains.
16	8.6	Entire section	Section 8.3 replaces this entire section.
17	8.7	Entire section.	Section 8.5 modifies this operation.
18	8.8	Entire section.	Section 8.13 replaces this entire section.
19	8.9	Entire section.	This entire section is replaced by ANSI/SCTE 41 2001
20	8.10	Entire section.	SCTE 28 does not require this resource.
21	8.11	Table 87 & 91	Table 87 is replaced by Table 3.1-B and Table 91 is replaced by Table 3.1-C.
22	8.11	EIA-679-B Part B contained two sections numbered 8.11. This was corrected in CEA-679-C Part B, where they are now numbered 8.11 and 8.12. The Section formerly numbered 8.12 in EIA-679-B Part B has been renumbered 8.13 in CEA-679-C Part B.	

Table 3.1-A CEA 679 Part B Compliance Exceptions			
Item	**Section**	**Issue**	**Comment**
23	8.12.2	Host Control sections	Host Control operation in Sections 8.12.1 and 8.12.2 of CEA-679-C is replaced by Section 8.8.
24	8.12.3-8.12.6	Extended Channel Support	These sections of CEA-679-C Part B are replaced by Sections 5 and 8.9
25	8.13	Entire section	Section 8.13 of CEA-679-C is replaced by Section 8.10.
26	A.2.2.1	Status register description.	IIR flag and operation description is given in Section 6.5.2 of SCTE 28.
27	A.4.1.3	Entire section	Removed by section 6.7.6.
28	A.5.5.1	"Hosts shall support 5V working and may optionally support 3.3V working."	Section 6.1.2 specifically states that the POD module is only implemented as a 3.3V device.
29	A.5.5.2	Table 119	1. MCLKO has been moved from pin 57 to pin 14. Pin 14 is shared with A14 and becomes an I/O. Pin 57 shall be VS2# always. 2. Pin 11 Address 9 is also DRX. 3. Pin 12 Address 8 is also CRX. 4. Pin 22 Address 7 is also QTX. It is also changed to an I/O. 5. Pin 23 Address 6 is also ETX. It is also changed to an I/O. 6. Pin 24 Address 5 is also ITX. It is also changed to an I/O. 7. Pin 25 Address 4 is also CTX,
30	A.5.5.10	Item 1 of the power management features list: "Except in standby mode, ..."	POD Standby mode is not defined in SCTE 28.
31	A.5.5.10	Item 2 of the power management features list.	Section 6.1.2 modifies this to 1 amp.
32	A.5.5.10	Item 4 of the power management features list.	Section 6.1.2 modifies this to 250 ma on Vpp.
33	A.5.6	Item 5 of the metaformat list regarding ID number.	Section 6.1.1 changes this to 0341h.

Table 3.1-A CEA 679 Part B Compliance Exceptions			
Item	**Section**	**Issue**	**Comment**
34	A.5.6	Item 5 of the metaformat list lists STCI_STR as "NRSS_CI_V1.00".	Section 6.2 changes this to "OPENCABLE_ POD_ MODULE".
35	A.5.6	Item 7 of the metaformat list lists system name as "NRSS_HOST"	Change this to "OPENCABLE_ HOST"
36	A.5.6	Item 8 of the metaformat list lists physical device name as "NRSS_CI_MODULE".	Section 6.2 changes this to "OPENCABLE_ POD_ MODULE".
37	A.5.6	CISTPL_LINKTARGET is not included.	Section 6.3 requires this tuple per PCMCIA recommendations.
38	B	Entire appendix	Should be replaced by appendix C.
39	C.1	Entire section	SCTE 28 does not support this resource.
40	C.2	Entire section	SCTE 28 does not support this resource.
41	C.3	Entire section	SCTE 28 does not support this resource.
42	D	Entire section	Homing operation is modified by Section 8.13.
43	E	Entire section.	SCTE 28 supports only one POD Module per POD/Host Interface
44	F	Entire section	SCTE 28 does not require smart-card resource.
45	A.2.1.2.2-A.2.2.2	Single buffer implementation	SCTE 28 does not support a single buffer implementation in the POD.

- The requirement of section 7.2.6.1 of CEA-679-C Part B that the Host or POD module must support earlier versions of a resource shall only apply to versions of resources described in this specification.

- Transport layer timeout period shall be modified from 300 ms to 5 seconds.

- Hardware Interface Description – The term "command register" is used erroneously. It shall therefore be referred to as "Control Register."

- The requirement of Section 8.3.3 of CEA-679-C Part B to support an integer number of APDUs in a body of a single SPDU shall be changed to only a single APDU shall be supported in the body of a SPDU.

- The requirement of Section 8.4.3.5 of CEA-679-B Part B that the POD shall implement its CA application such that when ca_enable is present in ca_pmt_reply()

both at program level and elementary stream level, only the ca_enable at ES level applies for that elementary stream shall be applicable to a POD in a network that support different authorization at program level and elementary stream level.

- NOTE: Section 8.4.3.5 of CEA-679-C Part B states that the CA PMT Reply "may" also be sent after reception of a CA PMT object. Implementors should be aware that sending a CA_PMT object to the POD with ca_pmt_cmd_id set to 'ok_mmi' is not guaranteed to return a response.

- Products compliant with this specification shall provide a double buffer implementation. The single buffer implementation permitted by CEA-679-C shall not be used.

Table 3.1-B Replacement for CEA-679-C Table 87
Resource Identifier Values

Resource	class	type	Version	resource identifier
Resource Manager	1	1	1	00010041
Application Information	2	2	1	00020081
Conditional Access Support	3	1	2	00030042
Host Control	32	1	3	00200043
System Time	36	1	1	00240041
MMI	64	2	1	00400081
Low Speed Communication	96	**	2	0060xxx2
Homing	17	1	2	00110042
Copy Protection	176	3	1	00B000C1
Specific Application	144	1	1	00900041
Generic Feature	42	1	1	002A0041
Extended Channel	160	1	1	00A00041
Generic IPPV Support	128	2	1	00800081

** - See section 8.5 for details.

Table 3.1-C Replacement for CEA-679-C Table 91
Application Object Tag Values

apdu_tag	tag value (hex)	Resource	Direction Host ↔ POD
$T_{profile\_inq}$	9F 80 10	Resource Manager	↔
$T_{profile\_reply}$	9F 80 11	Resource Manager	↔
$T_{profile\_changed}$	9F 80 12	Resource Manager	↔
$T_{application\_info\_req}$	9F 80 20	Application Info	→
$T_{application\_info\_cnf}$	9F 80 21	Application Info	←
T_{server_query}	9F 80 22	Application Info	→
T_{server_reply}	9F 80 23	Application Info	←
$T_{ca\_info\_inq}$**	9F 80 30	CA Support	→
T_{ca_info}**	9F 80 31	CA Support	←

Table 3.1-C Replacement for CEA-679-C Table 91 Application Object Tag Values			
apdu_tag	**tag value (hex)**	**Resource**	**Direction Host ↔ POD**
T_{ca_pmt}**	9F 80 32	CA Support	→
$T_{ca\_pmt\_reply}$**	9F 80 33	CA Support	←
T_{ca_update}	9F 80 34	CA Support	←
$T_{OOB\_TX\_tune\_req}$	9F 84 04	Host Control	←
$T_{OOB\_TX\_tune\_cnf}$	9F 84 05	Host Control	→
$T_{OOB\_RX\_tune\_req}$	9F 84 06	Host Control	←
$T_{OOB\_RX\_tune\_cnf}$	9F 84 07	Host Control	→
T_{inband_tune}	9F 84 08	Host Control	←
$T_{inband\_tune\_cnf}$	9F 84 09	Host Control	→
$T_{system\_time\_inq}$**	9F 84 42	System Time	←
T_{system_time}**	9F 84 43	System Time	→
$T_{open\_mmi\_req}$	9F 88 20	MMI	←
$T_{open\_mmi\_cnf}$	9F 88 21	MMI	→
$T_{close\_mmi\_req}$	9F 88 22	MMI	←
$T_{close\_mmi\_cnf}$	9F 88 23	MMI	→
T_{comms_cmd}	9F 8C 00	Low speed comms.	←
$T_{connection\_descriptor}$	9F 8C 01	Low speed comms.	←
T_{comms_reply}	9F 8C 02	Low speed comms.	→
$T_{comms\_send\_last}$	9F 8C 03	Low speed comms.	←
$T_{comms\_send\_more}$	9F 8C 04	Low speed comms.	←
$T_{comms\_rcv\_last}$	9F 8C 05	Low speed comms.	→
$T_{comms\_rcv\_more}$	9F 8C 06	Low speed comms.	→
$T_{new\_flow\_req}$	9F 8E 00	Extended Channel	↔*
$T_{new\_flow\_cnf}$	9F 8E 01	Extended Channel	↔*
$T_{delete\_flow\_req}$	9F 8E 02	Extended Channel	↔*
$T_{delete\_flow\_cnf}$	9F 8E 03	Extended Channel	↔*
$T_{lost\_flow\_ind}$	9F 8E 04	Extended Channel	↔*
$T_{lost\_flow\_cnf}$	9F 8E 05	Extended Channel	↔*
$T_{program\_req}$	9F 8F 00	Generic IPPV Support	→
$T_{program\_cnf}$	9F 8F 01	Generic IPPV Support	←
$T_{purchase\_req}$	9F 8F 02	Generic IPPV Support	→
$T_{purchase\_cnf}$	9F 8F 03	Generic IPPV Support	←
T_{cancel_req}	9F 8F 04	Generic IPPV Support	→
T_{cancel_cnf}	9F 8F 05	Generic IPPV Support	←
$T_{history\_req}$	9F 8F 06	Generic IPPV Support	→
$T_{history\_cnf}$	9F 8F 07	Generic IPPV Support	←
$T_{feature\_list\_req}$	9F 98 02	Generic Feature Control	↔
$T_{feature\_list}$	9F 98 03	Generic Feature Control	↔
$T_{feature\_list\_cnf}$	9F 98 04	Generic Feature Control	↔
$T_{feature\_list\_changed}$	9F 98 05	Generic Feature Control	↔
$T_{feature\_parameters\_req}$	9F 98 06	Generic Feature Control	↔
$T_{feature\_parameters}$	9F 98 07	Generic Feature Control	↔
$T_{feature\_parameters\_cnf}$	9F 98 08	Generic Feature Control	↔
T_{open_homing}	9F 99 90	Homing	→
$T_{homing\_cancelled}$	9F 99 91	Homing	→
$T_{open\_homing\_reply}$	9F 99 92	Homing	←

Table 3.1-C Replacement for CEA-679-C Table 91 Application Object Tag Values			
apdu_tag	**tag value (hex)**	**Resource**	**Direction Host ↔ POD**
T_{homing_active}	9F 99 93	Homing	→
$T_{homing\_complete}$	9F 99 94	Homing	←
$T_{firmware\_upgrade}$	9F 99 95	Homing	←
$T_{firmware\_upgrade\_reply}$	9F 99 96	Homing	→
$T_{firmware\_upgrade\_complete}$	9F 99 97	Homing	←

* - Direction depends on if Host has modem. See section 8.9.
** These values are copied directly from Table 91 of CEA-679-C Part B and do not appear explicitly in this document.

4 SYSTEM ARCHITECTURE (INFORMATIVE)

4.1 Introduction

At the subscriber premises, a reception system includes a cable navigation device, or Host, and a POD Module. This combination allows the isolation of cable operator hardware specifics into a renewable POD Module and therefore provides the architectural foundation for retail availability of cable navigation devices. The POD Module interface consists of a standardized:

- Bi-directional access to the Out-Of-Band RF Front End, or alternatively access to forward Out-Of-Band Messaging supplied by one or more DSG Tunnels via the DOCSIS Set-top Gateway (DSG) [2]
- In-band MPEG-2 Transport Stream input and output, and
- CPU interface.

The Host-POD Interface will operate in one of two modes, a mode using SCTE 55-1 or SCTE 55-2 OOB channels (the OOB mode), or a mode that uses the DOCSIS Set-top Gateway (DSG) for the forward OOB messaging and the normal DOCSIS IP channel for return traffic (the DSG mode).

In the first mode of operation, the signaling functions are split between the Host and the POD Module such that only the RF processing and QPSK demodulation and modulation are done in the Host. The Advanced Host will operate in either of these two modes, OOB or DSG, based on network configurations. All other Hosts will operate in the OOB mode.

The remainder of the processing, including all of the Data-link and MAC protocols, is implemented in the POD Module. This split was chosen for the following reasons:

- SCTE 55-1 and SCTE 55-2 use common modulation (QPSK), but in all other respects are quite different. Only the parts of the protocol stack common to both OOB schemes are included in the Host.

- Future development of OOB protocols should not be precluded. By placing the majority of the OOB processing in the POD Module, the OOB can be renewed at a future time by replacement of the POD Module.

- It is important to the cable operator that the reverse (upstream) transmissions from any device are correctly controlled, because a single uncontrolled device can impair a significant portion of the shared access network. By implementing the media access control processing in the POD Module, the cable operator can maintain the integrity of the access network.

- All processing of conditional access messages is done in the POD Module. This approach is taken to protect from theft-of-service attacks.

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

- The Host optionally removes the IP headers of these packets as instructed by the POD (the POD specifies the number of bytes to be removed from the header of the IP packet).

- The resulting data packets are transmitted over the extended channel to the POD module.

The POD Module can be used in a number of different networks, as described in the sections below.

4.2 Two-way Networks

Figure 4.2-1 gives a schematic view of the system when the cable network includes an OOB return Data Channel based on ANSI/SCTE 55-1 2002 or ANSI/SCTE 55-2 2002.

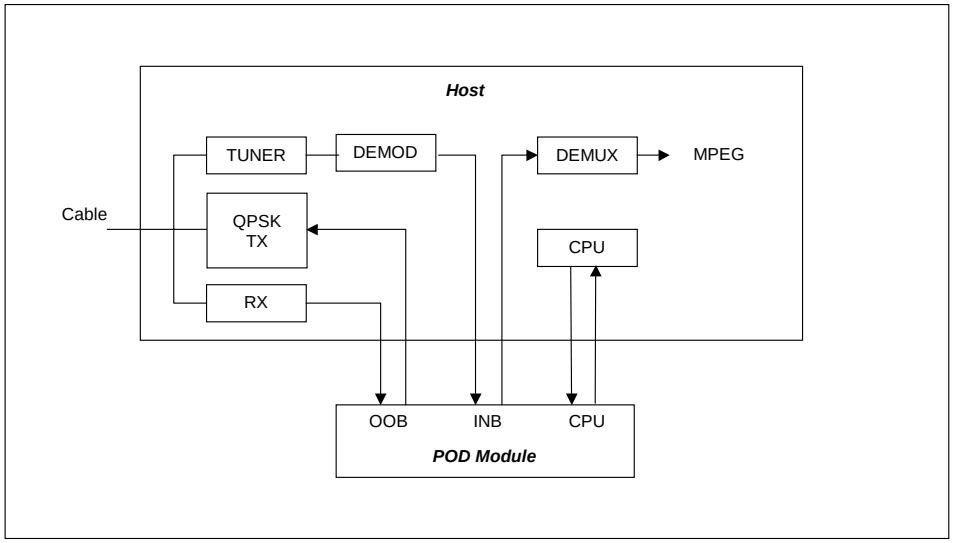

Figure 4.2-1 System with Two-way Network

The QPSK receiver circuit in the Host tunes and demodulates the QPSK Forward Data Channel (FDC). The receiver circuit adapts to the 1.544/3.088 Mbps or 2.048 Mbps FDC bit rate, and delivers the serial bit-stream and clock to the POD Module. (This serial data is used primarily to send conditional access entitlement management messages from the cable system to the POD Module. These messages are beyond the scope of this standard.)

Tuning of the QPSK receiver circuit is under control of the POD Module, as explained in Section 8.8.2. The tuning range is between 70 and 130 MHz.

In the return path, the POD Module generates QPSK symbols and clock and transfers them to the QPSK transmitter circuit in the Host. The transmitter circuit adapts to the 1.544/3.088 Mbps or 0.256 Mbps RDC bit rate. The QPSK transmitter circuit modulates the QPSK symbols onto a narrow band carrier.

Tuning and level control of the QPSK transmitter are under control of the POD Module as explained in Section 8.8.1. The tuning range is between 5 MHz and 42 MHz .

4.3 One-way Networks

The configuration shown in Figure 4.3-1 applies where there is a no return channel.

The QPSK transmitter in the Host is not active (and it is therefore omitted from the diagram). The receiver circuit operates in the same manner as described in Section 8.8.

An optional telephone modem may be used in one-way networks to allow limited interactive services. In this case, the standard telephone modem is incorporated into the Host.

The Host may opt to allow the POD to have access to the telephone modem. In the event that the Host permits the POD to have access to the telephone modem, the POD Module may access the telephone modem via the Low Speed Communications Resource defined in CEA 679-C Part B. Support of the Low Speed Communication Resource as defined in CEA 679-C Part B is optional.

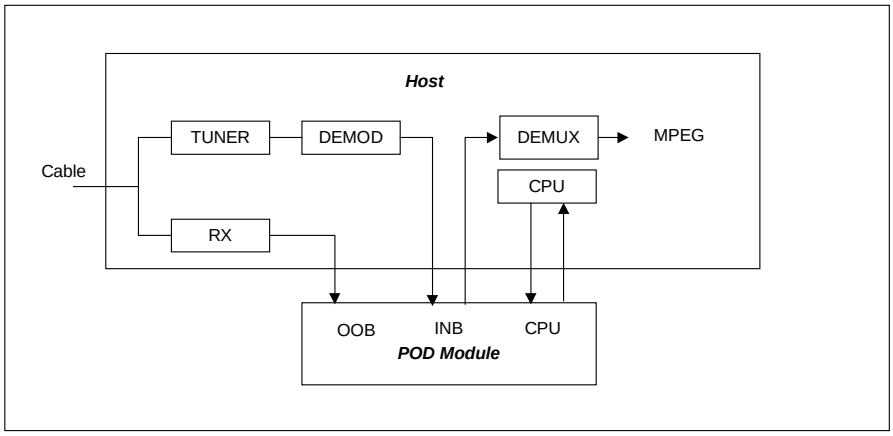

Figure 4.3-1 System with One-way Network

After POD Module initialization, the Host informs the POD Module about the available Low Speed Communication resources as defined by CEA-679-C Part B (see the Reference List in the Informative Annex). Then, when the POD Module requires setting up a connection with the cable headend, datagrams are sent to the telephone modem via the CPU interface as defined by CEA-679-C Part B.

4.4 Two-way Networks with DOCSIS

The configuration shown in Figure 4.4-1 applies where a DOCSIS capability exists in the Host.

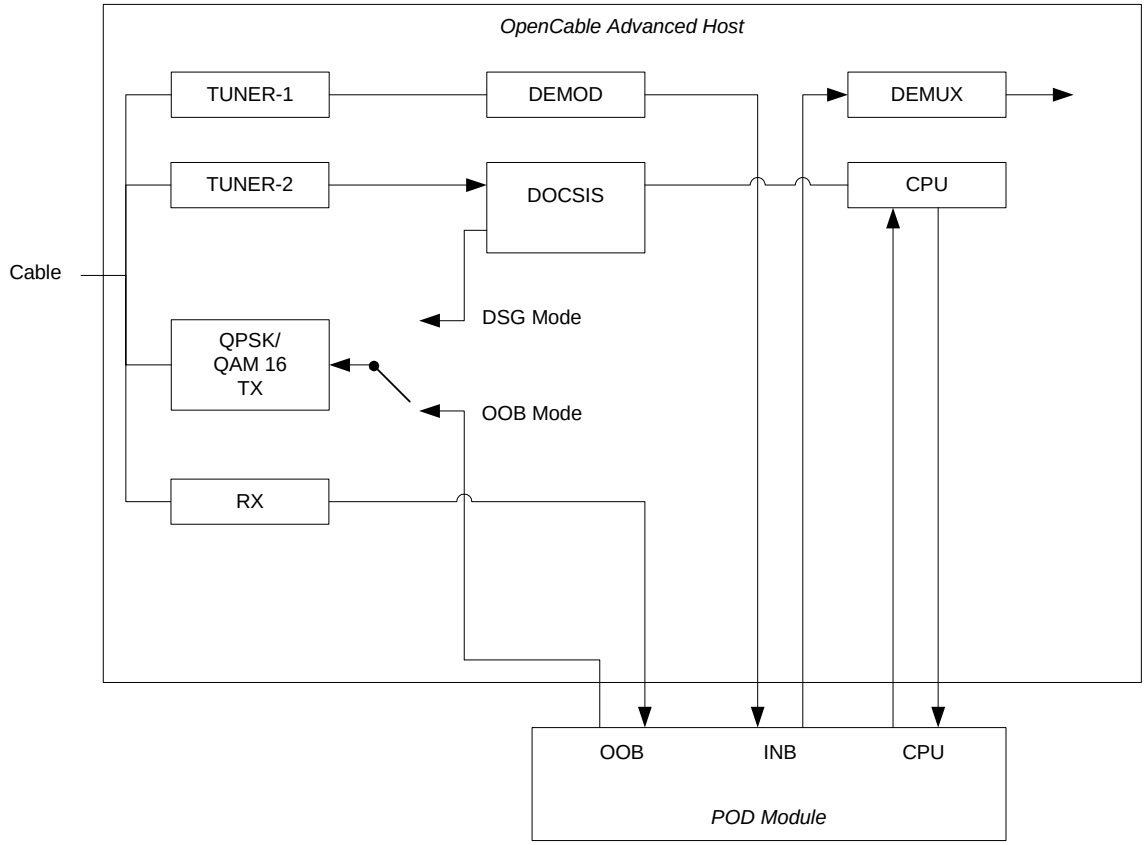

Figure 4.4-1 - System with DOCSIS Two-way Network

In this configuration a single upstream transmit path is shared between the POD Module and the DOCSIS modem. In order to prevent conflict between the DOCSIS upstream and the OOB upstream the system will operate in one of two modes.

- OOB mode – The downstream Conditional Access Messages and network management messages will be delivered to the POD Module via the QPSK receive interface on the POD Module using, e.g., SCTE 55-1, SCTE 55-2, or other agreed OOB specification. The upstream Conditional Access Messages and network management messages will be transmitted from the POD Module via the QPSK transmit interface on the POD Module using, e.g., SCTE 55-1, SCTE 55-2, or other agreed OOB specification.

- DSG mode – The downstream Conditional Access Messages and network management messages will be delivered to the POD Module by the Extended Channel using the DSG Service_type using the DOCSIS downstream in

accordance with the DOCSIS Set-top Gateway (DSG) Specification [1]. The upstream Conditional Access Messages and network management messages will be transmitted from the POD module via IP over the DOCSIS upstream channel using the Extended Channel. The DOCSIS bi-directional channel can be used by any applications running in the Host, simultaneously with the POD module's communication with the headend via the Extended Channel using DOCSIS. The use of the Extended Channel by the POD module for IP flows does not change DSG usage of the DSG Service_type on the Extended Channel.

The mode used is based on whether the DOCSIS Set-top Gateway is supported by the network. The POD informs the Host which of these modes is to be used as detailed later in this specification.

5 EXTENDED CHANNEL DATA FLOWS

5.1 Internet Protocol Flows (Informative)

The Extended Channel supports delivery of IP packets across the POD interface. Both unicast (point to point) and multicast (point to multipoint) addressing shall be supported by this protocol. If the Host is in OOB mode, then the POD module shall service the IP flow via utilization of the Host's RDC and shall supply the Host with an IP address. On request of a "new flow request" from the Host, the POD Module shall respond to the request to open the flow by obtaining an IP address for use by the Host. That IP address shall be returned in the "new flow confirmation" message. Informative Note: The POD is not required to grant a request for service type IP Unicast when requested by the Host.

In DSG mode, the Card resides at the Network Layer and the Host shall utilize its eCM to provide the Data Link Layer to the underlying DOCSIS network. When the Card wishes to utilize the DOCSIS network to transfer IP datagrams upstream, it shall first submit a "new flow request" to the Host to establish an IP flow to transfer datagrams between the Card and the Host's eCM interface. The Card shall submit its MAC address in its request to the Host for an IP flow.

If the Host grants the new IP flow request, then the Host shall utilizes DHCP to acquire an IP address for the Card, and shall send this information, along with the DOCSIS maximum transmission unit (MTU) (1500 bytes for IP datagrams) to the Card in a new flow confirmation. The Host shall then open an IP flow to the Card over the Extended Data Channel.

The Host shall utilize the MAC address provided in the Card's IP flow request to filter Ethernet frames from the eCM that are intended for the Card. The Host shall extract all unicast IP datagrams from Ethernet frames addressed to the CableCARD's MAC address and shall forward them over the Extended Data Channel to the CableCARD.

The Host shall utilize the Extended Data Channel's IP flow to forward IP datagrams it receives over the eCM interface on behalf of the Card. The Host shall not forward the Card any datagrams received over other interfaces (e.g. Ethernet port, USB port, etc.).

The Host shall forward all IP datagrams received from the Card to the eCM interface. The Host shall not forward any IP datagrams received from the Card to any other interface, including but not limited to: IEEE-1394, Ethernet, USB, 802.11a/b/g/n/x, Multimedia Over Coax Alliance (MoCA), etc. The Host shall resolve the destination MAC address of the IP datagrams that it receives from the Card and shall apply the appropriate MAC addresses to the Ethernet frames it sends upstream. If an established IP type of flow becomes unavailable for any reason, the device that has granted the flow shall report that fact to the one that has requested the flow. The "lost flow indication" transaction shall be used to report this type of event. One example case where a flow may become unavailable is due to a change in the state of the eCM that may have resulted from a change via SNMP to the eCM's operational state.

5.2 Flow Examples—QPSK Modem Case (Informative)

Figure 5.2-1 diagrams a POD-Host interface in which four flows have been set up. In this example case, the POD provides a full-duplex modem function for the benefit of the Host (as well as itself).

The rounded rectangle boxes represent applications. In this example, the Host has a Navigation application that receives Service Information data on the Extended Channel via the POD interface (#1). The Host has opened up three flows to receive MPEG data from the POD, and has supplied different PID values for filtering for each. The navigation function (#1) uses two SI flows in the example, and another application (#2) uses the third flow. The Host also has a web browser application (#3) and a Video On Demand (VOD) application (#4).

In Figure 5.2-1, the types of services that the POD is required to support are shown with black arrows. As shown in the figure, three flows delivering MPEG table sections are required. Flows that may be available at the option of the supplier of the POD are shaded gray. In the figure, the POD supports an IP flow, but a compliant POD can choose not to support the IP service type.

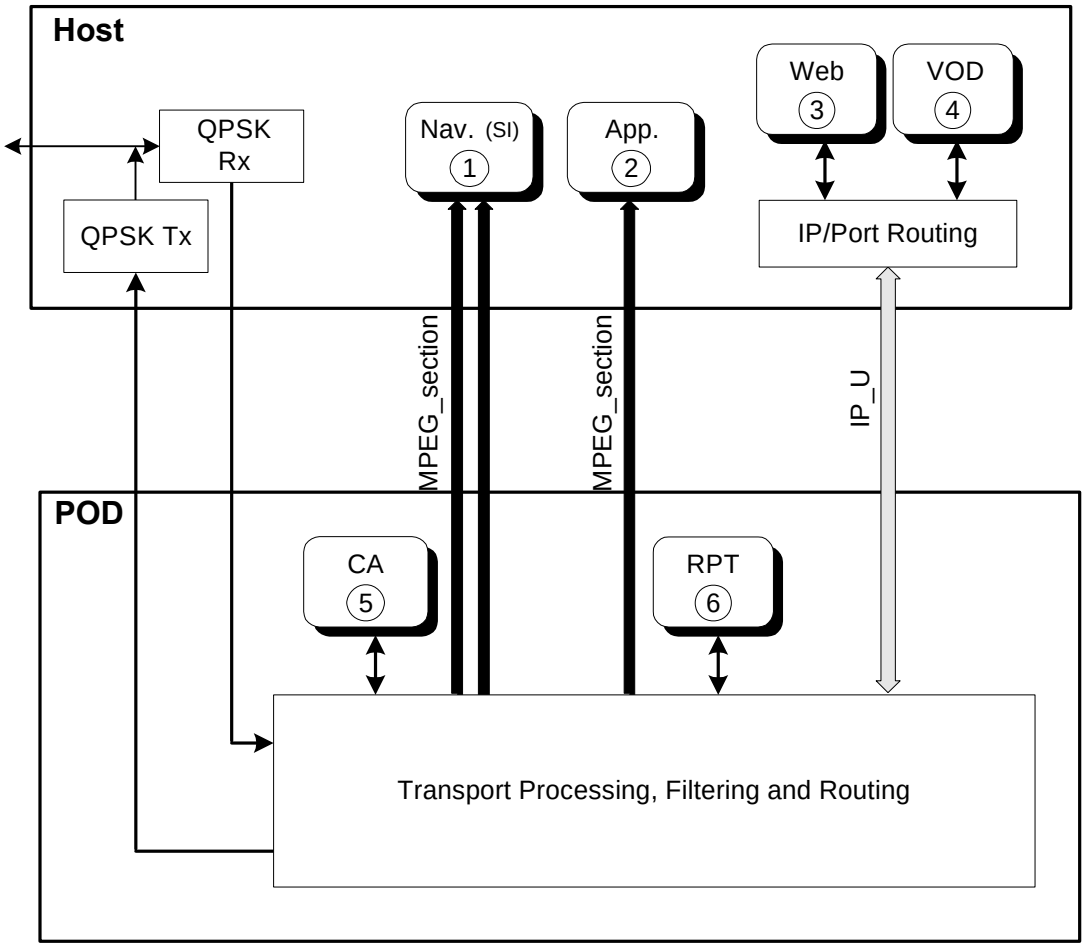

Figure 5.2-1 Flow Examples - QPSK Modem Case

The POD includes two applications of its own. The Conditional Access process (#5) receives data via downstream QPSK. The POD includes a pay-per-view reportback function (#6).

Note that none of these POD applications use flows that travel across the POD interface.

5.3 Flow Examples— High Speed Host Modem Case DSG Mode

In the next example case, the Host includes a High Speed Host Modem. Figure 5.3-1 diagrams a POD-Host interface in which five flows have been set up. When a Host includes a High Speed Host Modem, the Host is required to support at least one flow of service type IP Unicast (IP_U). As before, the POD must support three MPEG section flows if the Host requests them.

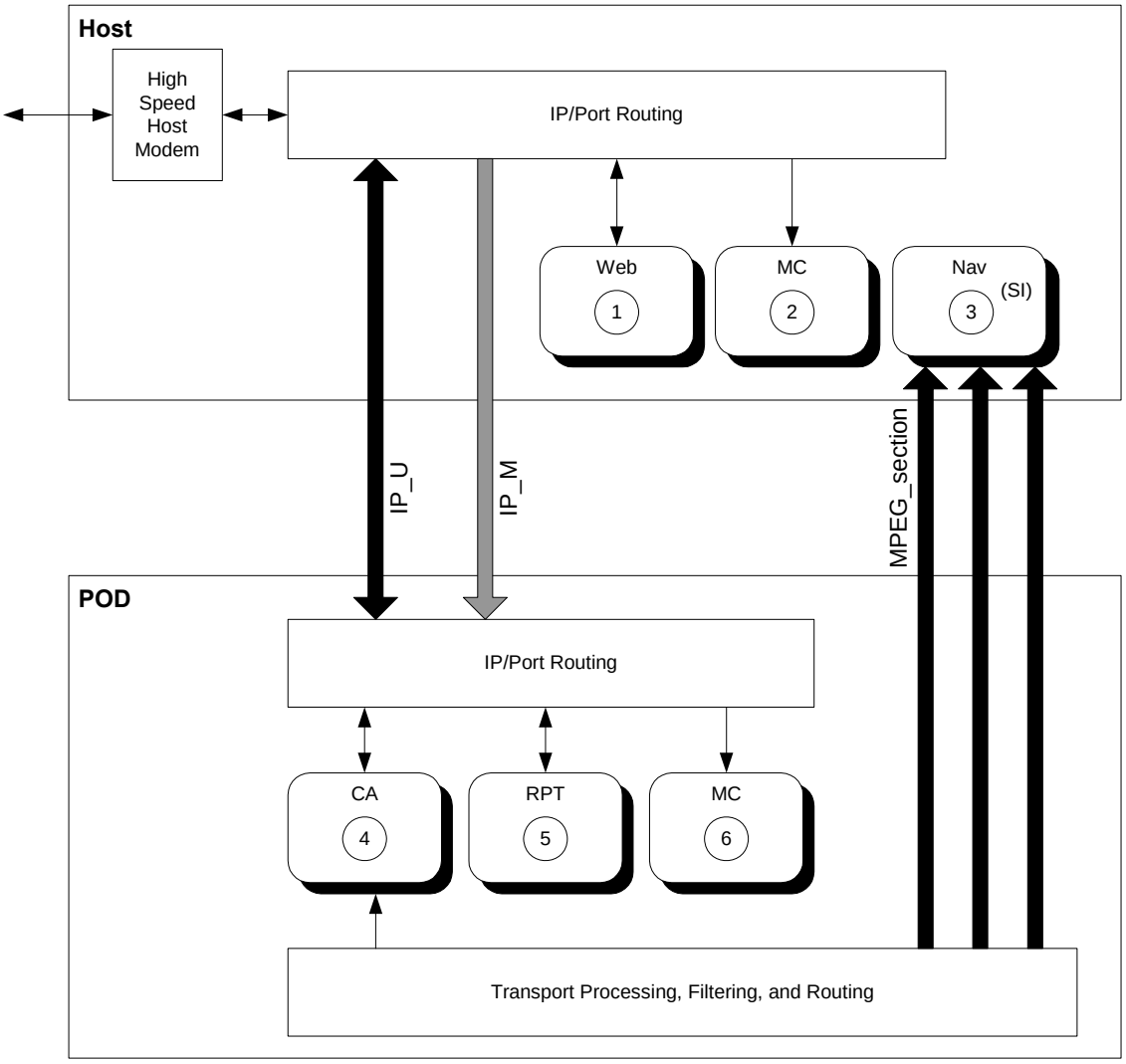

Figure 5.3-1 Flow Examples - High Speed Host Modem Case

In this example, the Host has a web browser application (#1), some application that uses multicast addressed packets (#2) and a Navigation application (#3) that receives Service Information data on the Extended Channel via the POD interface via three separate flows.

The Navigation application can open three different simultaneous flows, specifying different PID values for each. For example, it might set one to the base PID that carries SI network data including the Master Guide Table, Virtual Channel Table and System Time. It can set a second one to point to a PID value where Event Information Tables for a specific time slot may be found, and another to collect associated Extended Text Tables (ETTs).

The POD includes three applications of its own. The Host routes IP packets to the POD applications based on IP address. For unicast packets, those that match the IP address assigned to the POD will be routed across the interface. For multicast

packets, those matching the multicast group address associated with a particular flow will be delivered.

The POD includes a pay-per-view reportback function (#5) that uses standard IP packets for data transport. Finally, the POD includes some application (#6) that has registered with the Host to receive multicast-addressed IP packets through the Host modem.

5.4 Summary of Extended Channel Flow Requirement (Normative)

Compliance with this standard requires Host and POD devices to support certain flows. Other types of flows may be supported at the discretion of the Host or POD. The following table summarizes the requirements.

Table 5.4-A Flow Requirements

Requestor	Data Direction	Service Type	Number of Concurrent Flows
Host	POD → Host	MPEG section	6 or more
If Host implements High Speed Host Modem or if OCAP is supported:			
POD	Host ↔ POD	IP Unicast	At least 1
If Host implements DSG:			
POD	Host → POD	DSG	1

5.5 System/Service Information Requirements (Normative)

The POD module shall supply System and Service Information across the HOST-POD interface, using ***Service_type MPEG***_section, as defined in Section 8.9.1 and ANSI/SCTE 65 2002. The set of MPEG-2 tables provided to support the navigation function in the terminal device shall conform to one or more of the profiles specified in ANSI/SCTE 65 2002.

Note: 1 (Informative) Profiles 1 through 5 are compatible with terminal devices deployed as of Jan 1, 2000. Terminal devices that are intended to be portable across the US will need to function with any of the six profiles of ANSI/SCTE 65 2002. For operational considerations of various profiles, see section A.3 in ANSI/SCTE 65 2002

5.6 Emergency Alert Requirements (Normative)

The POD module may receive Emergency Alert messaging on either the FAT channels or the Out-of-Band Forward Data channel (OOB FDC). The EAS message syntax is compatible with MPEG-2 transport and is defined in SCTE 18 2002. For

FAT channel transmission, the EAS message shall appear in transport packets with the same PID as those used for Service/System Information (SI) and shall be transmitted by the POD to the HOST. The table ID for the EAS message is 0xD8 as defined in SCTE 18 2002. For out-of-band (OOB) transmission, EAS messages shall be processed by the POD module and shall be transmitted over the Extended Channel according to SCTE 18 2002.

6 PHYSICAL INTERFACE (NORMATIVE)

6.1 PC Card Compliance

6.1.1 POD Module Port Custom Interface (0341h)

The POD Module interface is registered to the PC Card Standard as the POD Module Custom Interface with the interface ID number (STCI_IFN) allocated to equal hexadecimal 341. In case the Host is not capable of operating with the POD Module, the Host shall ignore the POD Module.

The POD Module shall present the 16-bit PC Card memory-only interface following the application of VCC or the RESET signal. When operating in this configuration, D7-D0 are retained as a byte-oriented I/O port, and the capability to read the Attribute Memory is retained.

Only two address lines are required for four Bytes of register space. Pin CE2# is assigned to select the Extended Channel function required for the POD Module CPU interface to enable the access to the Extended Channel resource. Pin IOIS16# is never asserted.

The Host-POD interface 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.

6.1.2 Power Management

In order to remain compliant with the PC Card standard and to simplify the Host and POD Module implementations, and regardless of the powering state of the Host (i.e., active or standby), the following power management features are required.

- The Host shall permanently supply 3.3V on the VCC pins. The Host shall be capable of supplying up to a maximum of 1 amp total on the VCC pins (500 ma each) at 3.3 VDC per POD Module supported.

- The Host shall supply 5V on the VPP pins if requested by the POD Module Card Information Structure. The Host shall be capable of supplying up to 250 ma total on the VPP pins (125 ma each) at 5 VDC per POD Module supported.

- The Host shall continuously supply 3.3V on the VPP pins upon Host power-up and also when a POD module is not installed. When a POD module or a PC Card is installed, if the voltage sense pins are set as required per the Host-POD Interface Specification, the Host shall supply 5 V on the VPP pins only if requested by the POD Module Card Information Structure. Otherwise, the Host shall continue to supply 3.3 V on the VPP pins while the POD module/PC Card is installed. Upon removal of a POD module or a PC Card, the Host shall revert to or continue to supply 3.3V on the VPP pins. The Host shall be capable of supplying up to 250 ma total on the VPP pins (125 ma each) at 5 VDC per POD module supported.

- The Host shall not be required to support the separate nominal voltage parameter descriptors in the power descriptor structures for VPP1 and/or VPP2.

- The POD module shall only support the value of 0x2 in the Power field of the Feature Selection Byte (TPCE_FS) and the associated parameter descriptor according to section 3.3.2.3 of PC Card Standard, Volume 4 if the POD module requires a switched nominal voltage level of +5V on the VPP lines.

- There is no standby power mode for the POD module.

- The POD Module shall draw no more than 2.5 watts averaged over a period of 10 seconds.

- The Host OOB Receive circuitry must continue to operate in all powering states of the Host.

- The Host shall support hot insertion and removal of the POD Module.

- The POD Module shall implement the mechanical Low Voltage Keying.

- The POD Module shall force VS1 (pin 43) to ground and VS2 (pin 57) to high impedance until it switches to the POD Module Custom Interface mode.

- The POD Module shall support 3.3V hot insertion.

- The POD module does not have to meet the requirement of section 4.12.2 of the PCMCIA Electrical Specification to limit its average current to 70 mA prior to the POD Personality Change (writing to the Configuration Option Register).

6.1.3 Pin Assignment

The following table shows the function of various PC Card signals when the POD Module Port custom interface mode is set to active in the Host.

Differences between CEA-679-C Part B and this Host-POD Interface Specification affect the A4 to A9 signals, which are now assigned to the OOB RF I/Os, and CE2#, which is used to access the Extended Channel. The MCLKO is provided on pin 14 to be fully PC Card compliant. This is a modification from CEA-679-C (Part B). Pin 57

remains the PC Card VS2# signal. Shaded entries in Table 6.1-A highlight the differences between CEA-679-C Part B and this specification.

Table 6.1-A PC Card Signal Definitions							
Pin	**Signal**	**I/O**	**Comment**	**Pin**	**Signal**	**I/O**	**Comment**
1	GND	DC	Ground	35	GND		Ground
2	D3	I/O		36	CD1#	O	
3	D4	I/O		37	MDO3	I/O	(D11)
4	D5	I/O		38	MDO4	I/O	(D12)
5	D6	I/O		39	MDO5	I/O	(D13)
6	D7	I/O		40	MDO6	I/O	(D14)
7	CE1#	I		41	MDO7	I/O	(D15)
8	A10	I		42	CE2#	I	Extended Channel
9	OE#	I		43	VS1#	O	
10	A11	I		44	IORD#	I	(RFU)
11	DRX	I	(A9)	45	IOWR#	I	(RFU)
12	CRX	I	(A8)	46	MISTRT	I	(A17)
13	A13	I		47	MDI0	I	(A18)
14	MCLKO	I/O	(A14)	48	MDI1	I	(A19)
15	WE#	I		49	MDI2	I	(A20)
16	IREQ#	O	(READY)	50	MDI3	I	(A21)
17	VCC	DC	3.3V	51	VCC	DC	3.3V
18	VPP-1	DC	Switched 5V	52	VPP-2	DC	Switched 5V
19	MIVAL	I	(A16)	53	MDI4	I	(A22)
20	MCLKI	I	(A15)	54	MDI5	I	(A23)
21	A12	I		55	MDI6	I	(A24)
22	QTX	I/O	(A7)	56	MDI7	I	(A25)
23	ETX	I/O	(A6)	57	VS2#	O	
24	ITX	I/O	(A5)	58	RESET	I	
25	CTX	I	(A4)	59	WAIT#	O	
26	A3	I		60	INPACK#	O	
27	A2	I		61	REG#	I	
28	A1	I		62	MOVAL	O	(BVD2)
29	A0	I		63	MOSTRT	O	(BVD1)
30	D0	I/O		64	MDO0	I/O	(D8)
31	D1	I/O		65	MDO1	I/O	(D9)
32	D2	I/O		66	MDO2	I/O	(D10)
33	IOIS16#	O	(WP)	67	CD2#	O	
34	GND	DC	Ground	68	GND	DC	Ground

Note: “I” indicates signal is input to POD Module, “O” indicates signal is output from POD Module

6.2 POD Module Identification

The Host has two different ways to recognize a POD Module: (1) at the application level, using the Application Info CEA-679-C Part B resource, or (2) at the physical level as defined by PCMCIA. .

In PCMCIA memory mode, the Host accesses the POD Module's Attribute Memory to read the Card Information Structure (CIS) on the even addresses (first byte at address 000h, second byte at address 002h, etc.). Since the POD Module interface is a PC Card Custom Interface the CIS must include a custom interface subtuple (CCST_CIF) that provides the interface ID number (STCI_IFN) defined by PCMCIA (hex341).

For a more explicit identification, the CIS also includes in the tuple CISTPL_VER_1, the field name of the product of subtuple TPLLV1_INFO defined as "OPENCABLE_POD_MODULE".

This information in the CIS is mandatory to ensure backup operation in case of trouble when the CI stack is lost (e.g., power shut down, POD Module extraction).

6.3 Card Information Structure

The Card Information Structure (CIS) shall be readable whenever the SCTE POD module is powered, the SCTE POD module has been reset by the Host in accordance with section 4.12.1 of Volume 2, "Electrical Specification" of the PC Card Standard, the SCTE POD module is asserting the READY signal, and the POD Personality Change has not occurred. The CIS contains the information needed by the Host to verify that a POD module has been installed. . After the POD Personality Change, the CIS shall no longer be available.

The following table is the minimum set of tuples required for the POD Module.

Table 6.3-A CIS Minimum Set of Tuples
CISTPL_LINKTARGET
CISTPL_DEVICE_0A
CISTPL_DEVICE_0C
CISTPL_VERS_1
CISTPL_MANFID
CISTPL_CONFIG
CISTPL_CFTABLE_ENTRY
CISTPL_NO_LINK
CISTPL_END

6.4 Host-POD OOB Interface

This standard requires support for OOB signaling. This signaling is provided in one of two modes, Out of Band (OOB) Mode and DOCSIS Settop Gateway (DSG) Mode:

6.4.1 Out of Band (OOB) Mode

- The Host RF front-end specification provides the QPSK physical layer to support OOB (downstream and upstream) communications according to SCTE 55-1 or SCTE 55-2. The data link and media access control protocols for SCTE 55-1 or SCTE 55-2 are implemented in the POD Module.

The interface data rates are:

- Forward Receiver: 1.544/3.088 Mbps and 2.048 Mbps
- Reverse Transmitter: 772/1544 Ksymbol/s and 128 Ksymbol/s (i.e., 1.544/3.088 Mbps and 256 Kbps)

The transmit and receive interfaces for the Host-POD OOB Interface are shown in Figure 6.4-1 below. The receiver interface comprises a serial bit stream and a clock, while the transmitter interface comprises I and Q data, a symbol clock, and a transmit-enable signal. The clock signal should be transferred from the Host to the POD, as shown in Figure 6.4-1.

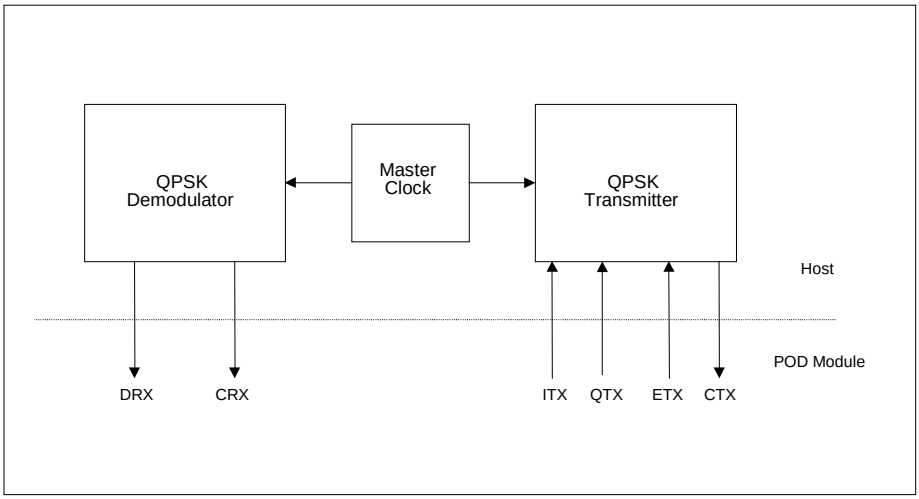

Figure 6.4-1 Host-POD Out-of-Band Interface

The interface symbols are defined below.

Table 6.4-A Transmission Signals for Host-POD Interface			
Signal	**Definition**	**Rates**	**Type**
DRX	RX Data	1.544/3.088 and 2.048 Mbps	I
CRX	RX Gapped Clock	1.544/3.088 and 2.048 MHz	I
ITX	TX I Channel	772/1544 and 128 Ksymbol/s	O
QTX	TX Q Channel	772/1544 and 128 Ksymbol/s	O
ETX	TX Enable	[n/a]	O
CTX	TX Gapped Symbol Clock	772/1544 and 128 KHz	I

1. DRX/CRX

DRX – The DRX data directly from the Host FDC QPSK receiver.

CRX – Gapped clock is a clock signal in which some of the clock cycles are missing, creating an artificial gap in the clock pattern. The clock rate is one of 1.544, 3.088 or 2.048 MHz

2. ITX, QTX – Differential encoding shall take place within the POD module. The Host shall map ITX,QTX directly to the phase states shown in Figure 5.4-3 below.

3. ETX – ETX is an output from the POD Module and an input to the Host. It is defined to be active high. When ETX is inactive, the values of ITX and QTX are not valid and the upstream transmitter shall not transmit such values. When ETX is active, the values of ITX and QTX are both valid and the upstream transmitter shall transmit these values.

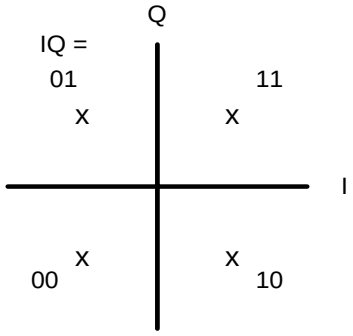

Figure 6.4-2. Phase States for Mapping ITX and QTX OK

6.4.2 DOCSIS Settop Gateway (DSG Mode

The Host DOCSIS cable modem provides the physical, data link, and media access control protocols. Unlike the OOB mode, the data link and media access control protocols for SCTE 55-1 or SCTE 55-2 are bypassed in the POD Module. 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 POD Module via IP over the DOCSIS upstream channel using the Extended Channel.

The interface data rates are:

- Downstream direction: 2.048 Mbps
- Upstream direction: Limited by DOCSIS return channel capacity

The first two bytes of the 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 POD is reliable. It is the responsibility of the POD vendor to implement error detection in the DSG encapsulated data. The POD 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 POD 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 5.4-2 below shows how the DSG packets are transported across the POD/Host interface with and without removal of the header bytes. Prior to transmission across the POD/Host interface, the CRC of the DSG packet received from the eCM is removed, then optionally header bytes may be removed in order from the Ethernet header through the IP header and the UDP header resulting in the removal of X header bytes where X is defined by the POD as per the remove_header_bytes of the set_dsg_mode() APDU (note that X may be zero, thus no header bytes are removed). A two byte field containing the DSG byte count of the resulting data payload is prepended to the remaining frame and transmitted across the POD/Host interface.

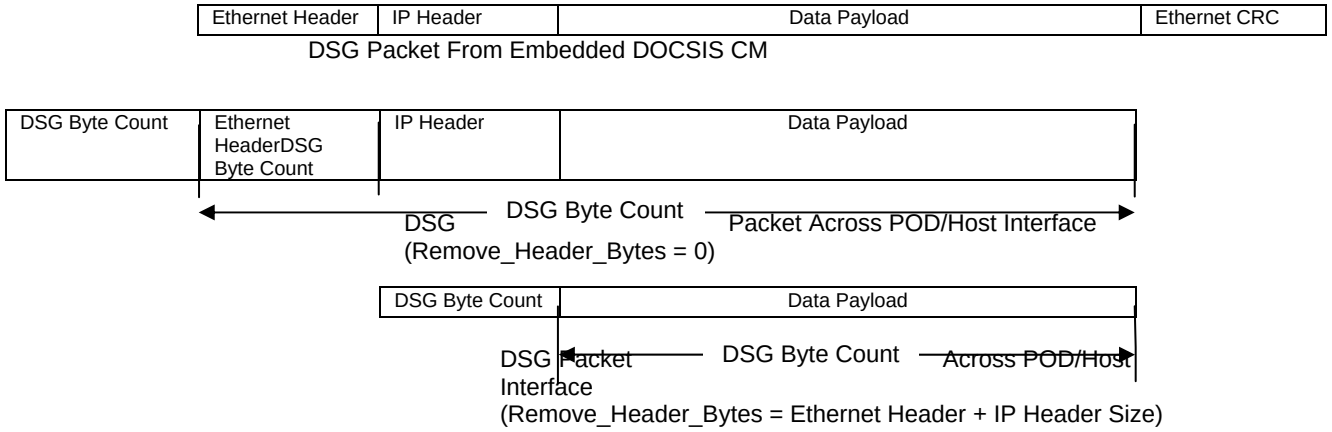

Figure 6.4-3. DSG Packet Format Across POD/Host Interface

6.4.3 Timing and Voltage Parameters

All POD signal requirements and timing requirements shall comply with Table 6.4-B and Figures 6.4-2 and 6.4-3, and shall be measured with no less than the maximum load of the receiver as defined in table 4-19 of [10].

The PC Card A7, A6, and A5 pin definitions have been modified to QTX, ETX, and ITX. These pins will be driven by the POD and will have Data Signal characteristics per table 4-19 of [10]. Additionally, the signals MOVAL and MOSTRT will be driven by the POD and will have Data Signal characteristics per table 4-19 of [10]. The remaining signals follow the signal type assignments as listed in table 4-19 of [10].

All signal voltage levels are compatible with normal 3.3V CMOS levels.

Table 6.4-B POD Signal Parameters						
Parameter	**Signal**	**Unit**	**Min**	**Type**	**Max**	**Conditions**
Frequency	CTX	kHz			3088	
Frequency	CRX	kHz			3088	
Hold (T_{HCTX})	CTX	ns	250			Note 1,2
Hold (T_{HCRX})	CRX	ns	250			Note 1,2
Delay (t_D)	ETX, ITX, QTX	ns	5		180	Note 1,2
Set-up (Tsu)	DRX	ns	10			From time signal reaches 90% of high level (rising) or 10% of high level (falling) until CRX mid-point transition
Hold (Th)	DRX	ns	5			From CRX mid-point transition until signal reaches 10% of high level (rising) or 90% of high level (falling)

Note1: Refer to Figure 6.4-2 POD Return Data Channel Timing Diagram.

Note2: AC Timing is measured with Input/Output Timing Reference level at 1.5V.

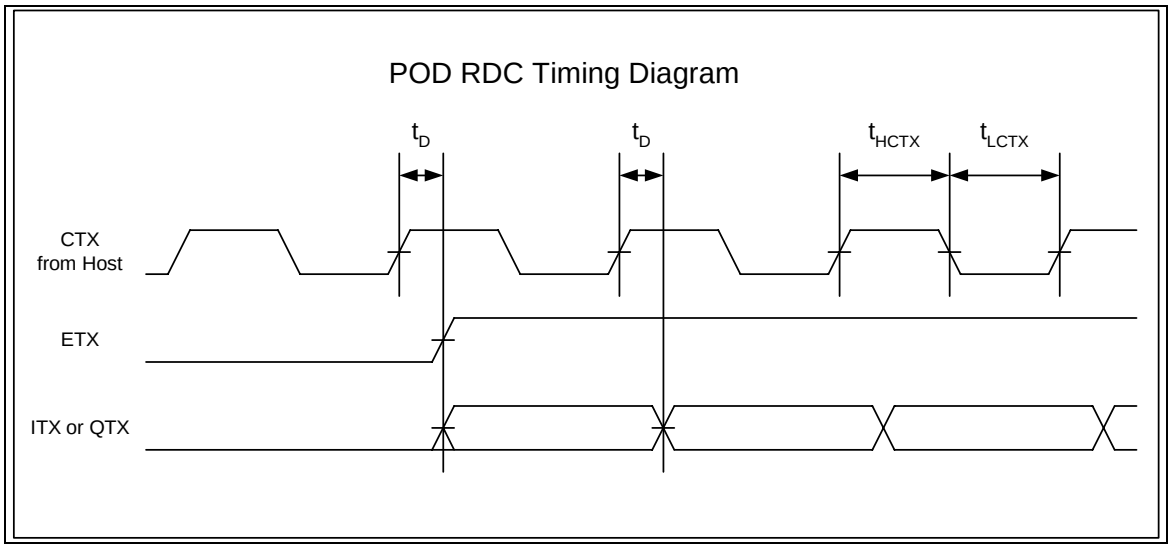

Figure 6.4-3 POD Output Timing Diagram

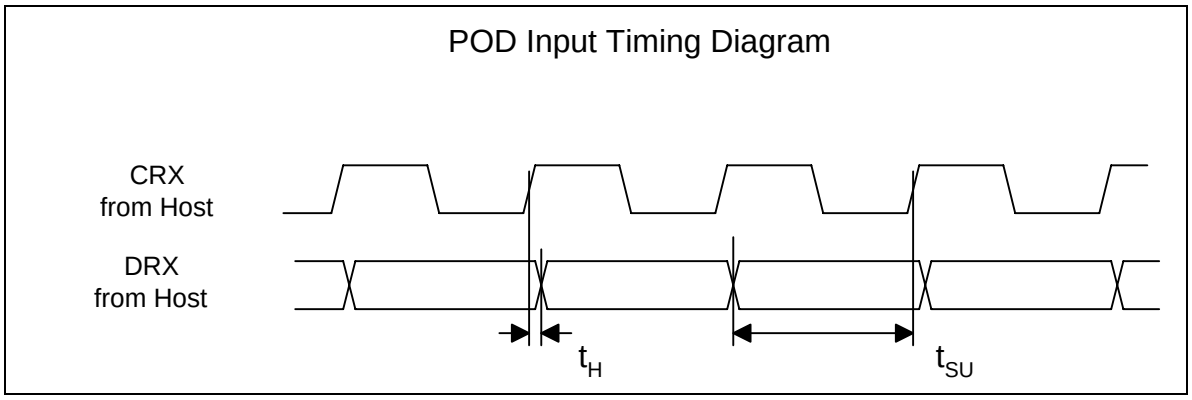

Figure 6.4-4 POD Input Timing Diagram

6.5 CPU Interface

With OOB traffic included, the POD Module requires more bandwidth and connections on the CPU Interface than are supported by the Data Channel alone. Two communication paths shall share the same pins on the PC Card connector.

Data Channel – This channel is compliant with the Command Interface protocol of CEA-679-C Part B, plus the interrupt mode extension. POD Module applications shall use this path when they require support from Host resources.

Extended Channel – This second communication only includes physical and link layers. The purpose of the ***Extended Channel*** is to provide a communication path between the POD Module and the Host such that applications in one (e.g. Host, POD Module) can communicate with the headend via a link layer or modem function in the other (POD Module, Host respectively). Whereas the content and format of the messages for the ***Data Channel*** are well defined, the content and format of the messages for the ***Extended Channel*** are application specific.

Depending on whether the POD Module or the Host is acting as the modem (or link device), the ***Extended Channel*** has a reversible function as described in figure 6.5-1 and figure 6.5-2.

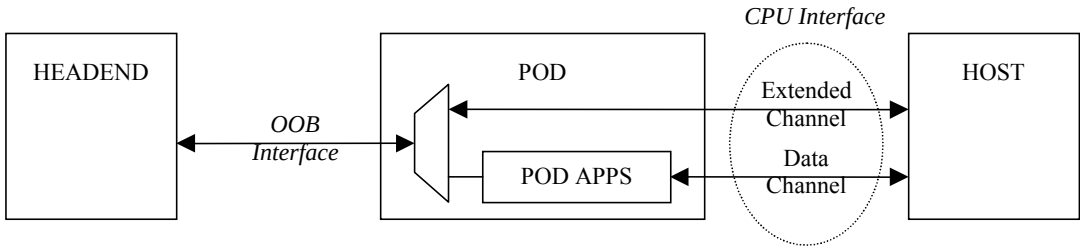

Figure 6.5-1 Modem-in-the-POD Module System Overview

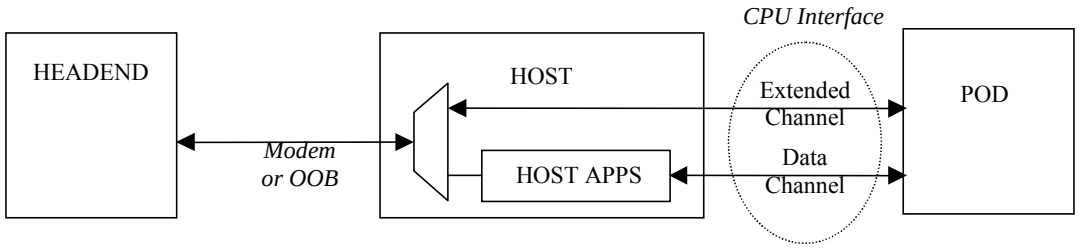

Figure 6.5-2 Modem in-the-Host System View

When the ***Data Channel*** is physically activated by CE1# (Card Enable 1) as defined by CEA-679-C Part B, the ***Extended Channel*** is enabled by CE2# (Card Enable 2), which is not used by CEA-679-C Part B.

The ***Extended Channel*** includes the same type of registers as defined by CEA-679-C Part B for the Command Interface. The POD Module enables access to the ***Extended Channel*** after the initialization phase. At this time, the CE2# signal interpretation begins, and the Extended hardware interface registers can be read and written. The signals mentioned in the table below are all inputs for the POD Module. The registers depicted in figure 6.5-3 are part of the POD Module.

Table 6.5-A Extended Interface Registers

Extended Interface Reg.	REG#	CE2#	CE1#	A1	A0	IORD#	IOWR#
Standby mode	X	H	H	X	X	X	X
Ext_Data Write	L	L	H	L	L	H	L
Ext_Data Read	L	L	H	L	L	L	H
Ext_Command Write	L	L	H	L	H	H	L
Ext_Status_Reg. Read	L	L	H	L	H	L	H
Ext_Size (LS) Write	L	L	H	H	L	H	L
Ext_Size (LS) Read	L	L	H	H	L	L	H
Ext_Size (MS) Write	L	L	H	H	H	H	L
Ext_Size (MS) Read	L	L	H	H	H	L	H

The ***Extended Channel*** has its own data buffer that may have a different size than the ***Data Channel*** buffer.

Since there are two physical channels (data channel and extended channel), the behavior of the interface is defined in such a way that when the Host sets the RS_flag on either channel, the interface is reset for both channels. Therefore, if the Host sets an RS_flag after detection of an error condition, it should set the RS-flag for both channels.

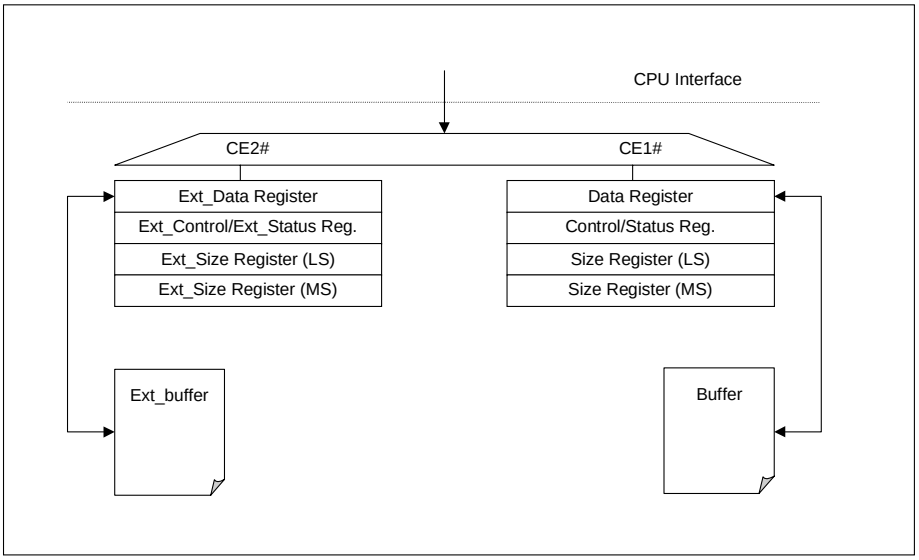

Figure 6.5-3 Map of Hardware Interface Registers

6.5.1 Control Register Modification

The following extension to the CEA-679-C Part B Command Interface shall be used in order to facilitate the interrupt mode over the ***Data Channel*** and the ***Extended Channel***.

The DA & FR bits of the Status Register should be gated onto the IREQ# line by two new Interrupt Enable bits for the Control Register: DAIE (bit 7) and FRIE (bit 6) respectively. The Control register now becomes:

Table 6.5-B Control Register Definitions

Bit	7	6	5	4	3	2	1	0
	DAIE	FRIE	R	R	RS	SR	SW	HC

RS, SR, SW and HC retain their function, as described in CEA-679-C Part B specification.

When set, DAIE allows any assertion of the DA bit in the Status register also to assert IREQ#.

When set, FRIE allows any assertion of the FR bit in the Status register also to assert IREQ#.

When IREQ# is asserted, the Host shall first check the data channel, and then the extended channel to determine the source of the interrupt.

6.5.2 Status Register Modification

The following extension to CEA-679-C Part B Status Interface shall be used in order to allow the POD to request the initialization process to occur. A new status bit called the Initialize Interface Request (IIR) is added to bit 4 of the Status Register to allow the POD Module to request that the interface be re-initialized. This bit exists in both the data channel and extended channel. When the POD module sets the IIR flag, the POD must also reset the IIR flag when the RS flag is set.

Table 6.5-C Status Register Definitions

bit	7	6	5	4	3	2	1	0
	DA	FR	R	**IIR**	R	R	WE	RE

6.6 Copy Protection on the FAT Channel

Copy protection shall be provided for 'high value' content delivered in MPEG transport streams flowing from the POD to the Host. Such protection, including scrambling of content from POD to Host and authenticated delivery of messages through the CPU interface for permitted use of 'high value' content, is defined in the POD Copy Protection System specification (see ANSI/SCTE 41 2001).

6.7 Host-POD Interface Initialization

This section defines the interface initialization procedure between the POD module and the Host.

6.7.1 Descriptions

Initialization is a very general term. The following are definitions of the how the term initialization is used in this section.

6.7.1.1 Interface Initialization Definition (Informative)

Any computing device must go through an initialization phase whenever a reset condition occurs, such as when initial power is applied, manual reset, or an unrecoverable software error condition occurs. What is covered in this section is the initialization of the interface between the host and the POD module. This is defined to be the ***interface initialization***.

6.7.1.2 POD Personality Change Definition (Informative)

The host and POD module shall initialize to the PCMCIA interface and will, at a particular point in the sequence, change to the POD interface. This point is defined as the ***POD Personality Change***.

6.7.1.3 Reset Definition

There are two possible resets that can occur in the POD interface, a hard reset (called PCMCIA reset) and a soft reset (called POD reset).

6.7.1.3.1 PCMCIA Reset

The PCMCIA reset is defined to be one in which the Host shall bring the RESET signal to the POD module active. The interface shall revert to the PCMCIA interface including no longer routing the MPEG data stream through the POD module. Obviously this will cause problems to the viewer and should be avoided except in the case that a catastrophic failure has occurred in the POD module or in the interface between the Host and the POD module.

6.7.1.3.2 POD Reset

The POD reset is defined to be when Host sets the RS bit in the control register anytime after the POD personality change has occurred. The Host shall set the RS bit in both the data channel and extended channel. The POD module shall detect whether the RS bit has been set in either channel and, if so, shall close all open sessions and transport connections and operation shall revert to that of just after the POD personality change. This reset shall prevent the change of routing of the MPEG data stream, thereby preventing the viewer from noticing any problems unless the video/audio stream being viewed is scrambled. Since the conditional access session is closed, the POD module shall cease descrambling the data stream until a new session is opened and the appropriate APDU transmitted to the POD module.

The POD reset should occur when the Host detects an error in the POD module interface or the POD module has set the IIR flag (see below).

6.7.1.3.3 Initialize Interface Request Flag

A status bit called the **Initialize Interface Request (IIR)** flag is included in bit 4 of the status register to allow the POD module to request that the interface be re-initialized. This bit exists in both the data channel and extended channel. When a condition occurs that the POD module needs to request an interface initialization, it shall set both IIR bits. Upon recognition of the IIR flag being set, the Host shall implement a POD reset. The POD module will clear the IIR flag when the RS bit is set. To further ensure reliable interoperability, POD modules shall be prohibited from sending LPDUs to the Host after setting the IIR bit and prior to recognizing an active RS bit.

6.7.1.3.4 Detailed POD Reset Operation

The following flowchart (POD Reset Sequence) is the required implementation of the POD RS operation. LPDUs shall not be transmitted until the completion of the POD Reset sequence.

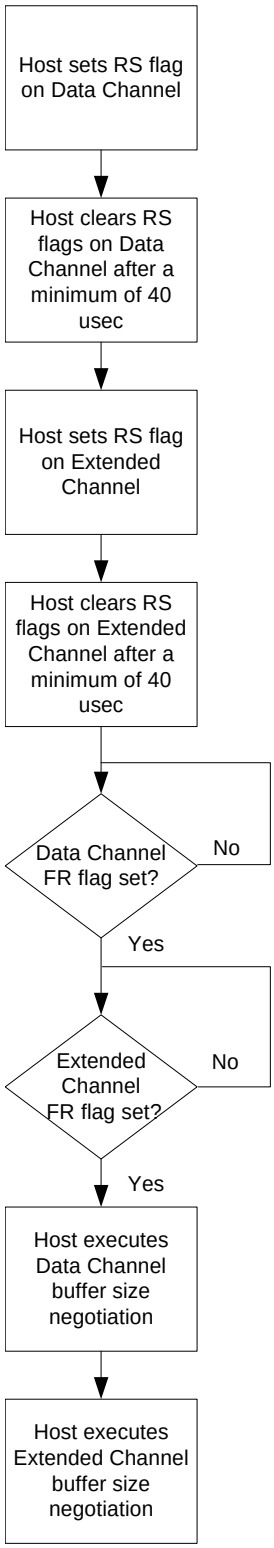

Figure 6.7-1 POD RS Operation

6.7.2 Configuration Option Register (Normative)

The Configuration Option Register (COR) in the POD module is only accessible prior to the POD Personality Change (see appendix D.2). After the POD personality change, the COR is no longer available. Any relevant configuration data must be transferred via the data or extended channels and is not covered in this document.

By writing the COR with the value defined by the Configuration-Table index byte, TPCE_INDX, from the CISTPL_CFTABLE_ENTRY tuple, the Host configures the module into the POD mode, thus causing the POD Personality Change.

6.7.3 Initialization Conditions

There are 4 possible conditions that can cause the PCMCIA interface initialization phase. Please see section 6.7.4 of this document for detailed operation. They are:

1. The Host and POD module are powered up at the same time. After both have performed their internal initialization, then the interface initialization will begin.

2. Host has been powered and in a steady state. A POD module is then inserted. After the POD module has performed its internal initialization, the interface initialization phase will begin.

3. The Host has performed a reset operation for some reason (spurious signal, brownout, software problem, etc.) that has not caused the POD module to reset. The Host shall go through its initialization and then shall perform a PCMCIA reset on the POD module. After the POD module has performed its internal initialization, then the interface initialization shall begin.

4. The POD module has performed a reset operation for some reason (spurious signals, software problem, etc.) that has not caused the Host to reset. The Host shall incorporate the timeout detection and will thus detect a timeout and will perform a POD reset.

6.7.4 OOB Connection and Disconnection Behavior

If a POD module is not connected to the Host, the OOB transmitter in the Host shall not operate. Upon connection of a POD module, the Host shall initiate, with the POD module, the low-level personality change sequence defined in Section 6.7.5 of this document. If successful, the Host shall then activate the OOB transmitter as instructed by the POD module.

The OOB receiver in the Host shall be connected only to the POD module interface.

6.7.5 Low Level Step by Step POD Personality Change Sequence

The POD Personality Change covers the detection of the POD module and the transition to the POD interface. A step-by-step operation for the interface initialization of the physical layer from the POD module's viewpoint is defined below.

1. The POD module is inserted or already present in a Host.

2. Please refer to section 4.12.1 of "PC Card Standard, Volume 2 Electrical Specification, March 1997" for timing diagrams and specifications.

- **Power-up:** Power is applied to the POD module with the RESET signal in a high-Z state for a minimum of 1 msec after VCC is valid (section 4.4.20 of the PC Card Electrical Specification). The POD module's READY signal (pin 16) shall be inactive (logic 0) within 10 usec after the RESET signal goes inactive (logic 0), unless the POD module will be ready for access within 20 msec after RESET goes inactive. Note that at this time the POD module shall only operate as an unconfigured PCMCIA module.

- **PCMCIA Reset:** The RESET signal goes active for a minimum of 10 usec. The POD module's READY signal (pin 16) shall be inactive (logic 0) within 10 usec after RESET goes inactive (logic 0), unless the POD module will be ready for access within 20 msec after RESET goes inactive. Note that at this time the POD module shall only operate as an unconfigured PCMCIA module.

3. After a minimum of 20 msec after RESET goes inactive (section 4.4.6 of "PC Card Standard, Volume 2 Electrical Specification, March 1997"), the Host shall test the POD module's READY signal. It shall not attempt to access the POD module until the READY signal is active (logic 1).

4. After the POD module has completed its PCMCIA internal initialization, it shall bring the READY signal active. At this time, all of the interface signals are defined by the PC Card interface standard for Memory Only Card interface (Table 4-1 of "PC Card Standard, Volume 2 Electrical Specification, March 1997"). The POD module must bring READY active within 5 seconds after RESET goes inactive (section 4.4.6 of "PC Card Standard, Volume 2 Electrical Specification, March 1997").

5. The Host shall read the Configuration Information Structure (CIS) available in the attribute memory to determine that the device is POD module, what version is used, and any other pertinent information. This data is outlined in section A.5.6 of CEA-679-C Part B with the revisions in section 3.1 of this document.

6. The Host shall read all the CCST_CIF subtuples to verify that the SCTE interface ID number (STCI_IFN) is present (0x341). (Informative Note--If it is not present, this means that a different PCMCIA module has been inserted which is not

capable of operating with the SCTE format, however, it may be capable of operating as a NRSS-B module (CEA-679-C Part B).)

7. The Host shall then write into the COR the value read in TPCE_INDX. Following this write cycle, the Host shall switch the address signals A4-A8 to the OOB interface signals and the inband transport stream signals. The Host must implement a pull-down resistor on the ETX signal to prevent spurious operation of the transmitter. It must also implement a pull down resistor on the MCLKO signal to prevent invalid inband transport data from being received prior to the POD's personality change.

8. At a minimum of 10 usec after the COR write signal, the POD module shall switch to the OOB interface signals and the inband transport stream signals.

9. In the event that the POD module requires additional initialization, it shall not bring the FR bit in the status register active until it is ready to begin communications with the Host.

10. This completes the physical link layer initialization.

The following diagram helps define this operation.

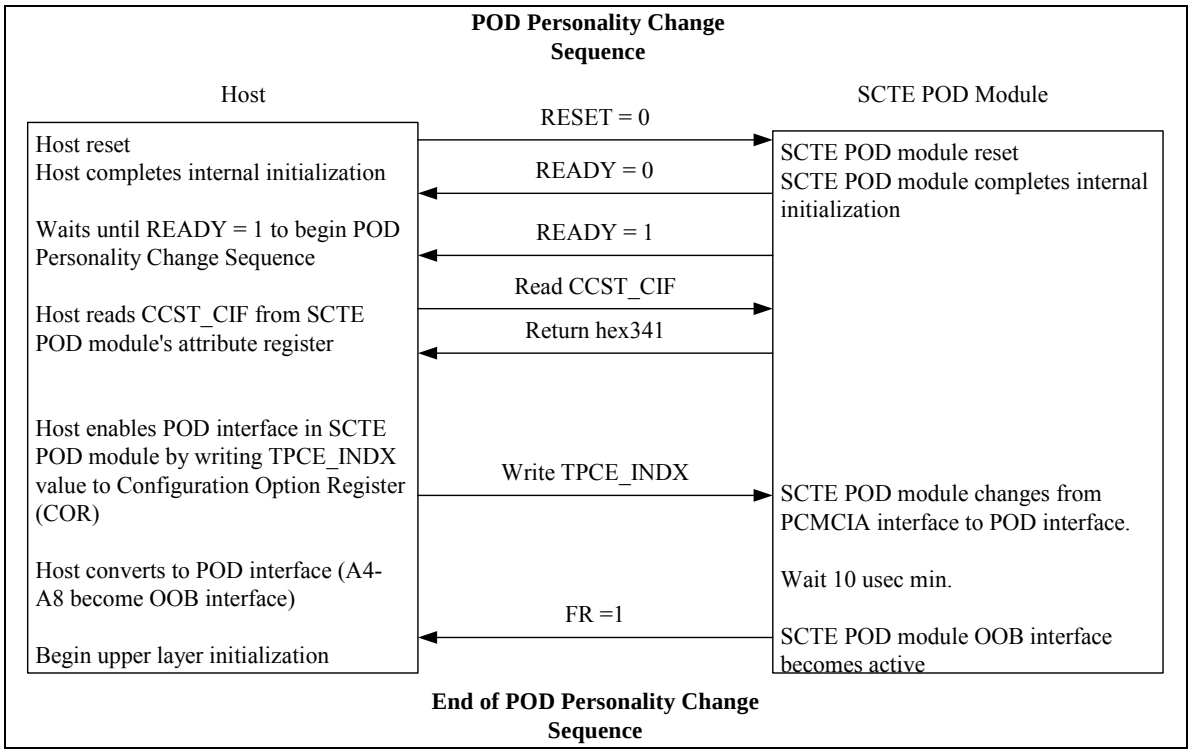

Figure 6.7-2 POD Personality Change Sequence

6.7.6 Initialization Overview

The following sections provide a description of the initialization procedure that shall occur between the POD module and the Host.

6.7.6.1 Physical Layer Initialization

The physical layer initialization covers the buffer size negotiation of both the data and extended channels, and the initialization of the Host-pod transport layer and resource manager. The following shall be implemented in the order listed.

6.7.6.1.1 Data Channel Initialization

The data channel is initialized by the Host writing a '1' to the RS bit in the data channel Control/Status Register. After a minimum of 40 usec, the Host will write a '0' to the RS bit in the data channel Control/Status Register. The POD module shall clear out any data in the data channel data buffer and configures the POD interface so it can perform the data channel buffer size negotiation protocol. When the POD module is ready, it sets the data channel FR bit to '1'.

6.7.6.1.2 Extended Channel Initialization

The extended channel is initialized by the Host writing a '1' to the RS bit in the extended channel Control/Status Register. After a minimum of 40 usec, the Host will write a '0' to the RS bit in the extended channel Control/Status Register. The POD module shall clear out any data in the extended channel data buffer and configures the POD interface so it can perform the extended data channel buffer size negotiation protocol. When the POD module is ready, it sets the extended channel FR bit to '1'.

6.7.6.1.3 Data Channel Buffer Size Negotiation

The Data Channel buffer size negotiation is covered in sections 5.5 and A.2.2.1.1 of CEA 679 Part B (reference [1]). There are 2 buffers which must be negotiated, the POD buffer for the data channel and the Host buffer for the data channel. According to Section A.2.2.1.1. of [1], the minimum buffer size for the POD module is 16 bytes and the minimum buffer size for the Host is 256 bytes. The maximum size for both is 65,535 bytes.

Using the protocol called out in section A.2.2.1.1 of [1], the Host will read the POD module's data channel buffer size, compare the result to its data channel buffer size, and write the smaller of the two buffer sizes to the POD module's data channel. All future data channel transaction buffer sizes shall be at most this buffer size. Note that a data channel transaction's buffer size can be smaller than the negotiated buffer size.

6.7.6.1.4 Extended Channel Buffer Size Negotiation

The Extended Channel buffer size negotiation is the same as the data channel. Note that the buffer sizes of the data and extended channels do not have to be the same. The minimum buffer size for the POD module is 16 bytes and the minimum buffer size for the Host is 256 bytes. The maximum size for both is 65,535 bytes.

Using the protocol called out in section A.2.2.1.1 of [1], the Host will read the POD module's extended channel buffer size, compare the result to its extended channel buffer size, and write the smaller of the two buffer sizes to the POD module's extended channel. All future extended channel transaction buffer sizes shall be at most this buffer size. Note that a extended channel transaction's buffer size can be smaller than the negotiated buffer size.

6.7.6.2 Link Connection

The link connection (LPDU) is covered in section A.3.2 of [1]. No explicit initialization of the Link Layer is required.

6.7.6.3 Host-POD Transport Layer Connection

The transport layer (TPDU) connection is covered in sections 7 and A.4.1 of [1]. Section A.4.1.3 of [1] shall be supported with the addition of the following: "TPDU chaining shall not be supported. The maximum length of the transport data shall be limited to 65,534 bytes."

The Host shall request to open a transport connection. The host shall open exactly one transport connection for each POD module.

Table 6.7-A Create Transport Connection

Syntax	Value	# of bits	Mnemonic
Create_T_C() {			
create_T_C_tag	82	8	uimsbf
length_field()	01	8	uimsbf
t_c_id	XX	8	uimsbf
}			

Where XX is defined by the Host. A transport connection ID (t_c_ID) value of zero is invalid.

The POD module shall respond with the following.

Table 6.7-B Create Transport Connection Reply			
Syntax	**Value**	**# of bits**	**Mnemonic**
C_T_C_Reply (){			
C_T_C_Reply_tag	83	8	Uimsbf
length_field()	01	8	Uimsbf
t_c_id	XX	8	Uimsbf
}			

A transport connection of ID "XX" now exists.

6.7.6.4 Resource Manager Session Initialization

The resource manager session initialization is covered in section 7.2.6.1 of [1].

First, the module requests a session to be opened to the Host's Resource Manager.

Table 6.7-C Open Session Request			
Syntax	**Value**	**# of bits**	**Mnemonic**
Open_session_request(){			
open_session_request_tag	91	8	uimsbf
length_field()	04	8	uimsbf
resource_identifier()	00010041	32	uimsbf
}			

The Host shall respond with the following.

Table 6.7-D Open Session Response			
Syntax	**Value**	**# of bits**	**Mnemonic**
Open_session_response(){			
open_session_response_tag	92	8	uimsbf
length_field()	07	8	uimsbf
session_status	00*	8	uimsbf
resource_identifier()	00010041	32	uimsbf
session_nb	YYYY	16	uimsbf
}			

* - assumes that the resource manager is always available.

The session number for the resource manager is YYYY and is created by the Host. A session number (session_nb) of zero is invalid.

The session is now created.

6.7.6.4.1 POD Resource Profile

The POD resource profile is obtained by the Host and is covered in section 8.4.1.1 of [1]. Since the POD module is designed to be the only module in a Host, it shall not report any resources to the Host.

First the Host's Resource Manager sends a Profile Inquiry to the module.

Table 6.7-E Profile Inquiry

Syntax	Value (hex)	# of bits	Mnemonic
profile_inq(){			
profile_inq_tag	9F8010	24	uimsbf
length_field()	00	8	uimsbf
}			

The POD shall respond with

Table 6.7-F Profile Reply

Syntax	Value (hex)	# of bits	Mnemonic
profile_reply(){			
profile_reply_tag	9F8011	24	uimsbf
length_field()	00	8	uimsbf
}			

6.7.6.4.2 Host Resource Profile

The Host shall send a profile_changed APDU so that the POD module shall then perform a profile_inq APDU to which the Host shall respond with its profile_reply APDU.

The Host sends:

Table 6.7-G Profile Changed			
Syntax	**Value (hex)**	**# of bits**	**Mnemonic**
profile_changed(){			
profile_changed_tag	9F8012	24	uimsbf
length_field()	00	8	uimsbf
}			

to which the module replies with:

Table 6.7-H Profile Inquiry			
Syntax	**Value (hex)**	**# of bits**	**Mnemonic**
profile_inq(){			
profile_inq_tag	9F8010	24	uimsbf
length_field()	00	8	uimsbf
}			

to which the Host shall reply with:

Table 6.7-I Profile Reply			
Syntax	**Value**	**# of bits**	**Mnemonic**
profile_reply(){			
profile_reply_tag	9F8011	24	uimsbf
length_field()	N*4	8	uimsbf
for(i=0; i<N; i++) {			
resource_identifier()	XXXXXXXX	32	uimsbf
}			
}			

where N is the number of resource identifiers and XXXXXXXX is each unique resource identifier.

NOTE: If a Host supports multiple versions of a given resource, each version of that resource will be reported as a resource identifier.

Now the module knows what resources are available in the Host.

6.7.6.5 Application Info Session Initialization

Each POD module application shall open a single session to the Application Information resource to pass application information and to manage application menu entry points. Once the session is created, the Host sends an application_info_req APDU to the POD module. The POD module will respond with the application_info_cnf APDU. Detailed operation of the application info is covered in section 8.4 of this document.

6.7.6.6 Conditional Access Application Initialization

A Conditional Access application in the POD module shall then create a single session to the CA Support resource in the Host to allow CA information from the SI and information about user-selected services to be given to the application. Once the session is created, the Host sends a CA Info Inquiry APDU to the application, which responds with CA Info APDU. The Host may then enter into a subsequent dialogue with the CA application to determine which selected services the CA application can descramble and under what conditions. This is described in section 8.4.3 of [1]. Under normal operating conditions, this session will never be closed.

6.7.6.7 Copy Protection

A Copy Protection application in the POD module shall create a session to the Copy Protection resource in the Host. Initialization of Copy Protection is covered in ANSI/SCTE 41 2001.

6.7.6.8 Extended Channel

An extended channel application in the POD module shall create a session to the Extended Channel Support resource to allow for the establishment of flows on the extended channel. These flows will be used for transferring IP packets and MPEG table sections across the POD/Host interface. Under normal operating conditions, this session will never be closed. Please refer to sections 4 and 8.9 of this document.

6.7.6.9 Host Control

A Host Control application shall create a session to the Host Control resource to allow the POD module to control various Host devices. Please refer to section 8.8 of this document for details on the Host Control resource.

6.7.6.10 Low Speed Communication

If reported by the Host as an available resource and the POD module implements a Low Speed Communication application, the POD module application shall create a session to the Low Speed Communication resource to allow the POD module to communicate to the headend through the Host.

6.7.6.11 Generic IPPV Support

If reported by the Host as an available resource and the POD module implements a Generic IPPV application, the POD module application shall create a session to the Generic IPPV resource to allow the Host to receive information on and purchase IPPV events.

6.7.6.12 System Time

If the POD module desires, it shall open a single session to the System Time resource to allow the POD module to receive system time from the Host.

6.7.6.13 Homing

If the POD module desires, it shall open a single session to the Homing resource in the Host to allow the POD module to determine when it may take control of the tuner. The Homing operation is defined in Section 8.13.

6.7.7 Interrupt Operation

Section 6.5.1 of this document defines that the PCMCIA IREQ# signal is available for use by the Host. This signal can be utilized by the Host to simplify the physical layer operation but currently cannot be used for transport layer operation.

6.7.7.1 Physical Level

The following diagram shows the POD module interrupt logical operation.

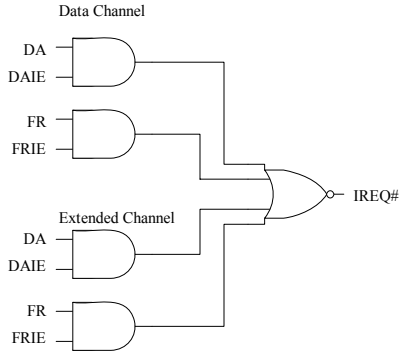

Figure 6.7-3 POD Module Interrupt Logical Operation

From this diagram, an interrupt shall occur whenever the DA or FR bits are set for either the data or extended channels and their corresponding interrupt enable bit is set.

6.7.7.1.1 Data Channel Operation

The Host/POD module relation on the data channel is defined to be a master-slave interface. The Host will periodically poll the POD module to determine if it has data.

The POD module will only transmit data to the Host after one of these polls. The interrupts are particularly useful when the transaction has to be fragmented. The method of interrupt implementation is dependent on the Host manufacturer and is not defined in this document.

6.7.7.1.2 Extended Channel Operation

The Host/POD module relation on the extended channel is defined in section 5.5 of this document. This is a peer type interface. The Host and POD module can transmit data over the extended channel at any time. The interrupt implementation is dependent on the Host manufacturer and is not defined in this document.

6.7.7.1.3 Priorities

Since the data and extended channel interrupts are logically OR'ed together to a single interrupt signal, a priority must be established. Since the data channel is defined to be the command interface, it shall have priority over the extended channel. Additionally, the data channel shall have less traffic overall than the extended channel.

This priority can be easily implemented by having the Host first read the data channel status byte and then the extended channel status byte when an interrupt occurs to resolve the source.

6.8 Mechanical Design

The mechanical design of the POD module shall follow either the PC Card or CardBus specifications called out in [13]. Additionally, any future modifications to the physical specification which are backwards compatible may be implemented.

7 LINK INTERFACE (NORMATIVE)

7.1 Data Channel

The link layer of the Data Channel is compliant to the link layer of CEA-679-C Part B Command Interface.

7.2 Extended Channel

The Extended Channel provides a data path between the POD and the Host. The QPSK modem (the traditional "out-of-band channel") is one such path. A High Speed Host Modem, when present, provides another.

The link layer of the Extended Channel fragments the datagram PDU, if necessary, over the limited buffer size of the physical layer and reassembles 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 8.9) to identify the end-to-end communication flow.

Table 7.2-A Extended Channel Link Layer Packet		
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' then 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' then this fragment is not the first.

- **FLOW_ID**: 3-byte flow identifier associating the data with a registered flow. The FLOW_ID value of zero is reserved is not to be assigned.

For data flows made available to the Host by the POD, the POD is responsible for link layer processing of messages to be transferred across the Extended Channel. It is the Host's responsibility to re-assemble received datagram PDU fragments, and to segment PDUs for delivery across the interface. For data flows made available to the POD by the Host, the roles are reversed.

Received datagram PDU fragments shall be reassembled into either IP packets or MPEG-2 table sections, 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) shall be 4096 for any Service Type.

7.2.1 Maximum PDUs

Datagram PDUs to be transmitted upstream shall be segmented into fragments not exceeding the negotiated buffer size. The maximum size of any PDU before fragmentation shall be 4096 bytes for downstream data for any Service Type. The maximum size of any PDU before fragmentation shall be 1500 bytes for upstream data for any Service Type.

8 APPLICATION INTERFACE (NORMATIVE)

8.1 Scope Introduction

The Host-POD Interface Specification requires the following extensions to the Host, the Host-POD Interface Specification requires the following extensions to CEA-679-C Part B on the Data Channel. The Extended Channel does not have an application layer.

Table 8.1-A Host-POD Interface Resources

Resource	CEA 679-C	Host-POD	Class	Type	Version	Resource ID
Resource Manager	Yes	Yes	1	1	1	00010041
MMI	Yes	Updated	64	2	1	00400081
Application Info	Yes	Updated	2	2	1	00020081
Low Speed Communication (Cable Return)	Yes[2]	Updated and Optional[3]	96	[50]	3	00605043
Low Speed Communication (Host Modem)	Yes[2]	Updated and Optional[3]	96	80	3	00608043
Conditional Access Support	Yes	Yes	3	1	2	00030042
Copy Protection	No	Yes	176	3	1	00B000C1
Host Control	Yes	Updated	32	1	3	00200043
Extended Channel Support	No	Yes	160	1	1[4]	00A00041
Generic IPPV Support	Yes	Updated and Optional[3]	128	2	1	00800081
Specific Application Support	No	Yes	144	1	1	00900041
Generic Feature Control	No	Yes	42	1	1	002A0041
Homing Resource[1]	No	Yes	17	1	2	00110042
System Time	Yes	Yes	36	1	1	00240041
Generic Diagnostic Resource	No	Yes	260	1	1	01040041

NOTES:

1. The Homing resource is defined in Section 8.13 of this standard.

2. The Resource ID delivered by a Host shall be either 0x00605043 for a Host device with a Cable Return Channel (e.g., SCTE 55-1 or SCTE 55-2), or 0x00608043 for a Host device with a Host modem (e.g., DOCSIS). If no Low Speed Communication Resource Identifier is reported by the Host than the Host device is assumed to be equipped with only a Forward Data Channel. The POD may use the presence of the Resource ID as a means to identify what type of Cable Return Channel is supported by the Host.

3. If a device manufacturer opts to implement an optional resource on a device, then the resource shall use the indicated resource ID.

4. The Resource Version depends on the implementation of a Host High Speed (DOCSIS) Modem. See the description of Table 8.9-A in Section 8.9.

Table 8.1-B Host-POD Interface Resource Loading				
Item	**Name**	**Maximum sessions at one time**	**Closes**	**Resource Location**
1	Transport Connection ID	1	No	Host creates TC_ID
2	Sessions total (sum of items 3-16)	128	N/A	N/A
3	Resource Manager	32	No	Host
4	MMI	1	No	Host
5	Application Info	1	No	Host
6	Low Speed Communication	1	Yes	Host
7	Conditional Access Support	1	No	Host
8	Copy Protection	1	No	Host
9	Host Control	1	No	Host
10	Extended Channel Support	2	No	Host
11	Generic IPPV Support	1	Yes	Host
12	Specific Application Support	32	Yes	Host
13	Generic Feature Control	1	No	Host
14	Homing	1	Yes	Host
15	Generic Diagnostic Support	1	No	Host
16	System Time	1	Yes	Host

NOTES:

1. A maximum of one Generic Diagnostic Support resource may be open at a time.

2. A POD module may assume that it is the only POD module in a Host, even if the Host supports multiple POD modules. It is the responsibility of the Host to handle the multiple interfaces. Specifications for multiple POD interfaces are beyond the scope of this standard and subject to further study.

3. After buffer negotiation, the Host will create a transport connection. The POD module will ignore the t_c_id value in the link layer when there is no transport connection established.

4. Only one program may be descrambled at a time, hence only one conditional access support session shall be opened. The conditional access session does not close according to section 6.7. Descrambling of multiple programs is beyond the scope of this standard and subject to further study

5. Only one copy protection session shall be open at a time. The copy protection session does not close according to section 6.7.

6. Two extended channel sessions may be open at a time, however only when the Host has modem capability (either phone, DOCSIS, or other); otherwise only one extended channel session may be open.
The extended channel session does not close according to section 6.7.

7. Only one host control session shall be open at a time. The host control session does not close according to section 6.7.

8. A maximum of one Generic IPPV session may be open at a time.

9. A maximum of one homing session may be open at a time.

10. A maximum of one system time session may be open at a time.

11. The POD shall limit the resource to the values defined in Table 8.1-B. The host shall at a minimum support the number of resources defined in Table 8.1-B.

8.2 Resource Manager

The CEA-679-C Part B ***Resource Manager*** resource shall be implemented.

8.3 Man Machine Interface

8.3.1 Introduction

The Man-Machine Interface resource resides in the Host. The POD 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 Man-Machine Interface resource provides

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

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

The **Man-Machine Interface** resource has been changed to type 2 to reflect the changes listed into this section compared to CEA-679-C.

Table 8.3-A Man Machine Interface Resource				
Resource	**Class**	**Type**	**Version**	**Identifier (hex)**
MMI	64	2	1	000400081

The **Man-Machine Interface** resource includes four APDUs as described in the following table:

Table 8.3-B Man Machine Interface Objects			
Apdu_tag	**Tag value (hex)**	**Resource**	**Direction Host <-> POD**
open_mmi_req()	9F8820	MMI	←
open_mmi_cnf()	9F8821	MMI	→
close_mmi_req()	9F8822	MMI	←
close_mmi_cnf()	9F8823	MMI	→

8.3.2 Open_mmi_req() & Open_mmi_cnf()

The POD shall send an ***Open_mmi_req()*** APDU to the Host when it wants to initialize an MMI dialog. The Host shall reply with an ***Open_mmi_cnf()*** APDU to confirm the status of the request.

For Host that supports more than one MMI dialog at the same time (multiple windows), the POD may send another ***Open_mmi_req()*** APDU, before it closes the previous one.

8.3.2.1 Open_mmi_req()

Table 8.3-C Open MMI Request Object Syntax		
Syntax	**# of bits**	**Mnemonic**
open_mmi_req(){		
open_mmi_req_tag	24	uimsbf
length_field()		
zero	8	0x00
url_length()	16	uimsbf
For (I=0; I < url_length; I++) {		
url_byte	8	uimsbf
}		
}		

Where:

url_byte

Each ***url_byte*** is one octet of a parameter that points to a HTML page in the POD and that needs to be queried by the Host display application using the Server_query() APDU when the MMI dialog will be opened

8.3.2.2 Open_mmi_cnf()

Table 8.3-D Open MMI Confirm Object Syntax		
Syntax	**# of bits**	**Mnemonic**
Open_mmi_cnf(){		
open_mmi_cnf_tag	24	Uimsbf
length_field()		
dialog_number()	8	Uimsbf
open_status()	8	Uimsbf
}		

where:

open_status

The status of the requested MMI dialog as defined in the following table.

Table 8.3-E Open Status Values	
Open_Status	**Value (hex)**
OK- Dialog Opened	00
Request Denied – Host Busy	01
Request Denied – Display Type not Supported	02
Request Denied – No Video Signal	03
Request Denied – No more Windows Available	04
Reserved	05-FF

dialog_number

A number supplied by the Host issued from an 8-bit cyclic counter that identifies each ***Open_mmi_cnf()*** APDU and allows the POD to close the MMI dialog.

8.3.3 Close_mmi_req() & Close_mmi_cnf()

The POD shall send a ***Close_mmi_req()*** APDU to the Host to close the MMI dialog previously opened with an ***Open_mmi_req()*** APDU. The Host shall reply with a ***Close_mmi_cnf()*** object to report the status of the close operation.

The Host may send a ***Close_mmi_cnf()*** without the POD having sent a ***Close_mmi_req()*** to inform about a close operation performed by the Host.

8.3.3.1 Close_mmi_req()

Table 8.3-F Close MMI Request Object Syntax		
Syntax	**# of bits**	**Mnemonic**
close_mmi_req(){		
close_mmi_req_tag	24	Uimsbf
length_field()		
dialog_number()	8	Uimsbf
}		

where:

dialog_number

The number of the MMI dialog provided by the ***Close_mmi_req().***

8.3.3.2 Close_mmi_cnf()

Table 8.3-G Close MMI Confirm Object Syntax		
Syntax	**# of bits**	**Mnemonic**
close_mmi_cnf(){		
close_mmi_cnf_tag	24	uimsbf
length_field()		
dialog_number()	8	uimsbf
}		

where:

dialog_number The number of the MMI dialog provided by the ***Close_mmi_req().***

8.4 Application Information

8.4.1 Introduction

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 (see Section 6.7.6.4).

The **Application Information** resource provides:

- Support to the Host to expose its display characteristics to the POD
- Support to the POD to expose its applications to the Host
- Support to 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 CEA-679-C. 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 8.4-A Application Information Resource				
Resource	**Class**	**Type**	**Version**	**Identifier (hex)**
Application_info	2	2	1	00020081

The **Application Information** resource includes four APDUs as described in the following table:

Table 8.4-B Table Application Information Objects			
Apdu_tag	**Tag value (hex)**	**Resource**	**Direction Host <-> POD**
application_info_req()	9F8020	Application Info	→
application_info_cnf()	9F8021	Application Info	←
server_query()	9F8022	Application Info	→
server_reply()	9F8023	Application Info	←

The method for initiating an application is beyond the scope of this document.

8.4.2 Application_info_req() & Application_info_cnf()

The Host shall send, as soon as the POD has opened the **Application Information** resource, an ***Application_info_req()*** APDU to the POD to advertise its display capabilities. The POD shall reply with an ***Application_info_cnf()*** APDU to describe its supported applications.

8.4.2.1 Application_info_req()

Table 8.4-C Application Information Request Object Syntax		
Syntax	**# of bits**	**Mnemonic**
Application_info_req() {		
Application_info_req_tag	24	uimsbf
length_field()		
Display_rows	16	uimsbf
Display_columns	16	uimsbf
Vertical_scrolling	8	uimsbf
Horizonal_scrolling	8	uimsbf
Multi_window	8	uimsbf
Data_entry_support	8	uimsbf
HTML_support	8	uimsbf
if (HTML_support==1) then {		
Link_support	8	uimsbf
Form_support	8	uimsbf
Table_support	8	uimsbf
List_support	8	uimsbf
Image_support	8	uimsbf
}		
}		

Where:

Display_rows

This field defines the number of rows of the display device.

Display_columns

This field defines the number of columns of the display device. Columns are one pixel in width.

Vertical_scrolling

This field defines if the Host display application supports vertical scrolling (Vertical_scrolling = 1), or not (Vertcal_scrolling = 0).

Default value is 0.

Horizontal_scrolling

This field defines if the Host display application supports horizontal scrolling (Horizontal_scrolling = 1), or not (Horizontal_scrolling =0).

Default value is 0.

Multi_window

This field is deprecated and shouldbe set to zero. **Data_entry_support**

This field defines the data entry capability of the Host, according to the following table. The POD may use this information in creating HTML pages.

Table 8.4-D Data Entry Support Values	
Data_entry_support	**Value (hex)**
None	0
Last/Next	1
Numeric Pad	2
Alpha Keyboard with Mouse	3
Reserved	4-FF

HTML_support

This field defines the HTML support capability of the Host display application, according to the following table. All Hosts shall support as a minimum the Baseline HTML profile. The Baseline HTML profile is defined in Appendix C.

Table 8.4-E HTML Support Values	
HTML_support	**Value (hex)**
Baseline Profile	0
Custom Profile	1
HTML 3.2	2
XHTML 1.0	3
Reserved	4-FF

Link_support

This field defines if the Host display application supports single or multiple Links, according to the following table.

Table 8.4-F Link Support Values	
Link_support	**Value (hex0**
One link	0
Multiple links	1
Reserved	2-FF

Form_support

This field defines if the Host display application supports Forms, according to the following table.

Table 8.4-G Form Support Values	
Form_support	**Value (hex)**
None	0
HTML 3.2 w/o POST method	1
HTML 3.2	2
Reserved	3-FF

Table_support

This field defines if the Host display application supports Tables, according to the following table.

Table 8.4-H Table Support Values	
Table_support	**Value (hex)**
None	0
HTML 3.2	1
Reserved	2-FF

List_support

This field defines if the Host display application supports Lists, according to the following table.

Table 8.4-I List Support Values	
List_support	**Value (hex)**
None	0
HTML 3.2 w/o Descriptive Lists	1
HTML 3.2	2
Reserved	3-FF

Image_support

This field defines if the Host display application supports Images, according to the following table.

Table 8.4-J Image Support Values	
Image_support	**Value (hex)**
None	0
HTML 3.2 – PNG Picture under RGB w/o resizing	1
HTML 3.2	2
Reserved	3-FF

8.4.2.2 Application_info_cnf()

Table 8.4-K Application Information Confirm Object Syntax		
Syntax	**# of bits**	**Mnemonic**
Application_info_cnf() {		
application_info_cnf_tag	24	uimsbf
length_field()		
pod_manufacturer_id	16	uimsbf
pod_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	bslbf
}		
application_url_length	8	uimsbf
for (J=0; J<application_url_length; J++)		
{		
application_url_byte	8	bslbf
}		
}		
}		

Where:

pod_manufacturer_id

The first byte specifies the POD module manufacturer, while the second byte, which is not defined here, can be used by the POD module manufacturer to privately identify product generation and derivatives.

Table 8.4-L Pod Manufacturer ID Values	
Pod_manufacturer_id	**Value (hex)**
Motorola	00XX
Scientific Atlanta	01XX
SCM Microsystems	02XX
Future	0300-FFFF

pod_version_number:

Privately defined by the POD module manufacturer. The POD module shall upgrade the **pod_version_number** each time the POD module is modified by using the POD Module Upgrade Host Interface (Section 8.13).

application_type

The POD module application type are:

Table 8.4-M Application Type Values		
Application_type	**Value (hex)**	**Description**
Conditional Access	00	conditional access application
POD-Host Binding Information	01	POD-Host binding information application (NOTE: see Figure 3.2-A of SCTE 41 2003)
IP Service	02	support for bi-directional IP transactions over the extended channel
Network Interface –SCTE 55-2	03	support for ANSI/SCTE 55-2 2002 PHY and MAC layers on the out-of-band channel
Network Interface – SCTE 55-1	04	support for ANSI/SCTE 55-1 2002 PHY and MAC layers on the out-of-band channel
Copy Protection	05	Copy protection application
Diagnostic	06	Diagnostic application
Undesignated	07	Undesignated application
Reserved	08-FF	

application_version_number

Defined by the POD application supplier. The POD module shall upgrade the application_version_number each time the POD application software is modified according to the POD Module Firmware Upgrade Host Interface (Section 8.13).

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 POD module's application with the application name. The application name, when selected by the user, triggers a Host initialized MMI dialog.

The Host shall display at least eight different POD module application name strings in its top menu. The application name length shall be limited to 32 characters.

application_url_byte

Defines the URL of the POD module application's top level HTML page in the POD module 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() object to initialize an MMI dialog with the POD module application, when an object identified by either the application name or the application URL is selected in the Host menu.

8.4.3 Server_Query() & Server_Reply()

The Host shall send a ***Server_query()*** APDU to the POD to request the information in the POD file server system pointed by a specific URL. The URL defines the access, Host, and location of the data that the Host is requesting. Upon receipt of the URL, the POD module 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.

8.4.3.1 Server Query

Table 8.4-N Server Query Object Syntax		
Syntax	**# 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
}		
}		

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 right MMI dialog.

Header_byte

Each **header_byte** is an octet of an optional parameter that uses the same format as the HTTP/1.1[16] 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 0. The POD module may also ignore this parameter.

url_byte

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 POD module, the URL must allow the transfer of a file of data from the POD module to the Host.

The access indicator is “pod”.

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 POD module, the properly constructed URL would be:

pod:///apps/user/program_guide/menu.html

If, after receiving a **server_reply** from the POD module, the Host has data that it wants to send to the POD module, 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 "&"s. This list is preceded by a "?". A properly constructed URL would be:

pod:///path/file?name1=value1&name2=value2&…

Such a URL sent to an application on the POD module as a response to a **server_reply** would cause the name-value pairs to be processed by the application. Data entered and selected at the Host may be sent to the POD module through the use of these name-value pairs as part of the URL in a **server_query** object.

8.4.3.2 Server Reply

Table 8.4-O Server Reply Object Syntax

Syntax	# of bits	Mnemonic
server_reply(){		
server_reply_tag	24	Uimsbf
length_field()		
transaction_number	8	Uimsbf
file_status	8	Uimsbf
header_length	16	Uimsbf
For (I=0; I < header_length; I++) {		
Header_byte	8	Uimsbf
}		
file_length	16	Uimsbf
For (I=0; I < file_length; I++) {		
file_byte	8	Uimsbf
}		
}		

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 right MMI dialog.

file_status

The file status is indicated by the following values:

Table 8.4-P File Status Values	
File_status	**Value (hex)**
OK	00
URL_not_found	01
URL_access_not_granted	02
Reserved	03-FF

Header_byte

This parameter is an optional parameter to pass HTTP/ 1.1[16] response headers. This header may contain additional information on served file, like mime type, expiration date, etc. Both Host and module may not support this extension and process only file bytes. A POD module not supporting headers shall set header_length to 0.

file_byte

Each file_byte is an octet of the contents of the requested file. A server reply object with file_length equals to 0 will be interpreted as a null file.

8.5 Low Speed Communication ([1])

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 POD via the Host-POD interface. All downstream OOB data shall be passed directly to/from the POD via the Host-POD OOB Interface. Support for Section 8.7 of CEA-679B Part B is optional.

Table 8.5-A Low Speed Communication Resource				
Resource	**Class**	**Type**	**Version**	**Identifier (hex)**
Low_Speed_Communication (Cable Return)	96	321	3	00605043
Low_Speed_Communication (Host Modem)	96	513	3	00608043

For Host devices with a RDC, the Low_Speed_Communication Identifier shall be 0x00605043 for a Host device with a Cable Return Channel only , or 0x00608043 for a Host with a Cable Return Channel and a Host Modem (e.g., DOCSIS).

A Host that supports FDC only (no RDC) shall not report a Low Speed Communication identifier of either 0x00605043 or 0x00608043.

A Host that has a DOCSIS Cable Modem but does not have a SCTE 55 transmitter is not supported and shall not report a Low Speed Communication identifier of either 0x00605043 or 0x00608043.

The following table summarizes this operation:

Table 8.5-B Low Speed Communication Resource ID Reporting Matrix			
SCTE 55 Receiver (FDC)	**SCTE 55 Transmitter (RDC)**	**DOCSIS Cable Modem**	**Low Speed Communication Resource ID**
Yes	No	No	None
Yes	No[1]	Yes	None
Yes	Yes	No	0x00605043
Yes	Yes	Yes	0x00608043

[1] This variation is not permitted.

[1]

8.6 Conditional Access

The CEA-679-C Part B ***Conditional Access Support*** resource shall be implemented in its entirety, with the following addition:

The POD shall inform the host of changes in CA states by sending the CA_UPDATE APDU defined below. A new version of the CA resource on the host shall process the CA_UPDATE APDU.

Table 8.6-A Conditional Access Support Resource				
Resource	**Class**	**Type**	**Version**	**Identifier (hex)**
Conditional Access Support	3	1	2	00030042

This extension includes the following objects:

Table 8.6-B Conditional Access Support Objects			
Apdu_tag	**Tag value (hex)**	**Resource**	**Direction Host <-> POD**
CA_update()	9F8034	Conditional Access Support	←

8.6.1 CA_update()

The POD Module shall use the ***CA_update()*** object to inform the Host when CA information for the currently tuned program has changed. Here CA_PMT(query) refers to a CA_PMT APDU with its ca_pmt_cmd_id parameter set to "query". Note that CA_UPDATE shall always reference the service to which the host is currently tuned. This is the last service for which a CA_PMT() was sent from the Host to the POD.

The different APDU tag prevents any confusion between CA_PMT_reply and CA_update APDUs during CA_PMT (query)/CA_PMT_reply exchanges. CEA-679-C Part B states that a CA_PMT (query) is sent by the Host to determine which conditional access resource can decrypt the specified service *when more than one*

conditional access resource is present. The reader should note that some conditional access implementations may send a CA_PMT (query) each time a service is tuned and with only one POD installed in the Host. While this behavior may differ from that defined in CEA-679-C Part B, it is done to determine if the currently tuned service can be descrambled by the POD. The POD may respond with a CA_PMT_reply specifying "descrambling possible" or "descrambling not possible". The Host would respond to a CA_PMT_reply (descrambling_possible) with a CA_PMT (ok_descrambling) to the POD.

Table 8.6-C Conditional Access Support CA_update Object Syntax

Syntax	# of bits	Mnemonic
CA_update() {		
ca_update_tag	24	Uimsbf
Length_field()		
Program_number	16	Uimsbf
Reserved	2	Bslbf
Version_number	5	Uimsbf
Current_next_indicator	1	Bslbf
CA_enable_flag	1	Bslbf
if (CA_enable_flag==1)		
CA_enable /* at program level */	7	Uimsbf
else if (CA_enable_flag==0)		
reserved	7	Bslbf
for (i=0; I<N; i++) {		
reserved	3	Bslbf
elementary_PID	13	Uimsbf
CA_enable_flag	1	Bslbf
if (CA_enable_flag==1)		
CA_enable /*at elementary stream level*/	7	Uimsbf
else if (CA_enable_flag==0)		
reserved	7	Bslbf
}		
}		

The syntax contains one possible ca_enable at program level and, for each elementary stream, one possible ca_enable at elementary stream level.

. When both are present, only ca_enable at ES level applies for that elementary stream

When none is present, the host does not interpret the ca_pmt_reply object.

The CA_enable field indicates whether the application is able to perform the descrambling operation. CA_enable values are coded as follows:

Table 8.6-D CA Enable Field Values	
CA_enable	**Value (hex)**
Descrambling possible	01
Descrambling possible under conditions (purchase dialogue)	02
Descrambling possible under conditions (technical dialogue)	03
Descrambling not possible (because no entitlement)	71
Descrambling not possible (for technical reasons)	73
RFU other values	other values 00-7F

The value "***descrambling possible***" means that the application can descramble with no extra condition (e.g.: the user has a subscription) or that the user has authorized the purchase of the elementary stream.

The value "***descrambling possible under conditions (purchase dialogue)***" means that the application has to enter into a purchase dialogue with the user before being able to descramble (pay per view program).

The value "***descrambling possible under conditions (technical dialogue)***" means that the application has to enter into a technical dialogue with the user before being able to descramble (e.g. : ask the user to select fewer elementary streams because the descrambling capabilities are limited).

The value "***descrambling not possible (because no entitlement***)" means that the user has no entitlement for the program or the user does not want to buy the program.

The value "***descrambling not possible (for technical reasons)***" means that the application cannot descramble the elementary stream for technical reasons (e.g. : all descrambling capabilities are already in use).

The protocol allows services to be selected for descrambling at either the program level or the elementary stream level. Where the host cannot support descrambling of different elementary streams by different modules, then it shall take as the CA_enable

value for the program the lowest of the CA_enable values returned for each elementary stream of the program. This ensures that any stream will be descrambled if it can be.

8.7 Copy Protection

A ***Copy Protection*** resource shall be required. The Copy Protection resource class for the Host-POD Interface is defined in Table 8.1-A of this standard.

8.8 Host Control

The CEA-679-C Part B ***Host Control*** resource is modified to give the POD Module the capability to set up the Host RF Receiver and the RF Transmitter, if any. Additional modifications have been included to give the POD module the capability to tune to the Host's In-band Receiver. This version of the Host Control resource includes both the OOB and inband tuning. While the OOB tuning is not used by the Host, the inband tuning is used. It is anticipated that the Host will perform its internal tuning operation without requiring to open a session to the Host Control resource. Therefore, when it receives any inband tuning APDUs from the POD module, it must decide whether to grant them or not. Typically, unless either a Homing resource has been opened and either the Host is in standby state or the POD module has transmitted a firmware_upgrade, the Host should not grant access to the inband tuner to the POD module. Support of version 1 in CEA-679-C Part B is optional.

Table 8.8-A Host Control Resource

Resource	Class	Type	Version	Identifier (hex)
Host Control	32	1	3	00200043

The POD module will have the Host Control resource open for control of the OOB receiver and transmitter and shall leave it open independent of the operation of the Homing resource.

The creation of the specification application resource includes the following objects:

Table 8.8-B Host Control Objects			
Apdu_tag	**Tag value (hex)**	**Resource**	**Direction Host <-> POD**
OOB_TX_tune_req()	9F8404	Host Control	←
OOB_TX_tune_cnf()	9F8405	Host Control	→
OOB_RX_tune_req()	9F8406	Host Control	←
OOB_RX_tune_cnf()	9F8407	Host Control	→
inband_tune_req()	9F8408	Host Control	←
inband_tune_cnf()	9F8409	Host Control	→

8.8.1 OOB_TX_tune_req() & OOB_TX_tune_cnf()

The POD Module shall use the ***OOB_TX_tune_req()*** object to set up the Host's RF Transmitter.

The Host shall respond with the ***OOB_TX_tune_cnf()*** object to the ***OOB_TX_tune_req()*** request.

Table 8.8-C OOB TX Tune Request Object Syntax		
Syntax	**# of bits**	**Mnemonic**
OOB_TX_tune_req() {		
OOB_TX_tune_req_tag	24	Uimsbf
Length_field()		
RF_TX_frequency_value	16	Uimsbf
RF_TX_power_level	8	Uimsbf
RF_TX_rate_value	8	Uimsbf
}		

.

Table 8.8-D RF TX Frequency Value								
Bit	7	6	5	4	3	2	1	0
	Frequency (MS)							
	Frequency (LS)							

RF_TX_frequency_value – This field defines the frequency of the RF Transmitter, in kHz.

Table 8.8-E RF TX Power Level								
Bit	7	6	5	4	3	2	1	0
	RF Power Level							

RF_TX_power_level - Power level of the RF Transmitter, in units of 0.5dBmV. The value 0x00 shall correspond to an output level of 0 dBmV.

Table 8.8-F RF TX Rate Value								
Bit	7	6	5	4	3	2	1	0
	Rate		Reserved					

RF_TX_rate_value

- ***Rate –*** Bit rate. 00 = 256 kbps
  01 = 2048 kbps
  10 = 1544 kbps
  11 = 3088 kbps

Table 8.8-G OOB TX Tune Confirm Object Syntax		
Syntax	**# of bits**	**Mnemonic**
OOB_TX_tune_cnf() {		
OOB_TX_tune_cnf_tag	24	Uimsbf
Length_field()		
Status_field	8	Uimsbf
}		

- ***Status_field*** – This field returns that status of the ***OOB_TX_tune_req().*** If the request was granted and the RF Transmitter set up 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 POD shall not attempt to perform RF transmit operations

after receiving an ***OOB_TX_tune_req()*** with ***Status_field*** set to 0x01. If any of the parameters passed to the Host are outside of its capability, then the Host shall transmit the ***OOB_TX_tune_req()*** with ***Status_field*** set to 0x03. Otherwise it will be set to one of the following values:

Table 8.8-H Status Field Values for OOB TX Tune Confirm

Status_field	Value (hex)
Tuning granted	00
Tuning Denied – RF Transmitter not physically available	01
Tuning Denied – RF Transmitter busy	02
Tuning Denied – Invalid Parameters	03
Tuning Denied – Other reasons	04
Reserved	05-FF

8.8.2 OOB_RX_tune_req() & OOB_RX_tune_cnf()

The POD Module shall use the ***OOB_RX_tune_req()*** object to set up the Host's RF Receiver.

The Host shall respond with the ***OOB_RX_tune_cnf()*** object to the ***OOB_RX_tune_req()*** request.

The OOB_RX_tune_cnf APDU should only be transmitted after either the requested frequency has been tuned and acquired ("tune time"), or 500msec has elapsed since receiving the Request, whichever comes first.

Table 8.8-I OOB RX Tune Request Object Syntax

Syntax	# 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
}		

- ***RF_RX_frequency_value*** – This field defines the frequency of the RF Receiver, in MHz. (Frequency = value * 0.05 + 50 MHz.)

Table 8.8-J RF RX Frequency Value								
Bit	7	6	5	4	3	2	1	0
	0	0	0	0	0	Value (MS)		
	Value (LS)							

RF_RX_coding_value – This field defines the RF Receiver characteristics.

Table 8.8-K RF RX Data Rate							
Bit	7-6	5	4	3	2	1	0
	Rate	0	0	0	0	0	Spec

- ***Rate –*** Bit rate = 2048 kbps when Rate = 00 or 01
  1544 kbps when Rate = 10
  3088 kbps when Rate = 11

- ***Spec –*** Spectrum is non-inverted when Spec=0 and inverted when Spec=1

Table 8.8-L OOB RX Tune Confirm Object Syntax		
Syntax	**# of bits**	**Mnemonic**
OOB_RX_tune_cnf() {		
OOB_TX_tune_cnf_tag	24	Uimsbf
Length_field()		
Status_field	8	Uimsbf
}		

- ***Status_field*** – This field returns that status of the ***OOB_RX_tune_req().*** If the request was granted and the RF receiver set up to the desired configuration, ***Status_field*** will be set to 0x00. Otherwise it will be set to one of the following values:

Table 8.8-M Status Field Values for OOB RX Tune Confirm	
Status_field	**Value (hex)**
Tuning granted	00
Tuning Denied – RF Receiver not physically available	01
Tuning Denied – RF Receiver busy	02
Tuning Denied – Invalid Parameters	03
Tuning Denied – Other reasons	04
Reserved	05-FF

8.8.3 inband_tune_req() (Normative)

The inband_tune_req() APDU allows for the POD module to request the Host to tune the inband QAM tuner. The APDU will allow support of tuning to either a source_id or a frequency with the modulation type.

Table 8.8-N Inband Tune Request Object Syntax		
Syntax	**# of bits**	**Mnemonic**
Inband_tune_req() {		
Inband_tune_tag	24	uimsbf
length_field()		
tune_type	8	uimsbf
if(tune_type == 0){		
source_id	16	uimsbf
}		
else if (tune_type == 1) {		
tune_frequency_value	16	uimsbf
modulation_value	8	uimsbf
}		
}		

tune_type – Determines the definition of the tune_frequency_value.

Table 8.8-O Tune Type Values		
Value (hex)	**Type**	
00	Source ID	
01	Frequency	
02-FF	Reserved	

source_id – When tune_type = 0, the source_id is a 16 bit unsigned integer in the range of 0x0000 to 0xFFFF that identifies the programming source associated with the virtual channel on a system wide basis. IN this context, a source is one specific source of video, text, data, or audio programming. For the purposes of referencing virtual channels to the program guide database, each such program source is associated with a unique value of source_id. The source_id itself may appear in an IPG database, where it tags entries to associate them with specific services. The value zero for source_id, if used, shall indicate the channel is not associated with a source_id.

tune_frequency_value – When tune type = 1, tune_frequency_value contains the frequency for the Host to tune. The frequency is calculated by multiplying tune_frequency_value by 0.05 MHz (50 KHz resolution).

Table 8.8-P Tune Value								
Bit	7	6	5	4	3	2	1	0
MSB	Value (MS)							
LSB	Value (LS)							

modulation_value – When tune type = 1, modulation_value sets the type of modulation for the inband tuner.

Table 8.8-Q Modulation Value	
Value (hex)	**Type**
00	QAM-64
01	QAM-256
02-FF	Reserved

8.8.4 inband_tuning_cnf (Normative)

After the Host has received the inband_tuning, it will respond with the following APDU.

Table 8.8-R Inband Tuning Confirm Object Syntax		
Syntax	**# of bits**	**Mnemonic**
Inband_tuning_cnf() {		
inband_tuning_cnf_tag	24	uimsbf
length_field()		
tune_status	8	uimsbf
}		

tune_status – The Host response to the inband_tuning APDU.

Table 8.8-S Tune Status Values		
Value (hex)	**Source**	**Comment**
00	Tuning accepted	Frequency, modulation type 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 the tuner.
05-FF	Reserved	

8.9 Extended Channel Support

For purposes of the ***Extended Channel***, the device (POD Module or Host) that provides the physical communications link to the headend is referred to as the "link device." The POD Module is the link device for the QPSK modem; the Host is the link device for the High Speed Host Modem.

The ***Extended Channel Support*** resource shall be created to register the interactive 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 channel for the POD. The POD 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 POD will terminate data received on the QPSK downstream OOB channel by using it itself (for example EMMs). In other cases, it may perform a filtering function and discard data known to be of no interest to the Host.

Supported system architectures imply two 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 POD Module and the data are transferred to/from the headend via the Host's High Speed Host Modem.

This resource has two versions. Version 1 of this resource is required for Hosts that do not have an embedded High Speed Host (DOCSIS) Modem and version 2 of this resource is required for Hosts that do have an embedded High Speed Host (DOCSIS) Modem. Version 2 of this resource includes support for all of the objects defined by version 1.

Table 8.9-A Extended Channel Resource				
Resource	**Class**	**Type**	**Version**	**Identifier (hex)**
Extended_Channel	160	1	1	00A00041
Extended_Channel	160	1	2	00A00042

This creation includes the following objects:

Table 8.9-B Extended Channel Objects				
Apdu_tag	**Tag value (hex)**	**Resource (Version)**	**Direction Host ←→ POD**	
			Host Modem	**POD Modem**
new_flow_req()	9F8E00	Extended Channel Support (1)	←	→
new_flow_cnf()	9F8E01	Extended Channel Support (1)	→	←
delete_flow_req()	9F8E02	Extended Channel Support (1)	←	→
delete_flow_cnf()	9F8E03	Extended Channel Support (1)	→	←
Lost_flow_ind()	9F8E04	Extended Channel Support (1)	→	←
Lost_flow_cnf()	9F8E05	Extended Channel Support (1)	←	→
inquire_DSG_mode()	9F8E06	Extended Channel Support (2)	→	→
Set_DSG_mode()	9F8E07	Extended Channel Support (2)	←	←
DSG_packet_error()	9F8E08	Extended Channel Support (2)	←	←

8.9.1 New_flow_req() & New_flow_cnf()

The application shall use the **new_flow_req()** object to register a new flow with the link device.

The link device shall return a n**ew_flow_cnf()** object in response to the n**ew_flow_req()** request.

Table 8.9-C New Flow Request Object Syntax		
Syntax	**# of bits**	**Mnemonic**
new_flow_req() {		
new_flow_req_tag	24	uimsbf
length_field()		
service_type	8	uimsbf
if (service_type == mpeg_section) {		
Reserved	3	bslbf
Pid	13	uimsbf
}		
if (service_type == ip_u) {		
mac_address	48	uimsbf
option_field_length	8	uimsbf
for (i=0; i<option_field_length; i++) {		
option_byte	8	uimsbf
}		
}		
if (service_type == ip_m) {		
Reserved	4	bslbf
multicast_group_id	28	uimsbf
}		
}		

Service_type – This field defines the type of the requested service.

Table 8.9-D Service Type Values for New Flow Request	
Service_type	**Value**
MPEG_section	0x00
IP_U – IP Unicast	0x01
IP_M – IP Multicast	0x02
DSG	0x03
Reserved	0x04-FF

MPEG_section – This Service_type is applicable only for flows between between POD module and the Host. The requested flow shall be in the form of MPEG-2 table sections (both long and short form). This type of flow is uni-directional, from the POD module to the Host only. It should be noted that the data may originate from either the QPSK modem or via the DSG interface. The value of the section_length field in these sections shall not exceed 4093 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 present. The 32-bit CRC is also present in short-form sections (as ijndicated 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 preset, the POD module 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 POD module 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 POD module may send these sections to the Host without any checks. In this case, the Host is responsible for validation of these sections.

IP_U – IP Unicast. This service type shall be used both for flows between the POD Module and an embedded cable modem in the Host (DSG mode), and between the Host and an SCTE 55 modem in the POD module (SCTE 55 mode). The requested flow shall be in the form of IP packets addressed to or from the POD Module's IP address when in DSG mode, and the Host's IP address when in SCTE 55 mode. This type of flow may be bi-directional. The maximum length of any IP packet in SCTE 55 mode shall be 1500 bytes. With respect to DSG mode, the requested flow from the POD Module to the Host shall be in the form of IP packets addressed to the applicable destination IP address as determined by the POD Module. The requested flow from the Host to the POD Module in DSG mode shall be in the form of IP packets addressed to the IP address of the POD Module. The maximum total length of any IP packet in DSG

mode shall be the DOCSIS maximum transmission unit (MTU) which is 1500 bytes. This MTU is relayed to the POD Module via the new_flow_conf() APDU.

IP_M – IP Multicast. This Service_type is shall be used both for flows between the POD module and a modem in the Host and for the Host and a modem in the POD module. The requested flow shall be in the form of multicast IP packets addressed to the multicast_group_ID assigned IP address. This type of flow is uni-directional, from network to application only. The maximum total length of any IP packet shall be 1500 bytes.

DSG – DSG extended channel interface. This data may only be transmitted from the Host to the POD module and only one flow may be open at a time. The format is defined in section 5.4 If the Host does not support DSG, then it shall return the "Request Denied – Service Type Unavailable" error code (0x02) in the new_flow_cnf() response.

PID – The 13-bit MPEG-2 Packet Identifier associated with the flow request. The POD shall be responsible for filtering the OOB MPEG-2 Transport Stream and delivering only MPEG table sections delivered on transport packets with the given value of PID.

multicast_group_ID – The 28-bit 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 IP_multicast_group_ID address.

MAC_address – The 48-bit MAC address of the entity requesting the unicast IP flow.

option_field_length – An 8-bit unsigned integer number that represents the number of bytes of option field data to follow.

option_byte – These bytes correspond to the options field of a DHCP message. One or more DHCP options per RFC 2132 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.

Conformance to this specification requires the Host and the POD Module to comply with the following requirements:

- The POD Module Interface shall support at least six concurrent MPEG_section Service_type flows.
- The POD Module Interface shall support at least one IP_U Service_type flow.
- If the Service Information Virtual Channel Table indicates that one or more services are defined as being transport out-of-band, the POD module shall provide one or more additional flows of the MPEG_section type.

- If the Host supports DSG, it shall support one DSG Service_type flow
- When the Host support a unicast IP flow, it shall use DHCP per RFC 2131 to obtain an IP address for POD module use. The Host shall provide the options parameters supplied by the POD module in the **New_flow_req()** to build the DHCP message, and add any other options as necessary or desired.
- The POD Module and Host are required to support only one outstanding **New_flow_req()** transaction at a time. The POD Module or Host shall send a **New_flow_cnf()** with a Status_field of 0x04 (Network Busy) when addtional **New_flow_req()** messages are received and one is pending.

Table 8.9-E New Flow Confirm Object Syntax

Syntax	# 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 == 0) {		
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=1; i<option_field_length; i++){		
option_byte	8	uimsbf
}		
}		
}		
}		

Status_field – This field returns the status of the New_flow_req(). If the request was granted and a new flow created, the Status_field will be set to 0x00. Otherwise it will be set to one of the following values:

Table 8.9-F Status Field Values for New Flow Confirm

Status_field	Value
Request Granted	0x00
Request Denied – Number of flows exceeded	0x01
Request Denied – Service_type not available	0x02
Request Denied – Network unavailable or not responding	0x03
Request Denied – Network Busy	0x04
Reserved	0x05-FF

flows_remaining – Indicates the number of additional flows of the same Service_type that can be supported. The value 0 indicates that no additional flows beyond the one currently requested can be supported.

FLOW_ID – The unique flow identifier for this application's data flow. The FLOW_ID value of zero is reserved and is not to be assigned.

Service_type – This field reflects the type of the requested service.

IP_Address – This field is the 32-bit IP address associated with the requested flow.

flow_type - an 8-bit unsigned integer number that represent(s) the protocols supported by the POD to establish the IP-U flow. The field has the following values:

0x00 UDP and TCP supported

0x01 UDP unly supported

0x02-0xFF reserved

flags - a 3-bit field that contains information as defined in the following table pertaining to limitations associated with the interactive network.

Table 8.9-G Flag field definitions		
BITS		
2	1	0
Reserved		no_frag

no_frag - a 1 bit Boolean that designates whether the network supports fragmentation. A value of 0_2 indicates that fragmentation is supported. A value of 1_2 indicates that fragmentation is not supported.

max_pdu_size - a 13 bit unsigned integer that designates the maximum PDU length that may be transmitted across the interface.

option_field_length - an 8 bit unsigned integer number that represents the number of bytes of option field data to follow.

option_byte - these bytes correspond to the options requested in the new_flow_req() message. The format of the field is as defined in RFC 2132. The end option (code 255) shall not be used.

8.9.1.1 new_flow_req IP Unicast DSG Mode Details

When the Host is configured for DSG mode, then the Host and POD Module shall interact as defined within this section:

- If the POD Module requires two-way communications in DSG mode, then the POD Module shall request a new IP Unicast flow using the new_flow_request() APDU
- When requesting the new flow, the POD Module shall at a minimum supply the Vendor Specific Options defined in Table 8.9-x and provide the MAC address of the POD Module in the MAC_address field.
- If the Host grants the request for the IP flow, then the Host shall acquire an IP address for the POD MODULE and forward this information to the POD Module in the IP_address field of the new_flow_cnf() APDU, the maximum transmission unit (MTU) for the DOCSIS network (1500 bytes for IP datagrams) in the max_pdu_size field and any requested options via the option_byte field.
- The Host shall acquire a unique IP address for the POD Module utilizing DHCP as defined in RFC-2131.
- The Host shall be responsible for resolving POD Module IP address conflicts as defined in RFC-2131.

- All Host DHCP transactions associated with acquiring the POD Module and IP address shall be over the embedded cable modem interface and shall not propagate to any other interface on the Host.
- The Host shall not use Network Address Translation (NAT) to provide an IP address for the POD Module.
- On receipt of an IP packet from the POD Module over the interface via the IP Unicast Flow, the Host shall acquire the appropriate destination MAC address, encapsulate the IP packet received over the interface within Ethernet frames utilizing the acquired destination MAC address.
- On receipt of an Ethernet frame from the embedded cable modem interface targeted to the MAC address of the POD Module, the Host shall extract the embedded IP packet and forward the packet to the POD Module via the granted Unicast IP Flow
- The Host shall not forward any Ethernet frames or IP packets destined to the POD Module to any interface other than the CHI.
- The Host shall only forward IP packets destined to the POD Module that have been received via the embedded cable modem interface or via applications resident on the Host.
- The Host shall not forward any IP packets received from the POD Module over the CHI to any interface other than the eCM interface.
- The POD Module shall implement the DHCP Vendor Specific Information Option (option 43) and Vendor Class Identifier Option (option 60) as specified in Tables 8.9-x and 8.9-y.

Table 8.9-H POD Module DHCP Vendor Specific Information (Option 43) Sub-option Encoding		
Sub-option	**Value**	**Description**
1	“<null>”	The request sub-option vector is a list of sub-options (within option 43) to be returned to client by the server upon reply to the request. None defined.
2	“CARD”	Device type of the entity making the DHCP request.
3	“ECM:ESTB:CARD”	Indicates that a POD Module is making a request via the eCM’s DOCSIS return channel
4	“<device serial number>”	Serial Number of POD Module. If Serial Number is not available, then other unique identifier (other than MAC Address) may be utilized
5	“<hardware version number>“	Hardware version number of POD Module
6	“<firmware version number>“	Firmware version number of POD Module
7	“<boot ROM version number>“	Boot ROM version number of POD Module
8	e.g. “0204DF”	A 6-octet, hexadecimal-encoded, vendor-specific Organization Unique Identifier (OUI) that may match the OUI in the embedded cable modem’s MAC address.
9	e.g. “XYZ-CARD-001”	Vendor model number of POD Module
51	e.g. “XYZ Corporation”	Vendor name
52	“yyyyyy”	POD Module capability using the encoding format per DOCSIS specification. Since there is no standard/required capability identification, Conditional Access vendor must provide documentation on the supported capability.
53	e.g. “000-01234-56789-000” (example is unit address of Motorola POD Module)	Conditional Access Vendor specific device identification
54	e.g., “00AA11BB22CC33DD”	64 bit Card_ID as specified in POD Module X.509 certificate

Table 8.9-I POD Module DHCP Vendor Class Indentifier (Option 60) Encoding		
Option	**Value**	**Description**
60	"OpenCable 2.0"	OpenCable Version

8.9.2 Delete_flow_req() & Delete_flow_cnf()

The interactive application shall use the ***Delete_flow_req()*** object to delete a registered data flow.

The link device shall respond with the ***Delete_flow_cnf()*** object to the ***Delete_flow_req()*** request.

Table 8.9-J Delete Flow Request Object Syntax		
Syntax	**# of bits**	**Mnemonic**
Delete_flow_req() {		
Delete_flow_req_tag	24	Uimsbf
Length_field()		
FLOW_ID	24	Uimsbf
}		

Table 8.9-K Delete Flow Confirm Object Syntax		
Syntax	**# of bits**	**Mnemonic**
Delete_flow_cnf() {		
Delete_flow_cnf_tag	24	Uimsbf
Length_field()		
FLOW_ID	24	Uimsbf
Status_field	8	Uimsbf
}		

- ***Status_field*** – This field returns the status of the ***Delete_flow_req***(). If the request was granted and the flow deleted, the Status_field will be set to 0x00. Otherwise it will be set to one of the following values:

Table 8.9-L Status Field for Delete Flow	
Status_field	**Value (hex)**
Request Granted	0x00
Reserved	0x01-0x02
Request Denied – Network unavailable or not responding	0x03
Request Denied – Network busy	0x04
Request Denied – FLOW_ID does not exist	0x05
Request Denied – Not authorized	0x06
Reserved	0x07-0xFF

8.9.3 Lost_flow_ind() & Lost_flow_cnf()

A link device shall indicate that a registered data flow has been lost by issuing the ***Lost_flow_ind()*** object.

The application shall respond with the ***Lost_flow_cnf()*** object in response to the ***Lost_flow_ind()*** object.

Table 8.9-M Lost Flow Indication Object Syntax		
Syntax	**# of bits**	**Mnemonic**
Lost_flow_ind() {		
Lost_flow_ind_tag	24	Uimsbf
Length_field()		
FLOW_ID	24	Uimsbf
Reason_field	8	Uimsbf
}		

- ***Reason_field*** – This field returns the reason the flow was lost. It will be set to one of the following values:

Table 8.9-N Reason Field Values for Lost Flow Indication	
Reason_field	**Value (hex)**
Unknown or unspecified reason	0x00
IP address expiration	0x01
Network down or busy	0x02
Lost or revoked authorization	0x03
Reserved	0x04-0xFF

Table 8.9-O Lost Flow Confirm Object Syntax		
Syntax	**# of bits**	**Mnemonic**
Lost_flow_cnf() {		
Lost_flow_cnf_tag	24	Uimsbf
Length_field()		
FLOW_ID	24	Uimsbf
Status_field	8	Uimsbf
}		

Status_field – This field returns the status of the ***Lost_flow_ind***(). If the indication was acknowledged, the Status_field will be set to 0x00. Otherwise it will be set to one of the following values:

Table 8.9-P Status Field Values for Lost Flow Confirm	
Status_field	**Value (hex)**
Indication Acknowledged	0x00
Reserved	0x01-0xFF

8.9.4 inquire_DSG_mode(), set_DSG_mode(), & DSG_packet_error()

Version 2 of the Extended Channel Support Resource adds the following three messages:

- **inquire_DSG_mode ()** - The Host can inquire of the POD the preferred operational mode for the network, either OOB mode or DSG mode.
- **set_DSG_mode ()** - The POD can inform the Host of the preferred operational mode for the network, either OOB mode or DSG mode.

- **DSG_packet_error ()** - The POD can inform the Host of errors that occur in receiving DSG packets.

The Host shall use the i**nquire_DSG_mode ()** object to inquire the preferred operational mode for the network.

Table 8.9-Q Inquire DSG Mode Object Syntax		
Syntax	**# of bits**	**Mnemonic**

inquire_dsg_mode () {
inquire_dsg_mode_tag
length_field()
}
	24	uimsbf

The POD shall use the s**et_DSG_mode ()** object to inform the Host of the preferred operational mode for the network. This message is sent in response to the i**nquire_DSG_mode ()** message or it may be sent as an unsolicited message to the Host. The method by which the POD determines the preferred operational mode is proprietary to the CA/POD system vendor.

Table 8.9-R Set DSG Mode Object Syntax		
Syntax	**# of bits**	**Mnemonic**
set_dsg_mode () {		
set_dsg_mode_tag	24	uimsbf
length_field()		
operational_mode	8	uimsbf
if (operation_mode==dsg_mode or operation_mode==dsg_one-way_mode) {		
number_mac_addresses	8	uimsbf
for (i=0; i< number_mac_addresses; i++) {		
dsg_mac_address	48	uimsbf
}		
remove_header_bytes	16	uimsbf
}		
}		

Operational_Mode – This field defines the preferred operational mode of the network.

Operational_mode	Value
OOB_mode	0x00
DSG_mode	0x01
DSG_One-Way_mode	0x02
Reserved	0x03-FF

OOB_mode - In this mode, the reverse transmitter is under the control of the POD module through the use the **OOB_TX_tune_req ()** message. 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 POD module uses the OOB-RDC for returning data to the cable headend.

DSG_mode - In this mode, the reverse transmitter is under the control of the Host for DOCSIS functionality. If the POD attempts to command the reverse transmitter with the **OOB_TX_tune_req()** message while the Host is operating in the DSG mode the Host will deny the tune request with a Tuning Denied – RF Transmitter busy status.

DSG_One-Way_mode - In this mode, the reverse transmitter must be disabled for both the OOB channel and the DOCSIS return channel. This mode could be used in one-way cable systems and for network diagnosis in two-way cable systems.

A default operational mode must be utilized when the Host and/or POD is unable to obtain the preferred operational mode. There are two potential default conditions that must be addressed. In particular:

a. Either the Host or the POD may not support the **Inquire_DSG_mode ()** and **Set_DSG_mode ()** messages.

b. The POD may not have acquired the preferred operational mode from the network due to possible network errors.

To insure backward compatibility in case (a) above, an Advanced Host will initialize in the default operational mode of OOB_mode. In case (b), the POD Module shall instruct the Host that the preferred operational mode of OOB_mode.

If the operational mode is either DSG_mode or One-Way_Mode, the POD Module may provide up to eight Ethernet MAC addresses and the number of header bytes to be removed from the DSG tunnel packets. In DSG or one-way mode, the Host must filter IP packets whose Ethernet destination address match any of the DSG_MAC_Adddresses specified, remove the specified number of header bytes from these packets, and generate a serialized bit-stream across the DRX pin.

Number_MAC_Addresses – The number of DSG MAC Addresses allocated by the CA/POD provider to carry DSG tunnels. A maximum of eight DSG tunnels per CA/POD provider are allowed.

DSG_MAC_Address– The Ethernet MAC addresses allocated by the CA/POD provider to carry the number of DSG tunnels specified by Number_MAC_Addresses.

Remove_Header_Bytes – The number of bytes to be removed from the DSG tunnel packets before generating a serial bit-stream. A value of zero implies that no header bytes be removed.

Table 8.9-S DSG packet_error Object Syntax		
Syntax	**# of bits**	**Mnemonic**
DSG_packet_error () {		
DSG_packet_error_tag	24	uimsbf
length_field()		
error_status	8	uimsbf
}		

DSG_packet_error () - The POD can inform the Host of errors that occur in receiving DSG packets. The Error_status indicates the type of error that occurred.

Error_status	Value
Byte_count_error	0x00
Reserved	0x01-FF

Byte_count_error – The POD did not receive the same number of bytes in the DSG packet as was signaled by the Host.

8.10 Generic IPPV Support

NOTE--The Generic IPPV Support resource is being deprecated, though it may still be in use; the preferred approach for supporting IPPV is to use the appropriate OCAP application.

The ***Generic IPPV Support*** resource provides Conditional Access information (in the POD Module) to the navigation application (in the Host). This allows subscriber access to Pay Per View functions such as purchase, cancellation and history review. The desired result is better subscriber recognition and increased IPPV usage.

If reported by the Host as an available resource and the POD module implements a Generic IPPV application, the POD module application shall create a session to the Generic IPPV resource to allow the Host to receive information on and purchase IPPV events.

ISO-8859-1 shall be used for the coding of text.

Table 8.10-A Generic IPPV Support Resources				
Resource	**Class**	**Type**	**Version**	**Identifier (hex)**
Generic IPPV Support	128	2	1	00800081

This creation includes the following objects:

Table 8.10-B Generic IPPV Support Objects			
Apdu_tag	**Tag value (hex)**	**Resource**	**Direction Host <-> POD**
Program_req()	9F8F00	Generic IPPV Support	→
Program_cnf()	9F8F01	Generic IPPV Support	←
Purchase_req()	9F8F02	Generic IPPV Support	→
Purchase_cnf()	9F8F03	Generic IPPV Support	←
Cancel_req()	9F8F04	Generic IPPV Support	→
Cancel_cnf()	9F8F05	Generic IPPV Support	←
History_req()	9F8F06	Generic IPPV Support	→
History_cnf()	9F8F07	Generic IPPV Support	←

8.10.1 Program_req() & Program_cnf()

The Host's navigation application shall use the ***Program_req()*** object to request the POD Module's CA information on a particular program.

The POD Module shall respond with the ***Program_cnf()*** object to the ***Program_req()*** request.

Table 8.10-C Program Request Object Syntax		
Syntax	**# of bits**	**Mnemonic**
program_req() {		
program_req_tag	24	uimsbf
length_field()		
transaction_id	8	uimsbf
transport_stream_id	16	uimsbf
program_number	16	uimsbf
source id	16	uimsbf
event_id	16	uimsbf
current_next indicator	8	uimsbf
reserved	7	
current_next	1	uimsbf
program_info_length	8	
for (i=0; i < program_info_length; i++) {		
ca_descriptor()		
/* ca descriptor at program level*/		
}		
}		

transaction_id – This field is a unique number generated by the Host to uniquely identify this transaction. The associated ***Program_cnf()*** message will include this ***Transaction_ID*** value. Hosts shall maintain a ***Transaction_ID*** counter and increment it by 1 (mod 256) for each new transaction.

transport_stream_id - A 16-bit unsigned integer field, in the range 0x0000 to 0xFFFF, that represents the MPEG-2 Transport Stream ID associated with the program being requested.

program_number - A 16-bit unsigned integer number indicating the program that is being requested.

source_id – A 16-bit unsigned integer number indicating the source_id of the program that is being requested. (This text should be inserted after the program_number field description.)

event_id – A 16-bit unsigned integer number specifying the event requested on the specified program_number. If the Event_ID is unknown, this field shall be set to all 0s.

current_next– Used to specify the current or next event on the specified program_number. Only relevant when Event_ID is set to 0. When not set, indicates that the current event is being requested. When set, indicates that the next event is requested.

program_info_length, CA descriptor – These fields shall be used by the Host to provide the POD Module with every program level ***CA descriptor*** of this MPEG program. The CA descriptor shall be, extracted from the PMT table by the Host navigation application.

Table 8.10-D Program Confirm Object Syntax		
Syntax	**# of bits**	**Mnemonic**
Program_cnf() {		
Program_cnf_tag	24	Uimsbf
Length_field()		
Transaction_ID	8	Uimsbf
Status_field	8	Uimsbf
If (Status_field == 0) {		
Option_nb	8	Uimsbf
For (Option_ID=1; I <= Option_nb;		
Option_ID++) {		
Purchase_type	8	Uimsbf
Purchase_price	16	Uimsbf
Purchase_validation	8	Uimsbf
Expiration_date	32	Uimsbf
Program_start_time	32	Uimsbf
Initial_Free_preview_duration	16	Uimsbf
Anytime_free_preview_duration	16	Uimsbf
Title_length	8	Uimsbf
for (J=0; J < Title_length; J++) {		
Title_txt	8	Uimsbf
}		
Text_length	8	Uimsbf
for (J=0; J < Text_length; J++) {		
Text_txt	8	Uimsbf
}		
Descriptor_length	16	Uimsbf
for (K=0; K < Desc_length; K++) {		
Descriptor()		Uimsbf
}	var	
}		
}		
}		

Status_field – This field returns the status of the ***Program_req().*** If the POD Module can provide the requested information on the pointed event then Status_field shall be set to 0x00. Otherwise it will be set to one of the following values.

Table 8.10-E Status Field Values for Program Confirm	
Status_field	**Value (hex)**
Request Granted	00
Request Denied – POD module busy	01
Request Denied – Unknown Event	02
Reserved	03-FF

- ***Option_nb*** – This field defines the number of options under which a particular event can be purchased

- ***Purchase_type*** – This field characterizes how the event may be purchased

Table 8.10-F Purchase Type Values for Program Confirm	
Purchase_type	**Value (hex)**
Viewing Only	00
Viewing and Right to Copy Once	01
Viewing and Right to Copy Unlimited	02
Subscription	03
Purchased for Viewing Only	04
Purchased with Viewing and Right to Copy Once	05
Purchased with Viewing and Right to Copy Unlimited	06
Un-purchasable	07
Reserved	08-FF

Viewing only - This program may be purchased for viewing only, without the right to make any copies, as defined by the operator.
NOTE: Through private agreements between a cable operator and content providers, the cable operator determines the pricing and right to copy options appropriate for his market.

Viewing and Right to Tape Copy Once - This program may be purchased for viewing and with the right to copy the analog video output and make one copy as defined by the operator.

Viewing and Right to Copy Unlimited - This program may be purchased for viewing and with the right to make unlimited copies as defined by the operator.

Subscription **-** This program is a subscription event, and is not purchasable as an IPPV event.

Purchased for Viewing Only **-**This program has already been purchased with viewing rights only, and without the right to make any copies as defined by the operator.

Purchased with Viewing and Right to Tape Copy Once **-**This program has already been purchased for viewing with the right to tape and right to make one copy as defined by the operator.

Purchased with Viewing and Right to Copy Unlimited – This program has already been purchased for viewing with the right to make unlimited copies as defined by the operator.

Un-purchasable **-** This is not a purchasable program.

Reserved – These values are currently undefined, but are reserved for future IPPV purchase options, including digital copyrights. These values may be expanded as they are defined.

Purchase_price – This 2-byte field provides event pricing information. The event price is given by the Denomination unit multiplied by the Value. For example, if the Denomination unit is 5 cents, and the Value is 79, the price would be $3.95.

Table 8.10-G Purchase Price for Program Confirm

Bit	7	6	5	4	3	2	1	0
	Denomination unit in cents (MS)							
	Value (LS)							

Purchase_validation – This parameter defines the level of validation the POD Module expects to validate the purchase.

Table 8.10-H Purchase Validation Value for Program Confirm	
Purchase_validation	**Value (hex)**
No CA validation required	00
PIN Code required for Purchase transaction	01
PIN Code required for Cancel transaction	02
PIN Code required for History transaction	03
PIN Code required for Purchase and Cancel transactions	04
PIN Code required for Purchase and History transactions	05
PIN Code required for Purchase, Cancel, History transactions	06
Reserved	07-FF

- ***Expiration_date*** – This field contains the expiration time of the event. It is a 32-bit unsigned integer quantity representing the expiration time as the number of seconds since 12 am, January 6th 1980.

- ***Program_start_time***: A 32 bit unsigned integer, defining the start time of the program, in GPS seconds since 12 AM January 6th, 1980.

- ***Initial_free_preview_duration***: A 16-bit unsigned integer, defining the duration of the free preview period. The duration is measured from the program_start_time.

- ***Anytime_free_preview duration***: A 16-bit unsigned integer, defining the duration of the Anytime_free_preview.

- ***Title_length, Title_txt*** – These fields allow the POD Module to provide a purchase option title.

- ***Text_length, Text_txt*** – These fields allow the POD Module to provide a purchase option text.

- ***Desc_length*** – A 16-bit unsigned integer that indicates the length of the block of optional descriptors to follow. If no descriptors are present, the length shall indicate zero.

- ***Descriptor()*** – A data structure of the form type-length-data, where *type* is an 8-bit descriptor type identifier, *length* is an 8-bit field indicating the number of bytes to follow in the descriptor, and *data* is arbitrary data. The syntax and semantics of the data are as defined for the particular type of descriptor. The **content_advisory_descriptor()** (as defined in section 6.7.4 of ATSC A/65) may be used to indicate the rating of the program. The program rating shall be coded according to the MPAA and V-Chip Rating and Content Advisories to be used for parental restrictions on program purchases.

8.10.2 Purchase_req() & Purchase_cnf()

The Host's navigation application shall use the ***Purchase_req()*** object to request a purchase of a particular program offer.

The POD Module shall respond with the ***Purchase_cnf()*** object to the ***Purchase_req()*** request.

Table 8.10-I Purchase Request Object Syntax

Syntax	# of bits	Mnemonic
Purchase_req() {		
Purchase_req_tag	24	Uimsbf
Length_field()		
Transaction_ID	8	Uimsbf
Option_ID	8	Uimsbf
PINcode_length	8	Uimsbf
For (I=0; I<=PINcode_length; I++) {		
PINcode_byte	8	Uimsbf
}		
}		

- ***PINcode_length, PINcode_byte*** – These fields allow the Host navigation application to pass the requested PIN code to the POD Module. In case no PIN code was requested, the ***PINcode_length*** is set to '0'.

Table 8.10-J Purchase Confirm Object Syntax		
Syntax	**# of bits**	**Mnemonic**
Purchase_cnf() {		
Purchase_cnf_tag	24	Uimsbf
Length_field()		
Transaction_ID	8	Uimsbf
Option_ID	8	Uimsbf
Status_field	8	Uimsbf
IPPVslot_ID	8	Uimsbf
Status_register	8	Uimsbf
Comment_length	8	Uimsbf
For (I=0; I<= Comment_length; I++) {		
Comment_txt	8	Uimsbf
}		
}		

- ***Status_field*** – This field returns the status of the ***Purchase_req().*** If the POD has validated the purchase, then ***Status_field*** shall be set to 0x00. Otherwise it will be set to one of the following values. When there are more than one reason to deny the purchase, ***Status_field*** is set to the lowest applicable value.

Table 8.10-K Status Field Values for Purchase Confirm	
Status_field	**Value (hex)**
Purchase Granted	00
Purchase Denied – POD Module busy	01
Purchase Denied – Unknown Transaction_ID or Option_ID	02
Purchase Denied – Invalid PIN code	03
Purchase Denied – Event already purchased	04
Purchase Denied – Blackout is active	05
Purchase Denied - Credit limit is exceeded	06
Purchase Denied - IPPV slot limit is exceeded	07
Purchase Denied – Spending limit is exceeded	08
Purchase Denied – Rating limit is exceeded	09
Purchase Denied – Check Comments	0A

Reserved	0B-FF

- ***Purchase Denied: IPPV_slot_limit is exceeded***: The POD is unable to make additional IPPV purchases until it has reported all of its unreported purchases to the headend.

- ***IPPVslot_ID*** - If ***Status_field*** is 0x00 (Purchase Granted) then ***IPPVslot_ID*** will contain the unique slot identifier that will later identify the purchasing transaction. If ***Status_field*** is any other value, ***IPPVslot_ID*** is reset to 0.

- ***Comment_length, Comment_txt*** – These fields allow the POD Module to explain, using plain text, why the purchase request has been granted or denied.

- ***Status_register*** - This field identifies the CA status of the program event. The designation of each bit is summarized in the following table.

Table 8.10-L Status Register for Purchase Confirm

Bit	7	6	5	4	3	2	1	0
	VPU	OPU	UPU	AUT	FRE	REP	CAN	VIE

- ***VPU*** is set to 1 when the program event has been purchased for viewing once.

- ***OPU*** is set to 1 when the program event has been purchased for taping once.

- ***UPU*** is set to 1 when the program event has been purchased for unlimited taping.

- ***AUT*** is set to 1 when the program event has been authorized.

- ***FRE*** is set to 1 when the free preview (initial or anytime) of the program event has been viewed.

- ***REP*** is set to 1 when the program event has been reported.

- ***CAN*** is set to 1 when the program event has been cancelled.

- ***VIE*** is set to 1 when the program event has been viewed.

8.10.3 Cancel_req() & Cancel_cnf()

The Host's navigation application shall use the ***Cancel_req()*** object to request a Cancellation of a particular purchased program offer.

The POD shall respond with the ***Cancel_cnf()*** object to the ***Cancel_req()*** request.

Table 8.10-M Cancel Request Object Syntax		
Syntax	**# of bits**	**Mnemonic**
Cancel_req() {		
Cancel_req_tag	24	Uimsbf
Length_field()		
IPPVslot_ID	8	Uimsbf
PINcode_length	8	Uimsbf
For (I=0; I<=PINcode_length; I++) {		
PINcode_byte	8	Uimsbf
}		
}		

Table 8.10-N Cancel Confirm Object Syntax		
Syntax	**# of bits**	**Mnemonic**
Cancel_cnf() {		
Cancel_cnf_tag	24	Uimsbf
Length_field()		
IPPVslot_ID	8	Uimsbf
Status_field	8	Uimsbf
Status_register	8	Uimsbf
Comment_length	8	Uimsbf
For (I=0; I<= Comment_length; I++) {		Uimsbf
Comment_txt	8	Uimsbf
}		
}		

- ***Status_field*** – This field returns the status of the ***Cancel_req().*** If the POD has validated the cancellation, then ***Status_field*** shall be set to 0x00. Otherwise it will be set to one of the following values. When there are more than one reason to deny the cancellation, ***Status_field*** is set to the lowest applicable value.

Table 8.10-O Status Field Values for Cancel Confirm	
Status_field	**Value (hex)**
Cancellation Granted	00
Cancellation Denied – POD Module is busy	01
Cancellation Denied – Unknown IPPVslot_ID	02
Cancellation Denied – Invalid PIN code	03
Cancellation Denied – Program already viewed or in progress	04
Reserved	05-09
Cancellation Denied - Check Comments	0A
Reserved	0B-FF

8.10.4 History_req() & History_cnf()

The Host's navigation application shall use the ***History_req()*** object to request the history of all purchased and cancelled program events held in POD memory.

The POD shall respond with the ***History_cnf()*** object to the ***History_req()*** request.

Table 8.10-P History Request Object Syntax		
Syntax	**# of bits**	**Mnemonic**
History_req() {		
History_req_tag	24	Uimsbf
Length_field()		
PINcode_length	8	Uimsbf
For (I=0; I<=PINcode_length; I++) {		Uimsbf
PINcode_byte	8	
}		Uimsbf
}		

- ***PINcode_length, PINcode_byte*** – These fields allow the Host navigation application to pass the requested PIN code to get IPPV history on events that required a PIN Code validation for History. In case no PIN code or a wrong PIN code is supplied, only history on events that do not require PIN Code validation for History will be provided.

Table 8.10-Q History Confirm Object Syntax		
Syntax	**# of bits**	**Mnemonic**
history_cnf() {		
history_cnf_tag	24	uimsbf
length_field()		
status_field	8	uimsbf
comment_length	8	uimsbf
for (i=0; i<= comment_length; i++) {		
comment_txt	8	uimsbf
}		
ippvslot_nb	8	uimsbf
for (i=0; i<= ippvslot_nb; i++) {		
ippvslot_id	8	uimsbf
purchase_type	8	uimsbf
purchase_price	16	uimsbf
status_register	8	uimsbf
purchase_date	32	uimsbf
cancel_date	32	uimsbf
event_date	32	uimsbf
title_length	8	uimsbf
for (j=0; j < title_length; j++) {		
title_txt	8	uimsbf
}		
text_length	8	uimsbf
for (j=0; j < text_length; j++) {		
text_txt	8	uimsbf
}		
descriptor_length	16	uimsbf
for (k=0; k < desc_length; k++) {		
descriptor()	var	
}		
}		
}		

- ***Status_field*** – This field returns the status of the ***History_req().*** If the POD has validated the History request, then ***Status_field*** shall be set to 0x00. Otherwise it will be set to one of the following values.

Table 8.10-R Status Field Values for History Confirm	
Status_field	**Value (hex)**
History Granted	00
History Denied – POD Module is busy	01
Reserved	02
History Denied – Invalid PIN code	03
Reserved	04-09
History Denied - Check Comments	0A
Reserved	0B-FF

- ***Purchase_date, Cancel_date, Event_date*** – These fields contain respectively the purchase time, the cancel time and the starting time of the event. They are 32-bit unsigned integer quantities representing the time as the number of seconds since 12 am, January 6th 1980.

- If the ***Cancel_date*** field contains all FFFFs, this indicates that no appropriate value is available for this field.

8.11 Specific Application Support

The ***Specific Application Support*** resource is intended for use when a vendor-specific application, which resides in either the POD or the Host, needs to communicate a private set of objects across the interface. Support for this resource is required in the Host and POD Module. The POD shall establish at least one session for communication with the Specific Application Support Resource.

8.11.1 Specific Application Support Connectivity

The POD Module shall open one or more ***Specific Application Support*** (SAS) sessions for private communications between vendor-specific POD Module applications and private Host applications, as shown in Figure 8.11-1. The POD Module, as the initiator of the sessions, is responsible for associating each session (by session number) with the appropriate vendor-specific POD Application. When a private Host application is ready to establish a connection with POD Module, an SAS Connect Request (s**as_connect_rqst**) message is sent to the POD over any opened SAS session. The POD 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 POD Module application. This session number, along with the private Host Application ID is returned to the Host via the SAS Connect Confirm message (s**as_connect_cnf**). This operation establishes the communication path between a specific pair of applications (vendor-specific POD application, private Host application).

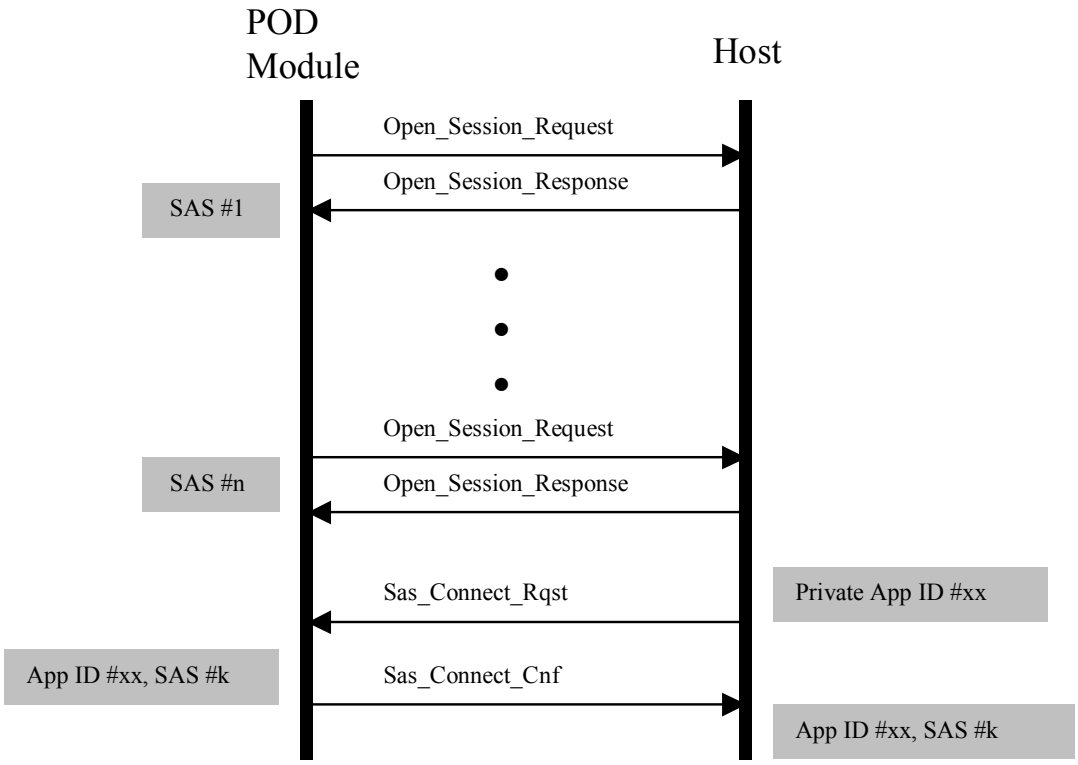

Figure 8.11-1

In some instances, the POD Module may receive an s**as_connect_rqst** before a session has been opened for the associated vendor-specific Application, as shown in Figure 8.11-2. In this case, the Pod Module shall establish the necessary SAS session and then respond with s**as_connect_cnf**.

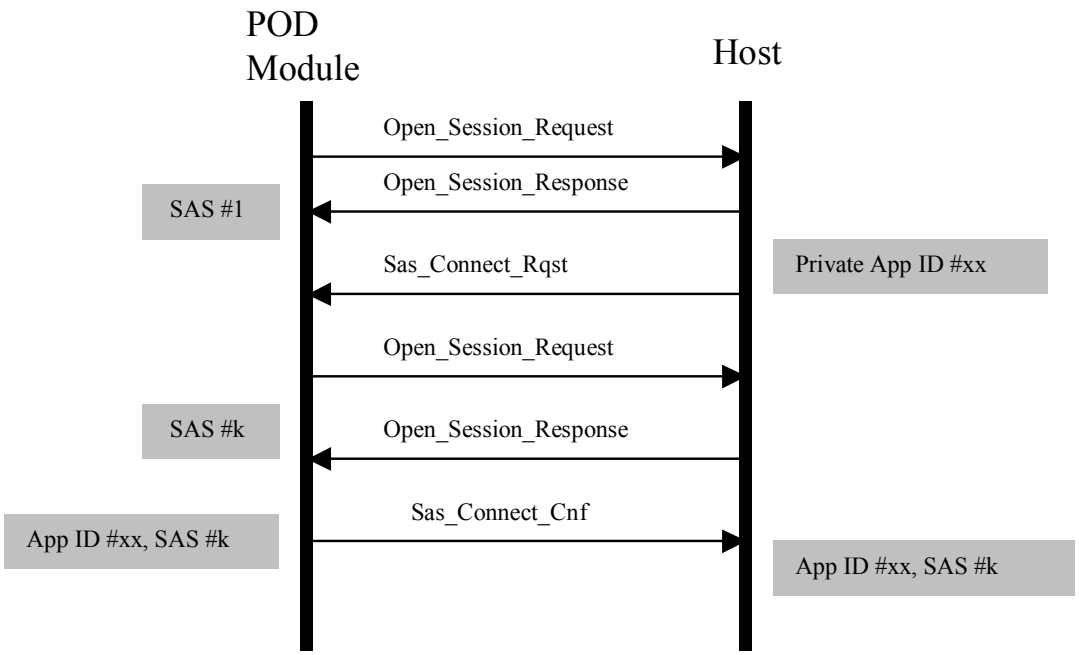

Figure 8.11-2

8.11.2 Resource Identifier

Table 8.11-A Specific Application Support Resource				
Resource	**Class**	**Type**	**Version**	**Identifier (hex)**
Specific Application Support	144	1	1	00900041

8.11.3 Application Objects

The **Specific Application Support** resource includes seven APDU's as described in the following table:

Table 8.11-B Specific Application Support Objects			
Apdu_tag	**Tag value (hex)**	**Resource**	**Direction Host <-> POD**
sas_connect_rqst()	9F9A00	SAS	→
sas_connect_cnf()	9F9A01	SAS	←
sas_data_rqst()	9F9A02	SAS	←→
sas_data_av()	9F9A03	SAS	←→
sas_data_cnf()	9F9A04	SAS	←→
sas_server_query()	9F9A05	SAS	←→
sas_server_reply()	9F9A06	SAS	←→

8.11.3.1 sas_connect_cqst() & cas_connect_cnf()

The Host shall send a s***as_connect_rqst()*** APDU to the POD Module to establish a connection between a private Host application and the corresponding POD Module vendor-specific application. The Pod shall reply with an s***as_connect_cnf()*** APDU to inform the Host of which SAS session is to be used for this connection.

8.11.3.1.1 sas_connect rqst()

Table 8.11-C sas_connect_rqst Object Syntax		
Syntax	**# of bits**	**Mnemonic**
sas_connect_rqst(){		
sas_connect_rqst_tag	24	uimsbf
Length_field()		
private_host_application_id	64	
}		uimsbf

8.11.3.1.2 sas_connect_cnf()

Table 8.11-D sas_connect_cnf Object Syntax		
Syntax	**# of bits**	**Mnemonic**
sas_connect_cnf(){		
sas_connect_cnf_tag	24	uimsbf
Length_field()		
private_host_application_id	64	uimsbf
sas_session_status	8	uimsbf
reserved	16	uimsbf
}		

where:

private_host_application_id This is a unique identifier of the private Host Application.

Informative Note: There is no need to register Private_Host_Application_IDs 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 POD modules.

sas_session_status The status of the requested connection as defined in the following table.

Table 8.11-E sas_session_status	
sas_session_status	**Value (Hex)**
Connection established	00
Connection denied – no associated vendor-specific POD application found	01
Connection denied – no more connections available	02
Reserved	03-FF

sas_session _nb The session number to be used for the designated Specific Application communications.

8.11.3.2 sas_data_rqst(), sas_data_av(), & sas_data_cnf()

Once a communication path has been established between the application pair (vendor-specific POD application, private Host application) via an SAS session, each of the applications can utilized the SAS APDUs to communicate with the other. The APDUs defined in this section are bi-directional in that they can originate from either side of the Host-POD Interface. The **sas_data_rqst()**APDU is used by one application to inform the other application that it is ready to process incoming data. The application which receives this APDU responds with an **sas_data_av()** APDU. When an application has data to send across the Host-POD Interface, an **sas_data_av()** APDU is sent. The receiving application responds with an s**as_data_cnf()** APDU to acknowledge that it is preparing to receive the available data.

8.11.3.2.1 sas_data_rqst()

Table 8.11-F sas_data_rqst Object Syntax		
Syntax	**# of bits**	**Mnemonic**
sas_data_rqst(){		
sas_data_rqst_tag	24	uimsb
Length_field()		
reserved	16	uimsb
}		

8.11.3.2.2 sas_data_av()

Table 8.11-G sas_data_av Object Syntax		
Syntax	**# of bits**	**Mnemonic**
sas_data_av(){		
sas_data_av_tag	24	uimsb
Length_field()		
reserved	16	uimsb
sas_data_status	8	uimsb
transaction_nb	8	uimsb
}		

8.11.3.2.3 sas_data_av_cnf()

Table 8.11-H sas_data_cnf Object Syntax		
Syntax	**# of bits**	**Mnemonic**
sas_data_av_cnf(){		
sas_data_av_cnf_tag	24	uimsbf
Length_field()		
reserved	16	uimsbf
transaction_nb	8	uimsbf
}		

where:

sas_data_status The status of the available data defined in the following table.

Table 8.11-I sas_data_status	
sas_data_status	**Value (Hex)**
Data Available	00
Data Not Available	01
Reserved	02-FF

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 will be set to zero.

8.11.3.3 sas_server_query() & sas_server_reply()

When data availability has been confirmed, an ***sas_server_query()*** APDU is sent to initiate the transfer of Application Specific data. The ***sas_server_reply()*** APDU shall be used to respond to the query and transfer data.

8.11.3.3.1 sas_server_query()

Table 8.11-J sas_server_query Object Syntax		
Syntax	**# of bits**	**Mnemonic**
sas_server_query(){		
sas_server_query_tag	24	uimsb
Length_field()		
reserved	16	uimsb
transaction_nb	8	uimsb
}		

8.11.3.3.2 sas_server_reply()

Table 8.11-K sas_server_reply Object Syntax		
Syntax	**# of bits**	**Mnemonic**
sas_server_reply(){		
sas_server_reply_tag	24	uimsbf
Length_field()		
reserved	16	uimsbf
transaction_nb	8	uimsbf
Message_length	16	uimsbf
for (i =0; i< message_length; i++)		
{		
message_byte	8	uimsbf
}		
}		

8.12 Generic Feature Control Support

The ***Generic Feature Control*** resource enables the Host device to receive control of features which are considered generic to Host devices (set-top terminal, television, VCR, etc.). 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 POD Module 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 POD module, the POD module shall open only one session to the Host and should never close the session.

8.12.1 Parameter Storage

8.12.1.1 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.

8.12.1.2 POD

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

8.12.2 Parameter Operation

8.12.2.1 Feature List Exchange

Immediately after the session to the Generic Feature Control resource has been established, the POD module shall query the Host to determine which generic features are supported in the Host (**feature_list_req**). After the POD module receives the generic feature list from the Host (**feature_list**), the POD module shall send its confirmation of the feature list to the Host (**feature_list_cnf**). The Host shall then query the POD module to determine which generic features are supported in the POD module and the headend (**feature_list_req**). The POD module 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.

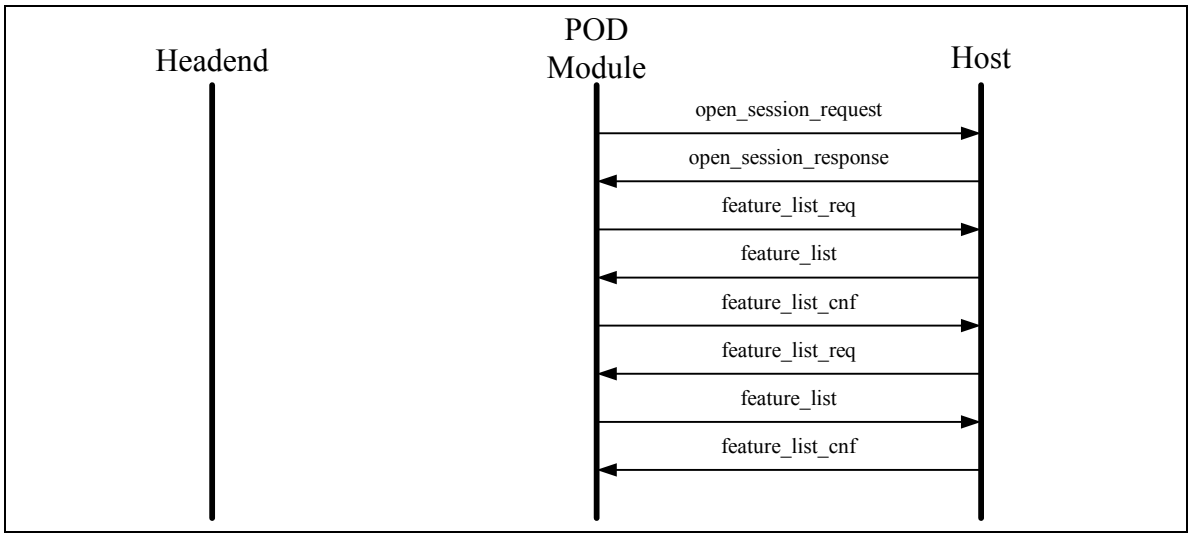

Figure 8.12-1 Generic Feature List Exchange

If the generic feature list on the Host or the POD module changes, then the changed device shall send a generic feature list changed APDU to the other device (**feature_list_changed**). The other device shall then perform the generic feature list exchange to obtain the new list.

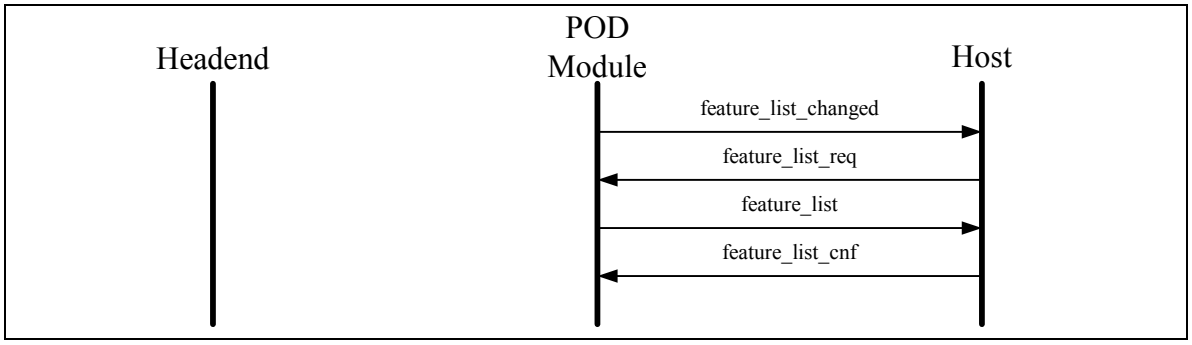

Figure 8.12-2 POD Module Feature List Change

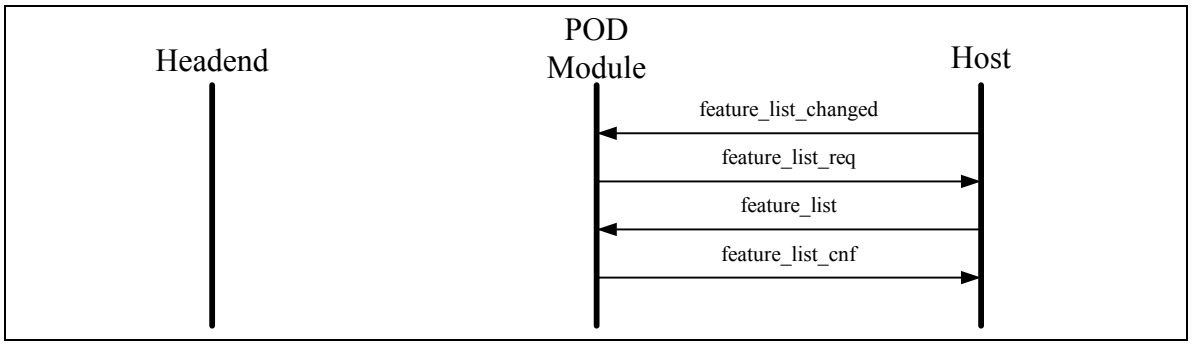

Figure 8.12-3 Host Feature List Change

8.12.3 Host to POD Module Transfer

After the feature exchange has occurred, the POD module may request the Host to send its feature parameters (**feature_parameters_req**). After any request, the Host shall send to the POD module the parameters for all the generic features in the Host's generic feature list (**feature_parameters**). The POD module shall reply with the confirmation (**feature_parameters_cnf**). The POD module may utilize these generic feature parameters, transfer them to the headend, or ignore them.

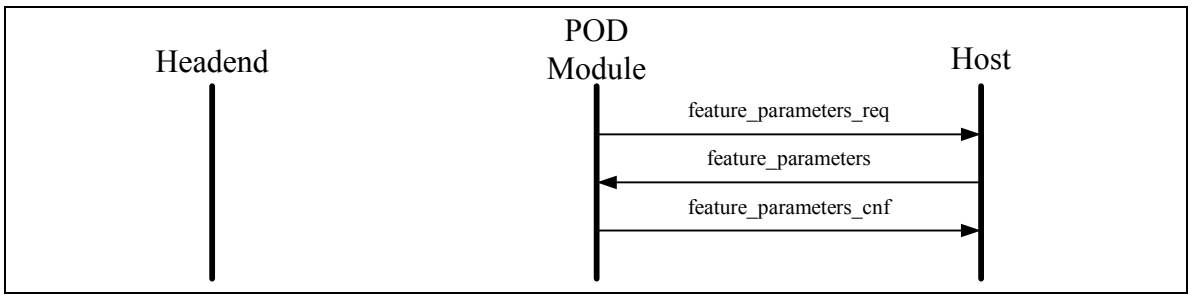

Figure 8.12-4 Host to POD Module Feature Parameters

Anytime any of the parameters of the generic features that are in the POD module generic feature list are changed in the Host, for whatever reason, the Host shall transmit these new parameters to the POD module (**feature_parameters**). The POD module shall reply with the confirmation (**feature_parameters_cnf**).

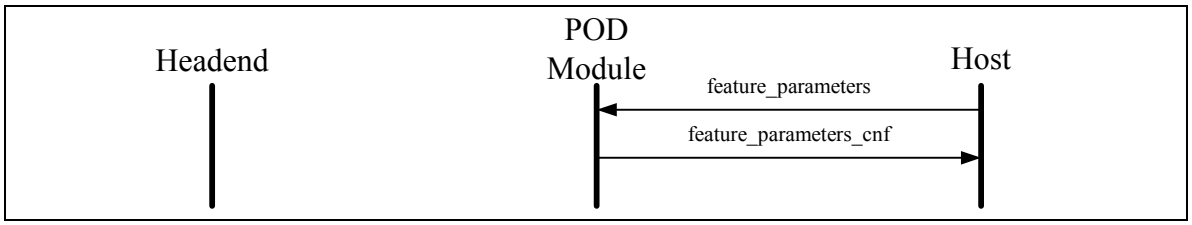

Figure 8.12-5 Host Parameter Update

The POD module may request, at any time the session is open and the generic feature list exchange has occurred, the current parameters in the Host. The POD module shall do this by sending a feature parameters request (**feature_parameters_request**) as shown in figure 8.12-4.

8.12.3.1 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 POD module receives the generic feature's parameters is proprietary to the POD manufacturer.

After the session has been established, when the POD module receives a message from the headend containing generic feature parameters, the POD module shall transfer this information to the Host (**feature_parameters**). The Host shall replace the parameters with the values in the APDU. If the POD module 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.

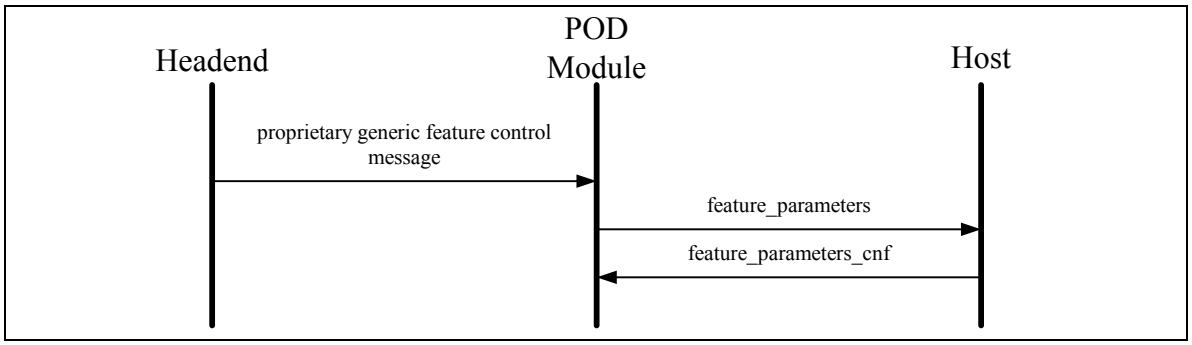

Figure 8.12-6 POD Module to Host Feature Parameters

8.12.4 Resource Identifier

The following resource identifier shall be utilized for the Host to open in the POD module.

Table 8.12-A Generic Feature Control Resource				
Resource	**Class**	**Type**	**Version**	**Identifier (hex)**
Generic Feature Control	42	1	1	002A0041

8.12.5 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 8.12-B Generic Feature IDs	
Feature ID	**Feature**
00	Reserved
01	RF Output Channel
02	Parental Control PIN
03	Parental Control Settings
04	IPPV PIN
05	Time Zone
06	Daylight Savings Control
07	AC Outlet
08	Language
09	Rating Region
0A	Reset PIN
0B	Cable URLs
0C	Emergency Alert Location Code
0D-3F	Reserved for future use
70-FF	Reserved for proprietary use

8.12.6 Application Objects

The following is a list of the application objects (APDUs).

Table 8.12-C Generic Feature Control Objects			
Apdu_tag	**Tag value (hex)**	**Resource**	**Direction Host <-> POD**
Feature_list_req	9F 98 02	Generic Feature Control	↔
Feature_list	9F 98 03	Generic Feature Control	↔
Feature_list_cnf	9F 98 04	Generic Feature Control	↔
Feature_list_changed	9F 98 05	Generic Feature Control	↔
Feature_parameters_req	9F 98 06	Generic Feature Control	←
Feature_parameters	9F 98 07	Generic Feature Control	↔
Feature_parameters_cnf	9F 98 08	Generic Feature Control	↔

8.12.6.1 Feature List Request

Either the Host or POD shall send this APDU to the POD module or the Host to query the generic features that it supports.

Table 8.12-D Feature List Request Object Syntax		
Syntax	**# of bits**	**Mnemonic**
feature_list_req() {		
feature_list_req_tag	24	uimsbf
length_field()		
}		

- feature_list_req_tag Value = 0x9F9802

8.12.6.2 Feature List

After receiving the feature_list_req, the Host or POD module shall transmit this APDU to the POD module or Host which lists the generic features that the POD module and headend support control of.

Table 8.12-E Feature List Object Syntax		
Syntax	**# of bits**	**Mnemonic**
Feature_list() {		
Feature_list_tag	24	uimsbf
Length_field()		
Number_of_features	8	uimsbf
For(i=0; i<number_of_features; i++){		
feature_id	8	uimsbf
}		
}		

- **feature_list_tag** Value = 0x9F9803
- **number_of_features** Number of features to report
- **feature_id** Assigned feature ID number as defined in section 8.12.5 of this document.

8.12.6.3 Feature List Confirmation

After receiving the feature_list APDU, the Host or POD module shall transmit this APDU to the POD module or Host to confirm receiving it.

Table 8.12-F Feature List Confirm Object Syntax		
Syntax	**# of bits**	**Mnemonic**

feature_list_cnf() {
feature_list_cnf_tag
length_field()
}
	24	uimsbf

- feature_list_cnf_tag Value = 0x9F9804

8.12.6.4 Feature List Changed

Either the Host or the POD module shall send this APDU to inform the POD module or the Host that its feature list changes.

Table 8.12-G Feature List Changed Object Syntax		
Syntax	**# of bits**	**Mnemonic**

feature_list_changed() {
feature_list_changed_tag
length_field()
}
	24	uimsbf

8.12.6.5 Feature Parameters Request

After the feature exchange has occurred, the POD module may, at any time, send the feature parameters request to the Host. The Host shall not send this APDU to the POD module.

Table 8.12-H Feature Parameter Request Object Syntax		
Syntax	**# of bits**	**Mnemonic**

feature_parameters_req() {
feature_parameters_req_tag
length_field()
}
	24	uimsbf

- **feature_parameters_req_tag** Value = 0x9F9806

8.12.6.6 Feature Parameters

The Host shall send the **feature_parameters** of its feature list to the POD module 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 POD module. The POD module may ignore any feature parameters which it does not support.

The POD module may send the **feature_parameters** APDU at any time in response to a message that it receives from the headend.

Table 8.12-I Feature Parameters Object Syntax		
Syntax	**# of bits**	**Mnemonic**
feature_parameters() {		
feature_parameters_tag	24	uimsbf
length_field()		
number_of_features	8	uimsbf
for(i=0; i<number_of_features; i++){		
feature_id	8	uimsbf
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) {		
ippv_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()
}
}
}		

- **feature_parameters_tag** Value = 0x9F9807
- **number_of_features** Number of features to report
- **feature_id** Assigned feature ID number as defined in section 7.1.2.5 of this document.
- **rf_output_channel** RF output channel
- **p_c_pin** Parental Control PIN parameter
- **p_c_settings** Parental Control Settings parameter.
- **ippv_pin** IPPV PIN parameter.
- **time_zone** Time Zone parameter.

  This feature is only utilized if the cable system crosses time zones.
- **daylight_savings** Daylight Savings parameter.

  This feature is only utilized if the cable system encompasses both areas which recognize daylight savings and those which do not.
- **ac_outlet** AC Outlet parameter.
- **language** Language parameter.
- **rating_region** Rating Region parameter.
- **reset_pin** Reset PIN's
- **cable_urls** URL list
- **ea_location_code** EAS location code

8.12.6.7 Feature Parameters Confirmation

When the POD module or Host receives the feature_parameter APDU, it shall respond with the feature parameters confirmation APDU.

Table 8.12-J Feature Parameters Confirm Object Syntax		
Syntax	**# of bits**	**Mnemonic**
Feature_parameters_cnf() {		
feature_parameters_cnf_tag	24	uimsbf
length_field()		
number_of_features	8	uimsbf
for(i=0; i<number_of_features; i++){		
feature_id	8	uimsbf
status	8	uimsbf
}		
}		

feature_parameters_tag Value = 0x9F9808

number_of_features Number of features to report

feature_ID Assigned feature ID number as defined in section 8.12.5 of this document.

status Status of feature parameter
00 Accepted
01 Denied – feature not supported
02 Denied – invalid parameter
03 Denied – other reason
04-FF Reserved

8.12.7 Feature Parameter Definition

Each generic feature will have a parameter definition uniquely assigned. These parameters will be consistent for all APDUs. The following sections define these parameters if the specified features are implemented.

8.12.7.1 RF Output Channel Parameters

Table 8.12-K RF Output Channel Parameters Syntax		
Syntax	**# of bits**	**Mnemonic**
Rf_output_channel() {		
Output_channel	8	uimsbf
Output_channel_ui	8	uimsbf
}		

- **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-FF Reserved

8.12.7.2 Parental Control PIN Parameters

Table 8.12-L Parental Control PIN Parameters		
Syntax	**# of bits**	**Mnemonic**
P_c_pin() {		
P_c_pin_length	8	Uimsbf
For(i=0; i<p_c_pin_length; i++) {		
p_c_pin_chr	8	Uimsbf
}		
}		

- **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/IEC 10646-1:1993, Information technology — Universal Multiple-Octet CodedCharacter Set (UCS) — Part 1: Architecture and Basic Multilingual Plane. The first character received is the first character entered by the user.

8.12.7.3 Parental Control Settings Parameters

Table 8.12-M Parental Control Settings Parameters		
Syntax	**# of bits**	**Mnemonic**
P_c_settings() {		
P_c_factory_reset	8	Uimsbf
p_c_channel_count	16	Uimsbf
for(i=0; i<p_c_channel_count; i++) {		
reserved	4	'1111'
major_channel_number	10	Uimsbf
minor_channel_number	10	Uimsbf
}		
}		

- **p_c_factory_reset** Perform factory reset on parental control feature.
  00-A6 No factory reset
  A7 Perform factory reset
  A8-FF No factory reset

- **p_c_channel_count** Number of virtual channels to place under parental control

- **major_channel_number** For two-part channel numbers, this is the major number for a virtual channel to place under parental control. For one-part channel number semantics, see Table 5.27 of ANSI/SCTE 65 2002. Both two-part and one-part channel numbers shall comply with ANSI/SCTE 65.

- **minor_channel_number** For two-part channel numbers, this is the minor number for a virtual channel to place under parental control. For one-part channel number semantics, see Table 5.27 of ANSI/SCTE 65 2002. Both two-part and one-part channel numbers shall comply with ANSI/SCTE 65.

8.12.7.4 IPPV PIN Parameters

Table 8.12-N IPPV PIN Parameters		
Syntax	**# of bits**	**Mnemonic**
IPPV_pin() {		
IPPV_pin_length	8	uimsbf
for(i=0; i<IPPV_pin_length; i++) {		
IPPV_pin_chr	8	uimsbf
}		
}		

- **IPPV_pin_length** Length of the Purchase PIN. Maximum length is 255 bytes.

- **IPPV_pin_chr** Purchase PIN character. The value is coded as defined in ISO/IEC 10646-1:1993, Information technology. Universal Multiple-Octet Coded Character Set (UCS). Part 1: Architecture and Basic Multilingual Plane. The first character received is the first character entered by the user.

8.12.7.5 Time Zone Parameters

Table 8.12-O Time Zone Parameters		
Syntax	**# of bits**	**Mnemonic**
Time_zone() {		
time_zone_offset	16	tcimsbf
}		

- **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.

8.12.7.6 Daylight Savings Parameters

Table 8.12-P Daylight Savings Parameters		
Syntax	**# of bits**	**Mnemonic**
Daylight_savings() {		
daylight_savings_control	8	uimsbf
}		

- **daylight_savings_control** Enable daylight savings time control in the Host.
  00 Ignore this field

01 Do not use daylight savings
02 Use daylight savings
03-FF Reserved

8.12.7.7 AC Outlet Parameters

Table 8.12-Q AC Outlet Parameters		
Syntax	**# of bits**	**Mnemonic**
Ac_outlet() ac_outlet_control }	8	uimsbf

- **ac_outlet_control** AC outlet control
  00 User setting
  01 Switched AC outlet
  02 Unswitched AC outlet (always on)
  03-FF Reserved

8.12.7.8 Language Parameters

Table 8.12-R Language Parameters		
Syntax	**# of bits**	**Mnemonic**

Language() {
language_control
}
	24	Uimsbf

- **language_control** Language setting using ISO 639, Code for the Representation of Names of Languages, 1988, and ISO CD 639.2, Code for the Representation of Names of Languages: alpha-3 code, Committee Draft, dated December 1994.

8.12.7.9 Rating Region Parameters

Table 8.12-S Rating Region Parameters		
Syntax	**# of bits**	**Mnemonic**

Rating_region() {
Rating_region_setting
}
	8	Uimsbf

- **rating_region_setting** The 8-bit unsigned integer number defined in ATSC A/65, December 23, 1997, "Program and System Information Protocol for Terrestrial Broadcast and Cable" that defines the rating region in which the Host resides.

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

8.12.7.10 Reset PIN

Table 8.12-T Reset PIN		
Syntax	**# of bits**	**Mnemonic**

Reset_pin() {
reset_pin_control
}
	8	Uimsbf

- **reset_pin_control** Defines the control of resetting PIN(s).
  The reset value is defined by the manufacturer and not covered in this document.
  00 Do not reset any PIN
  01 Reset parental control PIN
  02 Reset purchase PIN
  03 Reset parental control and purchase PIN
  04-FF Reserved

8.12.7.11 Cable URLs

Table 8.12-U Cable URLs		
Syntax	**# of bits**	**Mnemonic**
Cable_urls() {		
number_of_urls	8	uimsbf
for(i=0; I<number_of_urls; I++) {		
url_type	8	uimsbf
url_length	8	uimsbf
for(I=0; i<url_length; I++) {		
url_chr	8	uimsbf
}		
}		
}		

- **number_of_urls** Number of URLs defined.

- **url_type** Type of URL, according to the following:
  00 Undefined
  01 Web portal URL

02 EPG URL
03 VOD URL
04-FF Reserved

- **url_length** Length of the URL.
  The maximum length is 255 bytes.

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

8.12.7.12 Emergency Alert Location Code

Table 8.12-V Emergency Alert Location Code

Syntax	# of bits	Mnemonic
EA_location_code() {		
state_code	8	uimsbf
county_subdivision	4	uimsbf
reserved	2	'11'
county_code	10	uimsbf
}		

- **state_code, county_subdivision, county_code**
  These fields shall be as defined in SCTE 18 2002.

8.13 POD Module Firmware Upgrade

8.13.1 Introduction (Informative)

The POD module will require that its firmware be upgraded occasionally. The mechanism of upgrading this firmware is unique to each POD module manufacturer's system. This operation can be facilitated by adding the interface outlined in this section. New versions of the Homing and Host Control resources are utilized which encapsulates the previous operations of the resources but adds new operations for facilitating the firmware upgrade.

8.13.1.1 Summary (Informative)

8.13.1.1.1 Firmware Upgrade (Informative)

A POD module may be designed to be capable of having its firmware reprogrammed. Generally, this is implemented with flash memory or battery backed up RAM. Occasionally, this firmware will be upgraded. There are generally two paths in which

the firmware can be upgraded: 1) over the cable network using the QAM inband channel, and 2) over the cable network using the QPSK OOB channel. Upgrade can be accomplished either by the methods defined in this document or by other methods. Since different system implementations affect the method of POD module upgrade, two types of upgrade states are offered, a "delayed" and an "immediate".

8.13.1.1.1.1 Delayed Upgrade (Informative)

When the POD module detects that a firmware upgrade is required and immediate upgrade has not been requested by the headend, then if the Homing resource is not already open, and the POD module requires utilizing the Homing resource, it will open a session to the Homing resource if it is not already open. The POD module will then wait until the open_homing APDU is received prior to beginning the upgrade. The POD module will inform the Host through the firmware_upgrade APDU that it will be doing a firmware upgrade. After receiving the firmware_upgrade_reply APDU, the POD module can use the Host Control resource to tune either the QAM or QPSK tuner in the Host to the appropriate frequency and modulation type. The Host will not modify the selected tuner until the POD module has indicated that the firmware upgrade has finished by sending the firmware_upgrade_complete APDU or a timeout condition occurs. The firmware_upgrade_complete APDU can also indicate to the Host whether a PCMCIA reset, POD reset, or no reset is required by the POD module. After receiving the firmware_upgrade_complete APDU, the Host will be free to change the QAM tuner.

The Host will send the open_homing APDU when it is in standby mode (power applied but in the "Off" state) as defined in [1].

8.13.1.1.1.2 Immediate Upgrade (Informative)

There are conditions in which the POD module will need to perform an immediate upgrade. When this is required, the POD module will have the option to use the interface upgrade mechanisms defined in this document. If using these mechanisms, the POD module will open the Homing resource, if it is not already open, and send a firmware_upgrade APDU. The Host will reply with a firmware_upgrade_reply when it is ready. The POD module will use the Host Control APDUs to tune either the QAM or QPSK tuner in the Host to the appropriate frequency and modulation type. The Host will not interrupt this process until it has either received a firmware_upgrade_complete APDU or a timeout condition occurs. An optional text message is included in the APDU to display to the user if the Host is not in standby.

Additionally, it is possible that an outside occurrence, such as a power failure, may cause the firmware to become corrupted. If this occurs, then the POD module is incapable of performing most of its functions. It is still able to perform some functions if ROM code is included in the design. Generally, this ROM code is fairly small since it is not upgradeable and is utilized only for verification of the firmware and loading the firmware in case of corruption. This ROM code, called bootloader

code in this document, must be carefully designed and verified since it cannot be modified.

The bootloader is called upon reset of the POD module CPU. It first performs basic initialization operations, then tests the main program memory to insure that it is valid, and if it is valid, starts executing out of the main firmware memory. The problem occurs that if the main program memory is not valid, then a mechanism is needed to allow for recovery of the main firmware.

For this rare condition, under this proposal, the bootloader will contain firmware which will allow the POD module to utilize the APDUs defined in this document for an immediate upgrade.

8.13.1.1.2 Inband Upgrade Considerations (Informative)

If the POD module utilizes the QAM inband channel for upgrades, then for normal upgrades it should utilize the delayed upgrade. The Host should then notify the POD module that it can upgrade when the Host is placed in the standby state by the user. If the Host has been in the on state for a long period of time or the POD module bootloader has detected corrupted memory, then an immediate upgrade is required in which case the Host will give control of the QAM tuner immediately to the POD module, independent of its state.

8.13.1.1.3 OOB Upgrade Considerations (Informative)

If the POD module utilizes the QPSK OOB channel for upgrades, then its operation will depend on whether applications can still operate while performing an upgrade. If they cannot, a delayed firmware upgrade should be used. The POD module will have to open the Homing resource and wait until the open_homing APDU is received prior to beginning the upgrade. If applications can operate during an upgrade, then an immediate firmware upgrade can be used.

8.13.1.1.4 Other Homing Operations (Informative)

If desired, the POD module can use the Homing resource for receiving other parameters over the inband channel when the Host is in standby state. If this is utilized, then the upgrade option should not be used so as to allow the Host to return to the on state at the users request.

8.13.2 Implementation

8.13.2.1 Introduction (Normative)

In order to meet these operations, there is a need for a mechanism whereby the POD module can inform the Host that a firmware upgrade is required, a optional text message to the user, and the type of upgrade path.

Note that it is the responsibility of the Host to inform the user when an immediate upgrade occurs and to determine when the recovery can occur for delayed upgrades.

8.13.2.2 Reset Implementation (Normative)

After the POD module has finished its firmware upgrade, it will either send the firmware_upgrade_complete APDU with the appropriate reset type or simply timeout based on the timeout type.

8.13.2.3 Host Operation (Normative)

While the POD module is performing its upgrade operation, its ability to support the normal POD interface may range from severely limited to entirely unimpaired. To accommodate any case some modifications to normal operation are required. The following is a list of those modifications as well as requirements to the Host.

1. If enabled by the firmware_upgrade APDU, the POD module shall still respond to the transport layer polls with a 5 second timeout. If the POD module fails to respond to the poll within 5 seconds, the Host shall perform a PCMCIA reset on the POD module.

2. The POD module may not be able to support session or application layer operations. The Host shall not initiate any new sessions or initiate any new application layer operations after transmitting a firmware_upgrade_reply APDU until either the firmware_upgrade_complete APDU is received or the POD module times out. However, the Host shall maintain all session connections so that if the POD module cancels the firmware upgrade, normal operation can continue. The POD shall provide a finite timeout value to the Host sufficient for all tuning and firmware update operations to complete before the timeout occurs, when the firmware update operation occurs under nominal network operating conditions. The timeout value is set taking into account relevant factors such as the nominal time for the POD module to determine and tune to the frequency or location of the update information, nominal repetition rate of the update information, and nominal size and transmission rate of the update information. Prior to initiating a firmware upgrade operation, the POD module shall close any sessions it will not be able to support during that operation, and restore sessions as necessary following completion (or cancellation) of the upgrade operation so that normal operation can continue. Note: If the POD module has set the response_interval in the system_time_inq() APDU to a non-zero value, the Host may continue to transmit the system_time() APDU to the POD module after transmitting a firmware_upgrade_reply() APDU.

3. If the download_timeout_period expires, the Host shall perform a PCMCIA reset on the POD module.

If the POD module sends a firmware_upgrade_complete with “No Reset Required”, then the Host shall resume normal operation with the POD module in all respects, including timeout and reset operation.

8.13.2.3.1 Timeout Types (Normative)

The firmware_upgrade APDU includes a variable called timeout_type which defines the type of timeout the Host is to utilize during a firmware upgrade. This can include the normal 5 second transport timeout and/or a download timeout timer which starts from the last firmware_upgrade APDU received or neither. It is highly recommended that the POD module not use the "No timeout" option.

8.13.2.3.2 Transport Layer Timeout (Normative)

Since the POD module may be incorporating flash memory which takes a longer time to program than the transport layer timeout period (5 seconds), using option 02 or 03 on the timeout_type variable in the firmware_upgrade APDU will cause the Host to cease implementing this timeout until either a firmware_upgrade_complete APDU is received or the download_timeout_period from the last firmware_upgrade APDU has passed, in which case the Host will perform a PCMCIA reset.

8.13.2.4 Upgrade Cancellation (Normative)

If the POD module cancels its firmware upgrade, then it will send the firmware_upgrade_complete APDU with the reset type set to 02, "no reset required".

8.13.2.5 Flowchart (Informative)

Figure 8.13.1 is a flowchart which shows the POD module/Host interface which uses the POD module upgrade methods defined in this document.

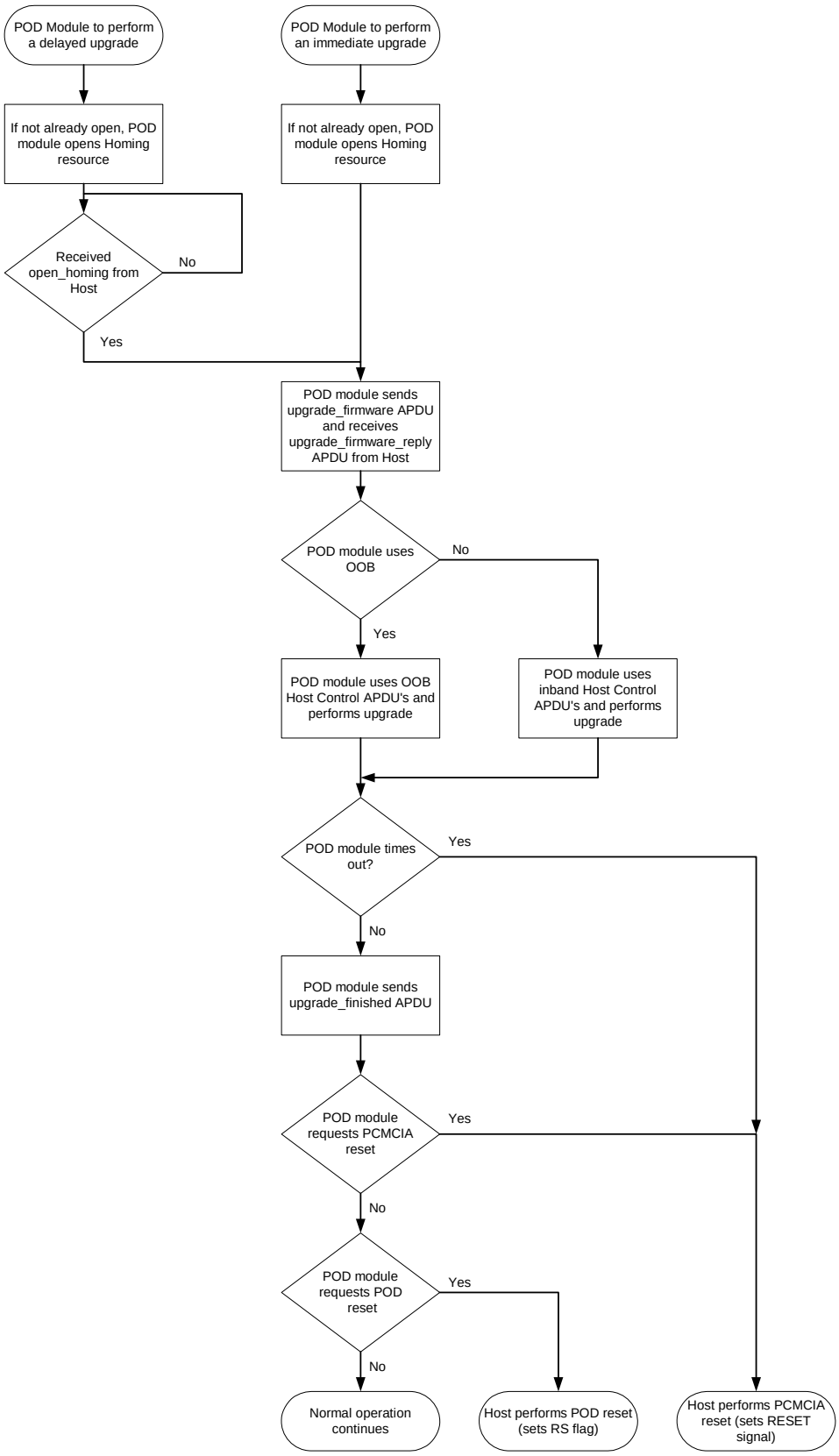

Figure 8.13-1 Firmware Upgrade Flowchart

8.13.3 Homing Resource (Normative)

8.13.3.1 Homing Resource Definition (Normative)

As defined in section 8.8.1.1 of [1], the Homing resource allows for the POD module to request specific services from the Host when the Host is in a standby state. When the Host is in a standby state, only the "immediate" modes will be supported. This resource shall be modified to the following definition.

Table 8.13-A Homing Resource				
Resource	**Class**	**Type**	**Version**	**Identifier (hex)**
Homing	17	1	2	00110042

The POD module will open the Homing resource when it requires a firmware upgrade or requires a service. The creation of Homing resource session includes the following objects:

Table 8.13-B Homing Objects			
Apdu_tag	**Tag value (hex)**	**Resource**	**Direction Host <-> POD**
open_homing	9F9990	Homing	→
homing_cancelled	9F9991	Homing	→
open_homing_reply	9F9992	Homing	←
homing_active	9F9993	Homing	→
homing_complete	9F9994	Homing	←
firmware_upgrade	9F9995	Homing	←
firmware_upgrade_reply	9F9996	Homing	→
firmware_upgrade_complete	9F9997	Homing	←

8.13.3.2 open_homing (Normative)

The open_homing APDU is transmitted by the Host to the POD module when it enters the standby state, either from power up or from user action. It shall send this independent of whether the Host Control resource has a session active.

Table 8.13-C Open Homing Object Syntax		
Syntax	**# of bits**	**Mnemonic**

open_homing() {
open_homing_tag
length_field()
}
	24	uimsbf

8.13.3.3 open_homing_reply (Normative)

The open_homing_reply APDU is transmitted by the POD module to the Host to acknowledge receipt of the open_homing APDU.

Table 8.13-D Open Homing Reply Object Syntax		
Syntax	**# of bits**	**Mnemonic**

open_homing_reply() {
open_homing_reply_tag
length_field()
}
	24	uimsbf

8.13.3.4 homing_active (Normative)

The homing_active APDU is transmitted by the Host to the POD module to inform the POD module that the homing request has been activated.

Table 8.13-E Homing Active Object Syntax		
Syntax	**# of bits**	**Mnemonic**

homing_active() {
homing_active_tag
length_field()
}
	24	uimsbf

8.13.3.5 homing_cancelled (Normative)

If the Host was not informed that a firmware upgrade was in progress, then it shall have the capability to close the homing state.

Table 8.13-F Homing Cancelled Object Syntax		
Syntax	**# of bits**	**Mnemonic**

homing_cancelled() {
homing_cancelled_tag
length_field()
}
	24	uimsbf

8.13.3.6 homing_complete (Normative)

When the POD module no longer needs the homing function, then it can transmit a homing_complete to the Host.

Table 8.13-G Homing Complete Object Syntax		
Syntax	**# of bits**	**Mnemonic**

homing_complete() {
homing_complete_tag
length_field()
}
	24	uimsbf

8.13.3.7 firmware_upgrade (Normative)

If the POD module uses an in-band channel to perform a firmware upgrade, it shall transmit the firmware_upgrade APDU to the Host. If the upgrade_source is equal to the QAM inband channel (01), then the Host shall immediately give access to the inband tuner through the Host Control resource tune APDU. The Host shall not interrupt a firmware upgrade until it receives the firmware_upgrade_complete APDU. If the Host is not in the standby mode, then it shall display the user_notification_text. The user_notification_text shall be in ISO-8859-1. The estimated time to download in download_time shall be in seconds.

Table 8.13-H Firmware Upgrade Object Syntax		
Syntax	**# of bits**	**Mnemonic**
firmware_upgrade() {		
firmware_upgrade_tag	24	uimsbf
length_field()		
upgrade_source	8	uimsbf
download_time	16	uimsbf
timeout_type	8	uimsbf
download_timeout_period	16	uimsbf
text_length	8	uimsbf
for(i=0; i<text_length; i++) {		
user_notification_text	8	uimsbf
}		
}		

upgrade_source – This shall define which path the POD module will use for its firmware upgrade.

Table 8.13-I Upgrade Sources		
Value (hex)	**Source**	**Comment**
00	Unknown	POD is not informing Host of source
01	QAM Inband Channel	Host Control resource will be used.
02	QPSK OOB Channel	Host Control resource will be used.
03-FF	Reserved	

download_time – the amount of time, in seconds, that it estimated to take for the firmware upgrade. If the value is 0000, then the value is unknown.

timeout_type – the type of timeout requested.

Table 8.13-J Timeout Types		
Value (hex)	**Timeout Type**	**Comment**
00	Both timeouts	Use both 5 seconds and download_timeout_period
01	Transport timeout only	5 second timeout on transport layer
02	Download timeout only	Value in download_timeout_period
03	No timeout	Host will not timeout POD module
04-FF	Reserved	

download_timeout_period – the amount of time, in seconds, after the Host has received the firmware_upgrade APDU that the Host should use to determine that the POD module has become unstable. After this time, the Host should perform a PCMCIA reset on the POD module. The Host's timer should be reset every time a firmware_upgrade APDU is received. A value of 0000 is defined to be an infinite timeout period.

user_notification_text – the text to be displayed to the user if the Host is not in standby mode.

8.13.3.8 firmware_upgrade_reply (Normative)

The Host will reply to the firmware_upgrade APDU. The POD module will not start the download operation until it receives this reply.

Table 8.13-K Firmware Upgrade Reply Object Syntax		
Syntax	**# of bits**	**Mnemonic**

firmware_upgrade_reply() {
firmware_upgrade_reply_tag
length_field()
}
	24	uimsbf

8.13.3.9 firmware_upgrade_complete (Normative)

After the POD module has completed its upgrade, it will transmit the firmware_upgrade_complete APDU to the Host. Included in this is whether the POD module needs a PCMCIA reset (RESET signal active), POD reset (RS flag active), or no reset. If there is no reset, then the Host may take control of the tuner if the source was inband.

Table 8.13-L Firmware Upgrade Complete Object Syntax		
Syntax	**# of bits**	**Mnemonic**
Firmware_upgrade_complete() {		
firmware_upgrade_complete_tag	24	uimsbf
length_field()		
reset_request_status	8	uimsbf
}		

reset_request_status – This contains the status of the reset for the POD module.

Table 8.13-M Reset Request Status Values		
Value (hex)	**Source**	**Comment**
00	PCMCIA reset requested	Host will bring RESET signal active then inactive.
01	POD reset requested	Host will set RS flag and begin interface initialization.
02	No reset required	Normal operation continues
03-FF	Reserved	

Note that if the POD module wishes to cancel the firmware upgrade, it can send the firmware_upgrade_complete APDU with no reset requested. Normal operation should continue if the Host receives this APDU.

8.14 Generic Diagnostic Support

The ***Generic Diagnostic Support*** resource enables the POD to request that the Host perform a diagnostic and report the status/results of the request to the POD. The POD may then use the diagnostic information to report diagnostics to the headend or the OSD diagnostic application. If the POD attempts to open a diagnostic support session and the Host replies that generic diagnostic support is not available, then the POD shall not request any diagnostic information from the Host.

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

Table 8.14-A Generic Diagnostic Support Resource				
Resource	**Class**	**Type**	**Version**	**Identifier**
Generic Diagnostic Support	260	1	1	01040041

This creation includes the following objects:

Table 8.14-B Generic Diagnostic Support Objects			
Apdu_tag	**Tag Value (hex)**	**Resource**	**Direction Host ⇔ POD**
Diagnostic_req()	9FDF00	Generic Diagnostic Support	←
Diagnostic_cnf()	9FDF01	Generic Diagnostic Support	→

8.14.1 Diagnostic_req()

The POD's diagnostic application shall use the Diagnostic_req() object to request the Host perform a specific set of diagnostic functions and report the result/status of the diagnostics to the POD's diagnostic application.

Table 8.14-C Diagnostic Request Object Syntax		
Syntax	**# of bits**	**Mnemonic**
Diagnostic_req() {		
Diagnostic_req_tag	24	uimsbf
length_field()		
number_of_diag	8	uimsbf
for (i = 0; i < number_of_diag; i++){		
Diagnostic_id	8	uimsbf
}		
}		

number_of_diag This field indicates the total number of self-diagnostic being requested

Diagnostic_id This field is a unique ID assigned to a particular diagnostic. The following is a list of diagnostics and their assigned diagnostic ID.

Table 8.14-D Diagnostic ID Values	
Diagnostic ID	**Diagnostic**
00	Set-Top memory allocation
01	Software version
02	Firmware version
03	MAC address
04	FAT status
05	FDC status
06	Current Channel Report
07	1394 Port
08	DVI status
09-FF	Reserved for future use.

8.14.2 Diagnostic_cnf()

The Host's diagnostic application shall use the ***Diagnostic_cnf()*** object to respond to a POD's request to perform a specific set of diagnostic functions.

Table 8.14-E Diagnostic Confirm Object Syntax		
Syntax	**# of bits**	**Mnemonic**
Diagnostic_cnf() {		
Diagnostic_cnf_tag	24	uimsbf
length_field()		
number_of_diag	8	uimsbf
for (i = 0; i < number_of_diag; i++){		
Diagnostic_id	8	uimsbf
Status_field	8	uimsbf
if (status_field == 0x00) {		
if (Diagnostic_id == 0x00) {		
Memory_report()		
}		
else if (Diagnostic_id == 0x01) {		
Software_ver_report()		
}		
else if (Diagnostic_id == 0x02) {		
Firmware_ver_report()		
}		
else if (Diagnostic_id == 0x03) {		
MAC_address_report()		
}		
else if (Diagnostic_id == 0x04) {		
FAT_status_report()		
}		
else if (Diagnostic_id == 0x05) {		
FDC_status_report()		
}		
else if (Diagnostic_id == 0x06) {		
Current_channel_report()		
}		
else if (Diagnostic_id == 0x07) {		
1394_port_report()		
}		
else if (Diagnostic_id == 0x08) {		
DVI_status_report()		
}		
}		
}		
}		

number_of_diag This field indicates the total number of self-diagnostic contained in the confirmation
Diagnostic_id This field echoes back the unique ID assigned to a particular diagnostic request.
status_field Status of the requested diagnostic.

Table 8.14-F 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.

8.14.3 Diagnostic Report Definition

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

8.14.3.1 Memory Report

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

Table 8.14-G Memory Report Syntax		
Syntax	**# of bits**	**Mnemonic**
memory_report() {		
number_of_memory	8	Uimsbf
if (i=0; i<number_of_memory; i++) {		Uimsbf
memory_type	8	Uimsbf
memory_size	32	uimsbf
}		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.

Table 8.14-H Memory Type Values	
Bit Value (Hex)	**Memory Type**
00	ROM
01	DRAM
02	SRAM
03	FLASH
04	NVM
05	Hard Drive
06	Video Memory
07	Other Memory
08 – FF	Reserved for future use.

memory_size Designates the physical size of the specified memory type. The units are kilobytes, defined to be 1024 bytes.

8.14.3.2 Software Version Report

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

Table 8.14-I Software Version Report Syntax		
Syntax	**# 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
}		
}		
}		

number_of_applications Total number of applications contained within the report.
application_status_flag Status of the software, either active, inactive or downloading.

Table 8.14-J Software Status Flag Values	
Bit Value (Hex)	**Software Status Flag**
00	Active
01	Inactive
02	Downloading
03-FF	Reserved for future use.

application_name_length Designates the number of characters required to define the application name.
application_name_byte ASCII character, 8-bits per character, of string that identifies the application.
application_version_length Designates the number of characters required to define the application version.
application_version_byte ASCII character, 8-bits per character, of string that identifies the version.
application_sign_length Designates the number of characters required to define the application signature.
application_sign_byte ASCII character, 8-bits per character, of string that identifies the application signature.

8.14.3.3 Firmware Version Report

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

Table 8.14-K Firmware Version Report Syntax

Syntax	# of bits	Mnemonic
firmware_ver_report() {		
Firmware_version	16	uimsbf
Firmware_date {		
firmware_year	16	uimsbf
firmware_month	8	uimsbf
firmware_day	8	uimsbf
}		
}		

firmware_version_length Designates the number of characters required to define the firmware version.
firmware_version_byte ASCII character, 8-bits per character, of string that identifies the firmware version.
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.

8.14.3.4 MAC Address Report

MAC address report shall contain the MAC address parameters associated with the Host.

Table 8.14-L MAC Address Report Syntax		
Syntax	**# of bits**	**Mnemonic**
MAC_address_report() {		
number_of_addresses	8	uimsbf
for (i=0; i<number_of_addresses; i++) {		
MAC_address_type	8	uimsbf
number_of_bytes	8	uimsbf
for (j=0; j<number_of_bytes; j++) {		
MAC_address_byte	8	uimsbf
}		
}		
}		

number_of_addresses Total number of MAC addresses contained on report
MAC_address_type Type of device associated with reported MAC address

Table 8.14-M MAC Address Type Values	
Bit Value (hex)	**MAC Address Type**
00	No addressable device available
01	Host
02	1394
03	USB
04	DOCSIS
05	Ethernet
06 – FF	Reserved

number_of_bytes The total number of bytes required for the MAC address.
MAC_address_byte One of a number of bytes of 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.

8.14.3.5 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 8.14-N FAT Status Report Syntax		
Syntax	**# of bits**	**Mnemonic**
FAT_status_report() {		
reserved	4	bslbf
PCR_lock	1	bslbf
modulation_mode	2	bslbf
carrier_lock_status	1	bslbf
SNR	16	simsbf
signal_level	16	simsbf
}		

reserved Reserved bits shall be set to 1111_2.
PCR_lock Indicates if the FAT channel receiver is locked to the currently tuned channel,
0_2 = not locked,
1_2 = locked.
modulation_mode Indicates if current forward transport is analog, 64-QAM or 256-QAM,
00_2 = analog,
01_2 = 64-QAM,
10_2 = 256-QAM,
11_2 = Reserved
carrier_lock_status Indicates if the current carrier is lock or not locked,
0_2 = not locked,
1_2 = locked.
SNR Numerical representation of the signal to noise ratio in tenths of a dB.
signal_level Numerical representation of the signal level in tenths of a dBmV.

8.14.3.6 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 8.14-O FDC Status Report Syntax		
Syntax	**# of bits**	**Mnemonic**
FDC_report() {		
FDC_center_frq	16	uimsbf
Reserved	6	bslbf
carrier_lock_status	1	bslbf
Reserved	1	bslbf
}		

FDC_center_frq Indicates the frequency of the FDC center frequency, in MHz. (Frequency = value * 0.05 + 50 MHz)

Table 8.14-P FDC Center Frequency Value															
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Frequency (MS)								Frequency (LS)							

reserved Reserved bits shall be set to 111111_2.
carrier_lock_status Indicates if the current carrier is lock or not locked,
0_2 = not locked,
1_2 = locked.

8.14.3.7 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 8.14-Q Current Channel Report Syntax		
Syntax	**# of bits**	**Mnemonic**
current_channel_report() {		
Reserved	2	Bslbf
channel_type	1	Bslbf
authorizartion_flag	1	Bslbf
purchasable_flag	1	Bslbf
purchased_flag	1	Bslbf
preview_flag	1	Bslbf
parental_control_flag	1	Bslbf
current_channel	16	Uimsbf
}		

reserved Reserved bits shall be set to 11_2.
channel_type Indicates if channel is analog or digital,
0_2 = analog,
1_2 = digital
authorization_flag Indicates if the set-top terminal is authorized for the currently tuned channel,
0_2 = not authorized,
1_2 = authorized.
purchasable_flag Indicates if the currently tuned channel may be purchased,
0_2 = not purchasable, 1_2 = purchasable.

purchased_flag Indicates if the currently tuned channel has been purchased,
0_2 = not purchased,
1_2 = purchased.
preview_flag Indicates if the currently tuned channel is in preview mode,
0_2 = not in preview mode,
1_2 = in preview mode.
parental_control_flag Indicates if the currently tuned channel is under parental control,
0_2 = channel is not blocked,
1_2 = channel is blocked.
current_channel Indicates the numerical representation of the currently tuned channel.

8.14.3.8 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 8.14-R 1394 Report Syntax		
Syntax	**# of bits**	**Mnemonic**
1394_port_report() {		
Reserved	3	Bslbf
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
}		

reserved Reserved bits shall be set to 111_2.
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_2 = not root,
1_2 = is root.
cycle_master_status Indicates if the set-top terminal is the cycle master node on the 1394 bus,
0_2 = not cycle master,
1_2 = is cycle master.
port_1_connection_status Indicates if port 1 of the 1394 PHY is connected to a 1394 bus,
0_2 = not connected,
1_2 = connected.
port_2_connection_status Indicates if port 2 of the 1394 PHY is connected to a 1394 bus,
0_2 = not connected,
1_2 = connected.
total_number_of_nodes Indicates the total number of nodes connected to the 1394 bus. A maximum of 65535 nodes may exist, excluding the Host (a maximum of 64 nodes per bus with a maximum of 1024).

8.14.3.9 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 8.14-S 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
frame_rate	8	uimsbf
aspect_ratio	2	bslbf
prog_inter_type	1	bslbf
reserved	5	
}		
}		

reserved All reserved bits shall be set to 1_2

connection_status Indicates if a connection exists on the DVI port:
00_2 = no connection exists,
01_2 = device connected – not repeater,
10_2 = device connected - repeater ,
11_2 = reserved

host_HDCP_status Indicates if HDCP is enabled on the DVI link,
0_2 = not enabled,
1_2 = enabled.

Device_HDCP_status Indicates the connected device's HDCP status (valid only when connection status is not equal to 00_2)
00_2 = non HDCP device,
01_2 = compliant HDCP device
10_2 = revoked HDCP device,
11_2 = 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.

frame_rate Indicates the frame rate associated with the video format on the DVI link as follows:

0x01=23.976 Hz
0x02=24 Hz
0x03=reserved
0x04=29.97 Hz
0x05=30 Hz
0x06=reserved
0x07=59.94 Hz
0x08=60 Hz.
0x09-0xFF=reserved

aspect_ratio Indicates the aspect ratio associated with the video format on the DVI link as follows:

00_2 = 4:3
01_2 = 16:9
10_2 = reserved
11_2 = reserved

prog_inter_type Indicates if the video is either progressive or interlaced on the DVI link,
0_2 = Interlaced,
1_2 = Progressive

8.14.3.10 HDMI Port Status Report

In response to a HDMI Status Report request, the Host shall reply with a HDMI Status Report, unless an error has occurred.

Table 8.14-T HDMI Status Report Syntax

Syntax	# of bits	Mnemonic
HDMI_status_report() {		
Rreserved	3	'111'
connection_status	2	Bslbf
host_HDMI_status	1	Bslbf
device_HDCP_status	2	Bslbf
video_format		
{		
horizontal_lines	16	Uimsbf
vertical_lines	16	Uimsbf
frame_rate	8	Uimsbf
aspect_ratio	2	Bslbf
prog_inter_type	1	Bslbf
Reserved	5	Bslbf
}		
}		

connection_status Indicates if a connection exists on the HDMI port

- 00_2 No connection exists
- 01_2 Device connected, no repeater
- 10_2 Device connected, with repeater
- 11_2 Reserved

host_HDMI_status Indicates if HDCP is enabled on the HDMI link

- 0_2 Not enabled
- 1_2 Enabled.

device_HDCP_status Indicates the connected device's HDCP status (valid only when connection_status is not equal to 00_2)

- 00_2 Non HDCP device
- 01_2 Compliant HDCP device
- 10_2 Revoked HDCP device
- 11_2 Reserved

video_format Indicates the current video format utilized on the HDMI port as defined in the following fields:

horizontal_lines Indicates the number of horizontal lines associated with the video format on the HDMI link.

vertical_lines Indicates the number of vertical lines associated with the video format on the HDMI link.

frame_rate Indicates the frame rate associated with the video format on the HDMI link as follows.

0x01 23.976 Hz
0x02 24 Hz
0x03 reserved
0x04 29.97 Hz
0x05 30 Hz
0x06 reserved
0x07 59.94 Hz
0x08 60 Hz
0x09-0xFF reserved

aspect_ratio Indicates the aspect ratio associated with the video format on the HDMI link as follows:

00_2 4:3
01_2 16:9
10_2 Reserved
11_2 Reserved

prog_inter_type Indicates if the video is progressive or interlaced on the HDMI link,

0_2 Interlaced
1_2 Progressive

8.15 Support for Common Download Specification

This section specifies a common download protocol for POD-Host Interface for Host devices with OOB data channels as well as the In-Band (IB) Forward Application Transport (FAT) channel, consistent with the OpenCable Common Download Specification.

8.15.1 Overview of Protocol (Informative)

The protocol described in this document is based on the DSM-CC data carousel, which provides a format for data objects on a broadcast carousel. Since a common transport layer protocol for the in-band data channel (also known as the Forward Application Transport or FAT Channel) is MPEG-2, it provides a convenient starting point for a common protocol to download operating software code objects.

Control messages, specific to each type of Host device on the network, provide a locator (frequency of the transport stream, modulation mode, and PID) for the code file on the data carousel. A second approach is also defined where a source ID is utilized that identifies the program source associated with the virtual channel that is utilized for carrying the DSM-CC Download Information Indication message and/or the code file. The DSM-CC Download Information Indication message contains information pertaining to various firmware images that are available for download for particular Host devices.

This document uses the following terminology to differentiate between the two download methods:

1) **OOB Forward Data Channel method**: This method places the Code Version Table (CVT), as defined in section 3.5.5, in the OOB Forward Data Channel. The POD acquires the CVT via the OOB, filters the CVT and passes relevant information to the Host, as defined within this document. The Host utilizes the information passed to it via the POD to determine if a download is available. Download via this method is not possible without the POD. The data carousel is carried on the IB and contains the code file image. The Host only knows of the existence of a download via the POD.

2) **IB FAT Channel method**: This method places the Code Version Download Table (CVDT), as defined in section 8.15.2.2.1, in the IB FAT Channel. The Host acquires the CVDT via the IB, filters the CVDT and determines if a download available. The POD is not used for filtering or reception of the table. The data carousel is carried on the IB and contains both the code file image and the CVDT. The Host knows of the existence of a download via the utilization of a source ID or via interaction with the POD.

Currently, the transport and message protocols between the Headend and the POD are proprietary. In order for any Host to decode the message, extensions to the existing POD-Host interface specification, are required. These extensions provide a common network interface to the Host. This approach requires some additional functionality in the POD, in that the POD translates the proprietary network protocols to the common one specified in this document.

There are four types of code upgrade protocols that may be used by the MSO to download code. Additionally, there may be devices on the network for which the MSO does not have code objects, in which case, download would not be supported. Summarizing these options:

- In-band, broadcast
- In-band, command
- In-band, on-demand
- DOCSIS
- Not supported

This specification defines protocols for the first three download methods; the fourth method is well documented in the DOCSIS specification. These methods specified here are based on the DSM-CC data carousel. The broadcast model is used when the MSO is not dynamically adding files to the broadcast carousel. The command model

is used when the MSO has just added a new code file and wants all applicable Host devices to download the code file immediately. The on-demand case allows the MSO to dynamically add and remove code images as new subscribers come online and require upgrades.

8.15.1.1 Common Download via the OOB Forward Data Channel

In the broadcast model, data is broadcast over the OOB Forward Data Channel that relates the manufacturer and hardware version to the locator for the code object in a DSM-CC data carousel. The POD filters these data and passes the appropriate data onto the Host. The Host can then tune to the appropriate broadcast MPEG multiplex stream and set the PID filters to the PID that identifies the code object in the multiplex stream.

Because every possible code object might not be carried all of the time on the broadcast carousel, the MSO may provide an on-demand capability. In this method, when a Host signs onto the network, the Headend is informed that a new Host is now on-line. If a new version of the software for that Host is available, the Headend loads the object onto the carousel and sends a message back to the Host identifying the location of this code object. If a code object is not available, the Host is informed that download is unsupported. After the Host has finished downloading the object and authenticates it, it sends a 'done' message so that the Headend can unload that object from the carousel.

8.15.1.2 Common Download via the IB Forward Application Transport Channel

The IB FAT common download utilizes a combination of a source ID that identifies the program source associated with the virtual channel that is utilized for carrying the DSM-CC Download Information Indication message and/or the code file. In this scenario the cable plant:

- places the DSM-CC Download Information Indication message and/or the code file on a multiplex,
- assigns the source ID, to the virtual channel that the DSM-CC Download Information Indication message and/or the code file will be mapped to,
- and if the multiplex is encrypted, authorizes the Host for that service.

The Download Information Indication message contains the data necessary for the Host device to determine if the code file is targeted for that particular device. If the code file is applicable to the device, then the Host device downloads the code file. If the code file is not applicable or no code file is present on a different MPEG multiplex, as defined by the CVDT, then the Host device terminates the common download process.

The Host device parses the contents of the DSM-CC Download Information Indication message in order to determine if a download exists for the device. There

are several methods in which the host will acquire the DSM-CC Download Information Indication message:

1. The Host device monitors the Version Number of the VCT. If the Version Number of the VCT changes, then the Host parses the VCT to see if the source ID is present. If the source ID is present, then when the Host is in a state that will not interrupt the user's service, the Host tunes to the channel defined by the source ID and parses the DSM-CC Download Information Indication message. If the DSM-CC Download Information Indication message indicates that a download is available, then the Host downloads the firmware code file.

2. The headend informs the POD that the DSM-CC Download Information Indication message has been modified and likewise the POD informs the Host that the DSM-CC Download Information Indication message has been modified. The Host parses the VCT to see if the source ID is present. If the source ID is present, then when the Host is in a state that will not interrupt the user's service, the Host tunes to the channel defined by the source ID and parses the DSM-CC Download Information Indication message. If the DSM-CC Download Information Indication message indicates that a download is available, then the Host downloads the firmware code file.

3. The Host device periodically parses the VCT and searches for the applicable source ID. If the source ID is present, then when the Host is in a state that will not interrupt the user's service, the Host tunes to the channel defined by the source ID and parses the DSM-CC Download Information Indication message. If the DSM-CC Download Information Indication message indicates that a download is available, then the Host downloads the firmware code file.

4. The headend commands the Host, via the POD, to tune to the channel defined by the given source ID or tune to the multiplex defined via the provided frequency, modulation type and mpeg number or PID, and parse the DSM-CC Download Information Indication message. If the DSM-CC Download Information Indication message indicates that a download is available, then the Host downloads the firmware code file**.**

After acquiring the Download Information Indication message, the Host device parses the Download Information Indication message and extracts the Code Version Download Table (CVDT), which is defined in this document. The CVDT is contained in the Private Data sector of the Download Information Indication message. The CVDT indicates the vendor ID and hardware version ID of all Host devices that have code files available for download. The CVDT also indicates the source ID associated with the multiplex and/or the frequency, modulation type and MPEG number of the stream that contains the applicable DSM-CC data carousel. The MSO may opt to place the applicable download object within the same stream that contains the CVDT; this would be indicated by the source ID being equal to the previously defined source ID.

It is the responsibility of the Host device manufacturer to ensure that the device has the means to verify that the code file is valid.

8.15.2 OPERATIONAL DETAILS (Informative)

8.15.2.1 Download Protocols

8.15.2.1.1 In-band, one-way, broadcast

No interaction with the Headend is supported for this method. This method utilizes a DSM-CC data carousel. If a code file for a particular host is not defined in the CVDT, then code download for that host is not supported.

For the In-Band method that utilizes on the Download Info Indication message, the DSM-CC data carousel is placed on a multiplex that is defined by a source IDThe Host device parses the VCT for this source ID. If the source ID is found, then the Host device acquires the DSM-CC data carousel, extracts the Download Info Indication message and determines if a download is available. If a download is available, then the Host device downloads the code object.

8.15.2.1.2 In band, two-way, command

This method is applicable only for the case where the CVDT is delivered via the Download Info Indication message. This method uses interaction with the Headend and also uses the DSM-CC data carousel as defined in Section 8.15.2.1.1. After the Headend receives a new code file for a Host and/or a set of Hosts that requires an immediate download, the Headend loads the new code file onto the in-band broadcast carousel and modifies the Download Info Indication message accordingly. The Headend then instructs the POD to command the Host to download the code file. Headend knowledge of the existence of the device is made by prior POD-Host interaction and subsequent POD-headend interaction.

8.15.2.1.3 In-band, two-way, on-demand

This method requires interaction with the Headend and also utilizes the DSM-CC data carousel. When a Host receives a CVT and determines that an image download is required, it will either

a) always notify the POD module to notify the headend, or

b) it will look at the DSM-CC carousel location in the CVT. If it does not detect its image file, it will notify the POD module to notify the Headend. After the Headend is notified that a particular Host has requested a download, the Headend loads the appropriate code file onto the in-band broadcast carousel.

If no code file is available for a particular host, then code download for that host is not supported.

8.15.2.1.4 DOCSIS

This method uses a two-way connection through a DOCSIS cable modem and utilizes the Trivial File Transport Protocol (TFTP) method used by DOCSIS for its operational software. For Advanced Terminal devices (with embedded DOCSIS), in DOCSIS enabled networks, this mechanism can, at the discretion of the operator, be used to upgrade the operating software. Further elaboration of this method is beyond the scope of this specification.

8.15.2.2 DSM-CC Data Carousel

All software objects are transported over the in-band, broadcast channel via the DSM-CC data carousel. The Download Information Indication message, as defined in Section 8.15.2.2.1 of this document, and the message sequence for data carousel scenario, as defined in section 7.5 of ISO/IEC 13818-6, Extensions for DSM-CC, are supported. The DSM-CC specification does not require the DSM-CC control messages. The Host-POD control messages are defined in this specification.

8.15.2.2.1 Download Info Indication Message

The Download Info Indication message is only utilized when an MSO is broadcasting a download utilizing the source ID method for download as defined in section 8.15.1.2. The DSM-CC Download Info Indication message is extended to include a Code Version Download Table (CVDT). The CVDT is placed in the Private Data bytes of the Download Info Indication message. The format of the CVDT is defined in the following table.

Table 8.15-A Code Version Download Table		
Syntax	**# of bits**	**Mnemonic**
code_version_download_table() {		
code_version_download_table_tag	24	uimsbf
length_field()		
number_of_entries	8	uimsbf
For(i=0; i<number_of_entries; i++){		
vendor_id	24	uimsbf
hardware_version_id	32	uimsbf
download_type	4	uimsbf
download_command	4	uimsbf
location_type	8	uimsbf
if(location_type == 0){		
Source_id	16	uimsbf
}		
else{		
Frequency_vector	16	uimsbf
transport_value	8	uimsbf
Stream_ID	8	uimsbf
if(stream_ID == 0){		
reserved	3	uimsbf
PID	13	uimsbf
}		
Else{		
Program_number	16	uimsbf
}		
}		
code_file_name_length	8	uimsbf
For(i=0;I<code_file_name_length;i++){		
code_file_name_byte	8	uimsbf
}		
code_verification_certificate()		
}		
}		

code_version_download_tag Value = 0xFF0FFF

number_of_entries Defines the total number of entries that are contained within the CVDT.

vendor_ID Organizationally Unique Identifier (OUI) assigned to the Host device vendor by the IEEE. A value of 0x000000 is not valid.

hardware_version_id Unique Hardware identifier assigned to each type of hardware from a particular vendor. A value of 0x0000 is not permitted.

download_type Defines the type of download.
00 Broadcast
01 Command
02 On Demand
03 DOCSIS TFTP
04 Download unsupported – no code object available. In this case, the remaining fields will have no meaning and must be set to all zeros.

download_command
00 Download now
01 Deferred Download
02 Download now, no exceptions
03-0F Reserved

location_type Defines the method in which the Host device is to utilize to acquire the DSM-CC stream. A value of zero (0) indicates that the stream location is defined in the channel map and may be found via the defined source ID. A value other than zero indicates that the Host device is to use the frequency, modulation type and PID or program number to acquire the DSM-CC stream.

source_id The VCT source ID that is associated with each program source. The source ID is utilized to locate the frequency that the DSC-CC data carousel is multiplexed on.

frequency_vector Frequency of the download carousel. The frequency is coded as the number of 0.25 MHz intervals. If download_type parameter equals 03, this parameter must be set to 0.

transport_value	Value Type
00	DOCSIS channel
01	FAT channel/QAM-64
02	FAT channel/QAM-256
03-FF	Reserved

stream_ID Defines the way in which the DSM-CC is to be located. A value of zero (0) indicates that the DSM-CC stream is to be located utilizing the defined PID.

Any other value indicates that the stream is to be located utilizing the program number.

PID Packet identifier of the stream that contains the code file.

program_number Defines the program number in which the DSM-CC stream resides.

code_file_name_length Length of the code file name

code_file_name_byte Name of the software upgrade file on the DSM-CC carousel. This is the name of the Code File [3] that is on the data carousel as well as in Host Flash.

code_verification_certificate Authentication certificate(s) per DOCSIS SCTE 23-2 2002

8.15.2.3 Download Operation

The download method used by the operator is optional and in part depends upon the capabilities of the network and the Hosts on the network.

8.15.2.3.1 One-Way Operation- OOB Forward Data Channel

The following figure describes the communication between the Headend and the POD module and the POD module and the Host.

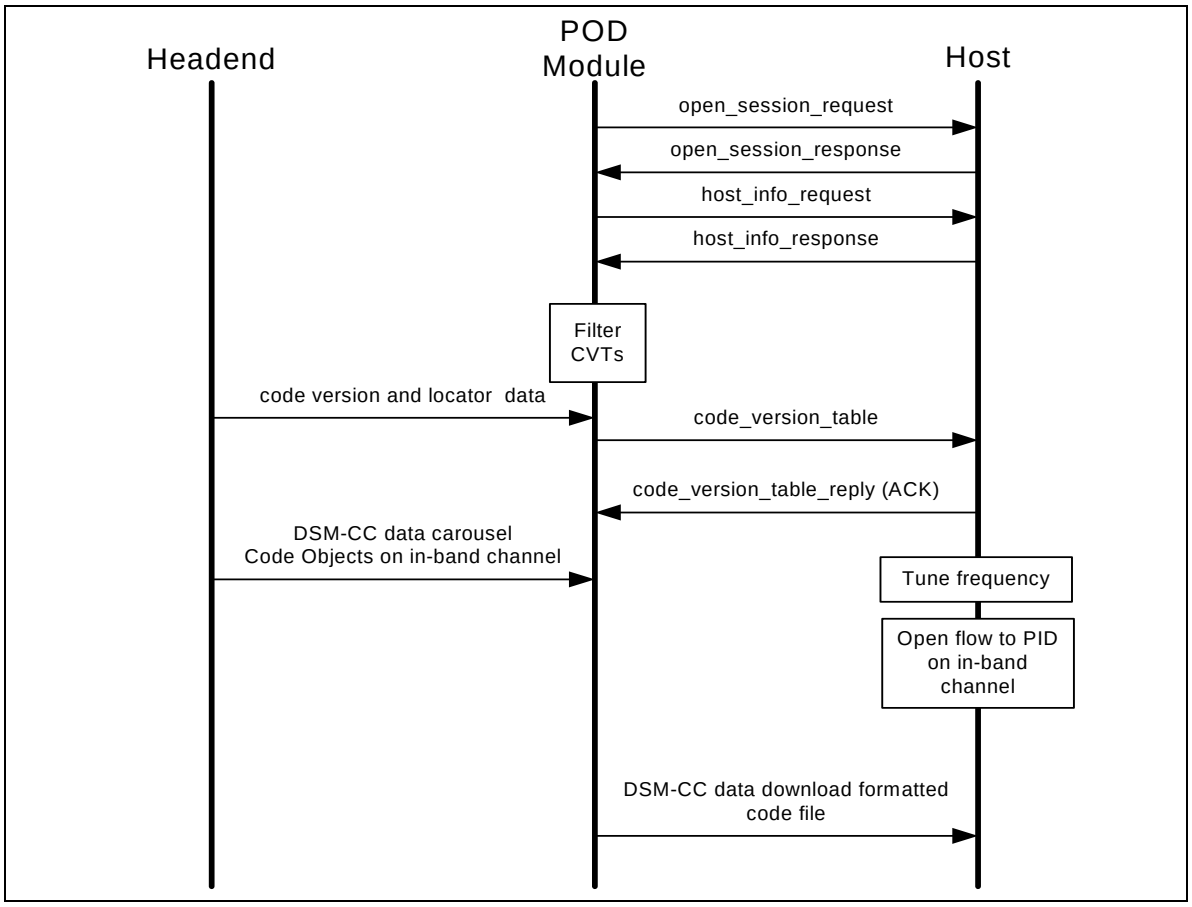

Figure 8.15-1 One-Way Operation

After a session is opened between the POD and the Host, the POD requests identification information from the Host. The Host responds with the vendor_id (OUI), hardware_version_id from the Host (host_info_response). The POD module uses this data to filter the Code Version Tables (CVT) that are broadcast over the OOB channel. Each CVT corresponds to a different hardware and software version. The locator is contained within the CVT and is the frequency of the transport stream and the PID for the specific data carousel of the code file. The POD transmits the proper CVT to the host. The Host determines if a download is required by comparing the code file name in the CVT to that store in the Host. If a download is required, the Host begins to download the code object by tuning to the proper in-band frequency and selecting the proper PID in the in-band multiplex stream.

8.15.2.3.2 One-Way Operation - IB Forward Application Transport Channel

The following figure describes the communication between the Headend and the POD module and the POD module and the Host.

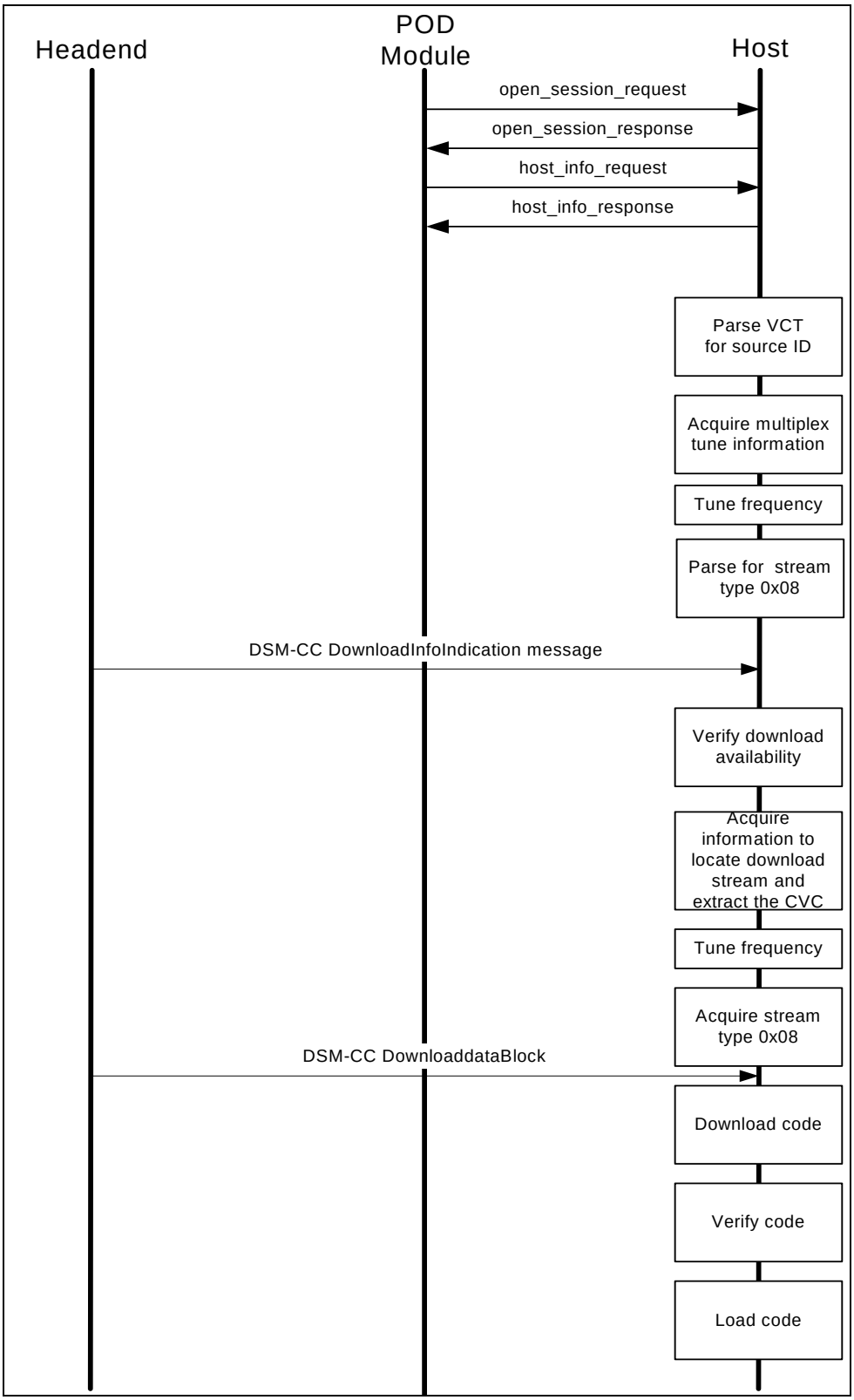

Figure 8.15-2 One-Way Operation – IB FAT Channel

8.15.2.3.3 Two-way Operation- OOB Forward Data Channel

The two-way operation is similar to the one-way operation, except that there is some handshaking between the POD and the Headend. After the Host receives a CVT and determines that a new code image exists, it is told in the CVT to either notify the POD module to inform the Headend of this hardware or to first determine if its code image is already loaded on the DSM-CC carousel and only notify the POD module to inform the Headend of the hardware if it is not. If download is not supported, the default CVT is sent ("Download Unsupported", download_type equals 03, see section 8.15.3.5).

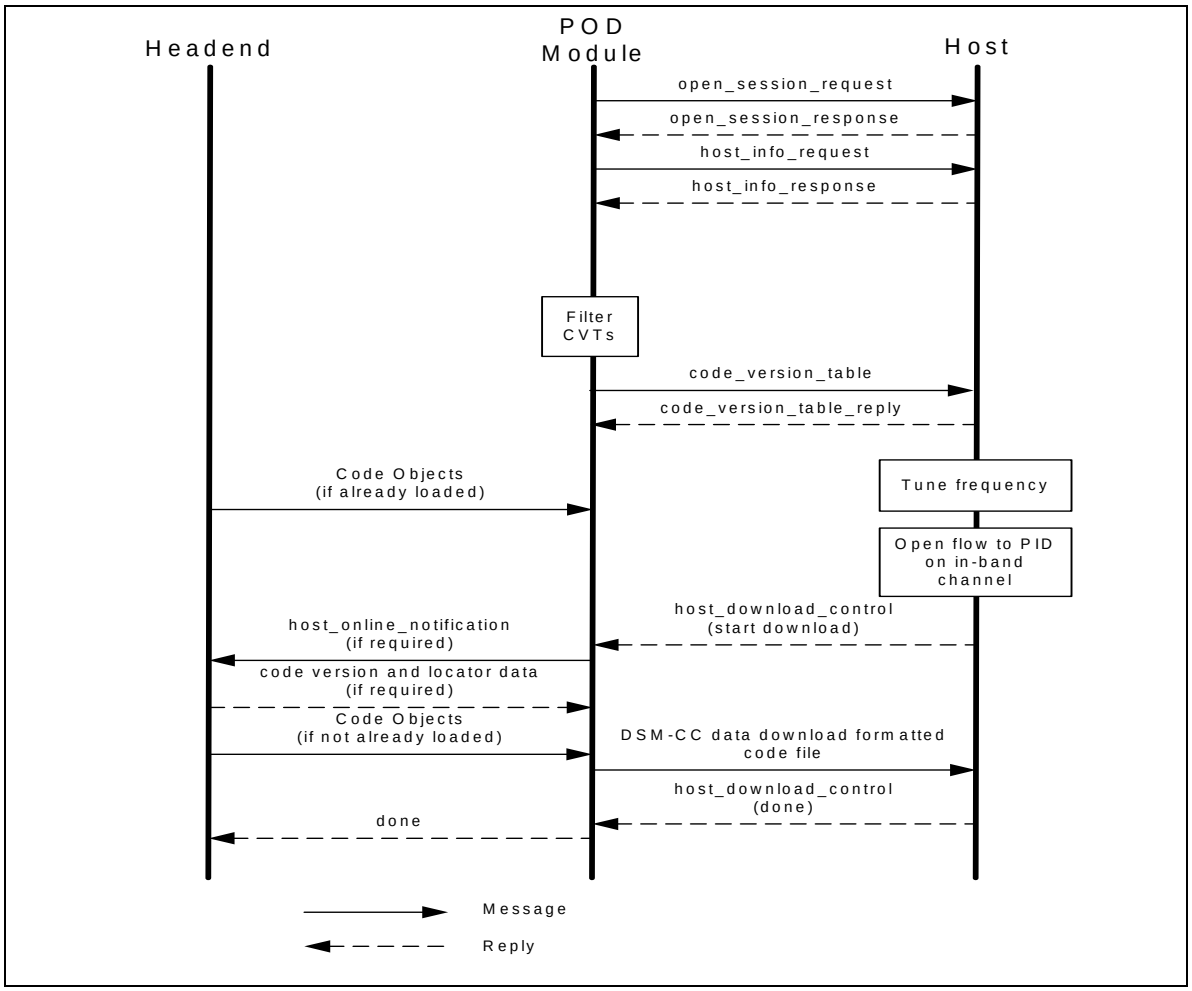

Figure 8.15-3 Two-Way Operation

8.15.2.3.4 Two-way Operation - Command Operation - IB Forward Application Transport Channel

The following figures describe the communication between the Headend and the Host for IB Forward Application Transport Channel command downloads.

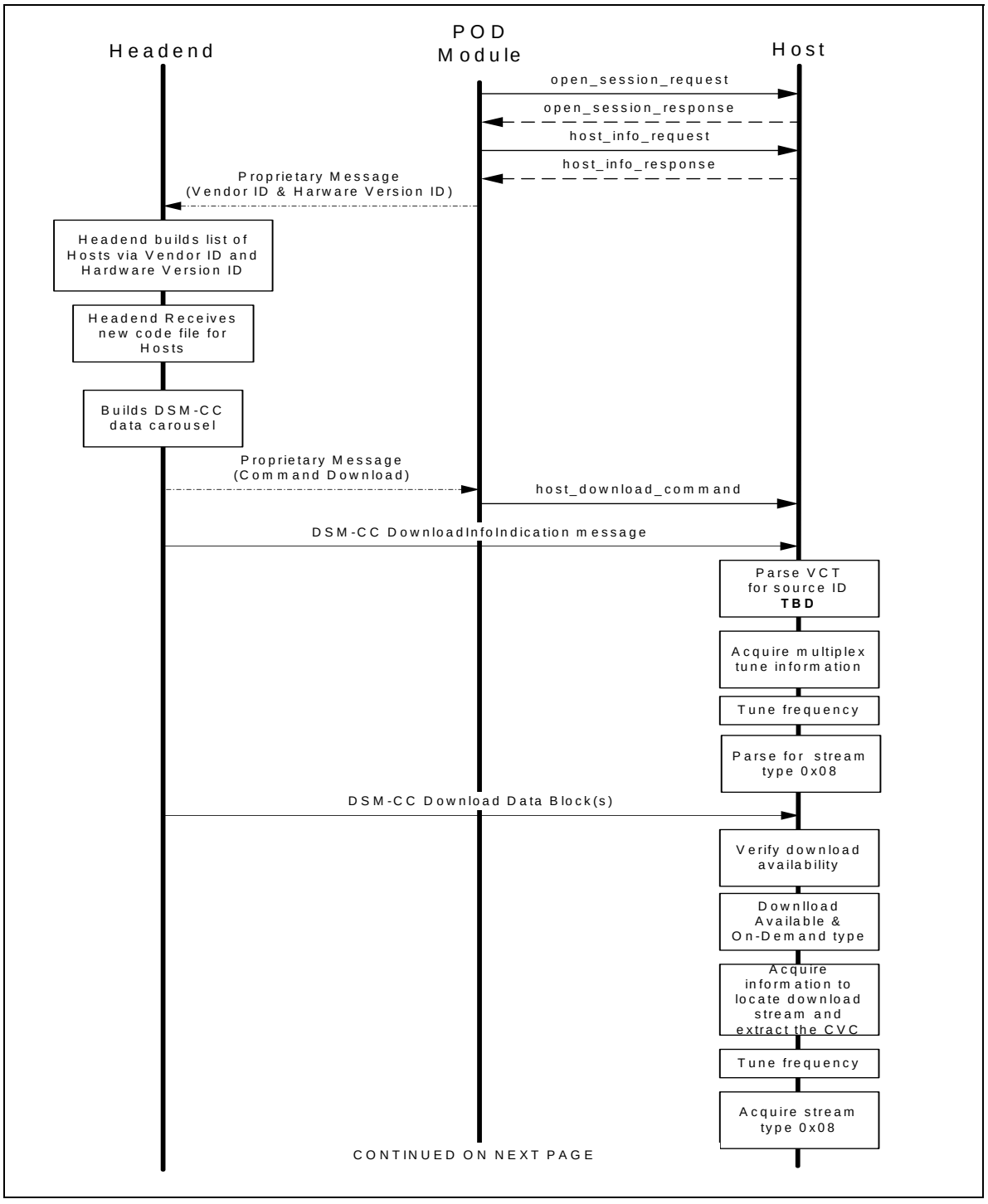

Figure 8.15-4 Two Way - Command Operation - IB FAT Channel

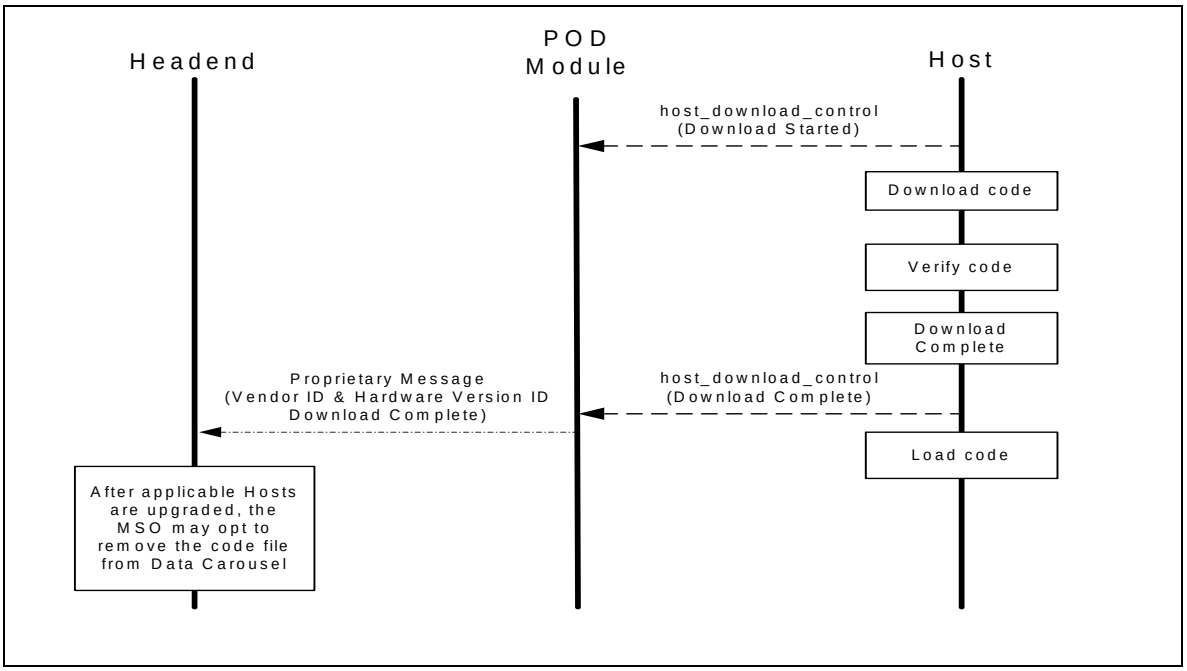

Figure 8.15-5 Two Way - Command Operation - IB FAT Channel (continued)

8.15.2.3.5 Two-way Operation - On-Demand Operation - IB Forward Application Transport Channel

The following figure describes the communication between the Headend and the Host for on-demand downloads.

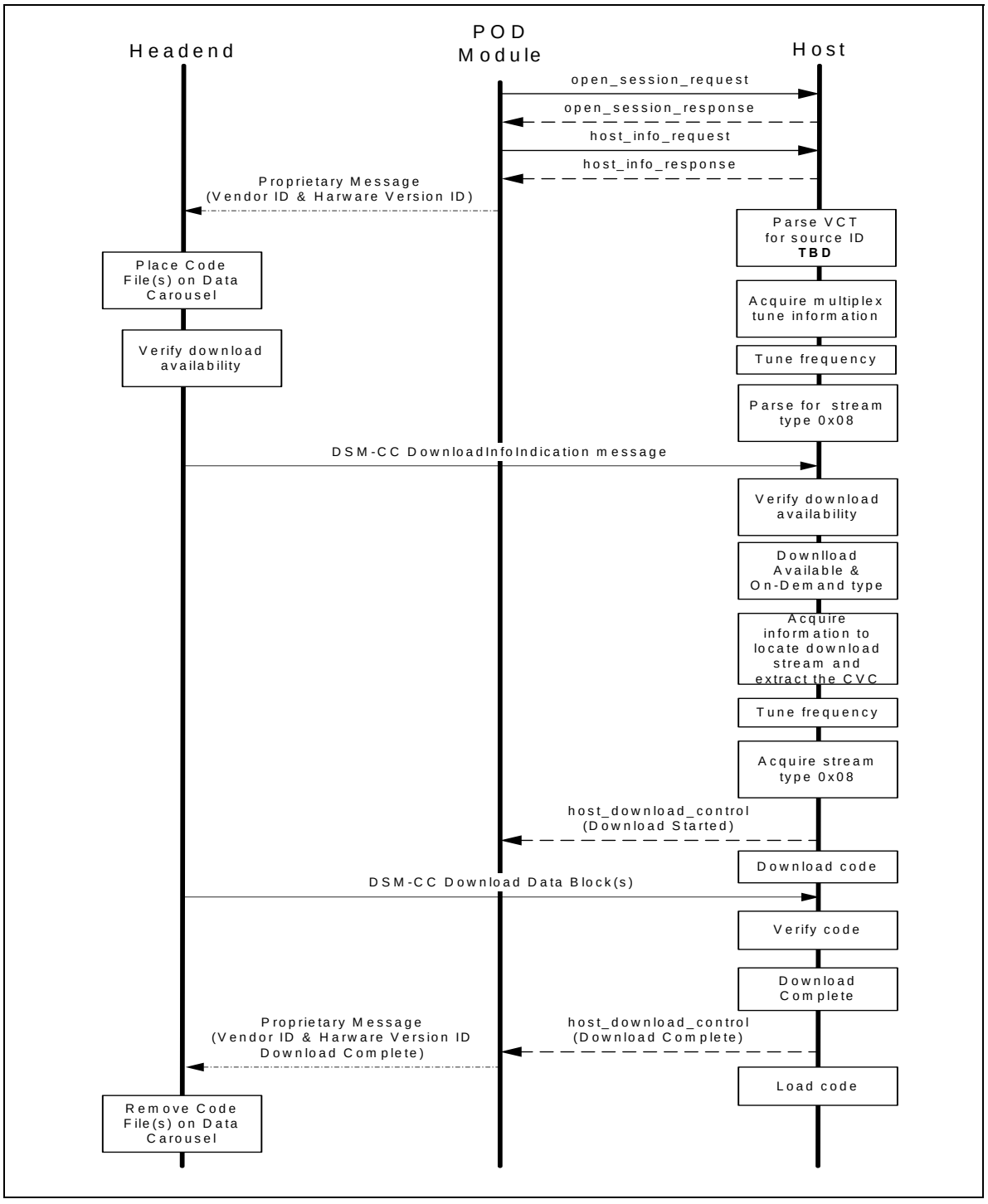

Figure 8.15-6 Two Way – On-Demand Operation - IB FAT Channel (continued)

8.15.2.4 Summary

The following figure summarizes the flow of events and the decision points in the download operation.

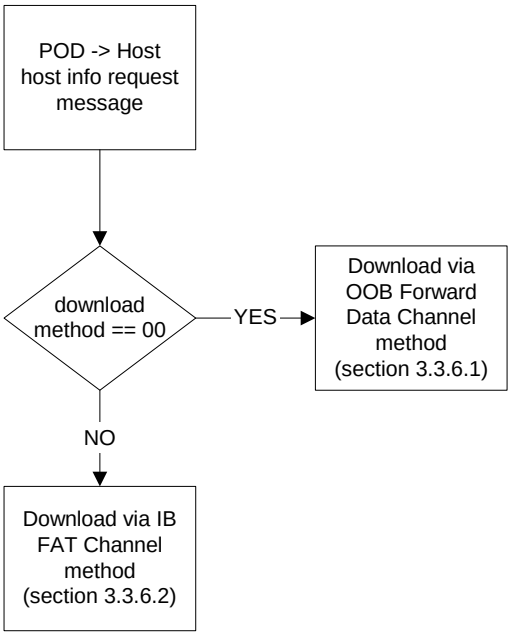

Figure 8.15-7 Flow chart summarizing download operations

8.15.2.4.1 OOB Forward Data Channel Summary

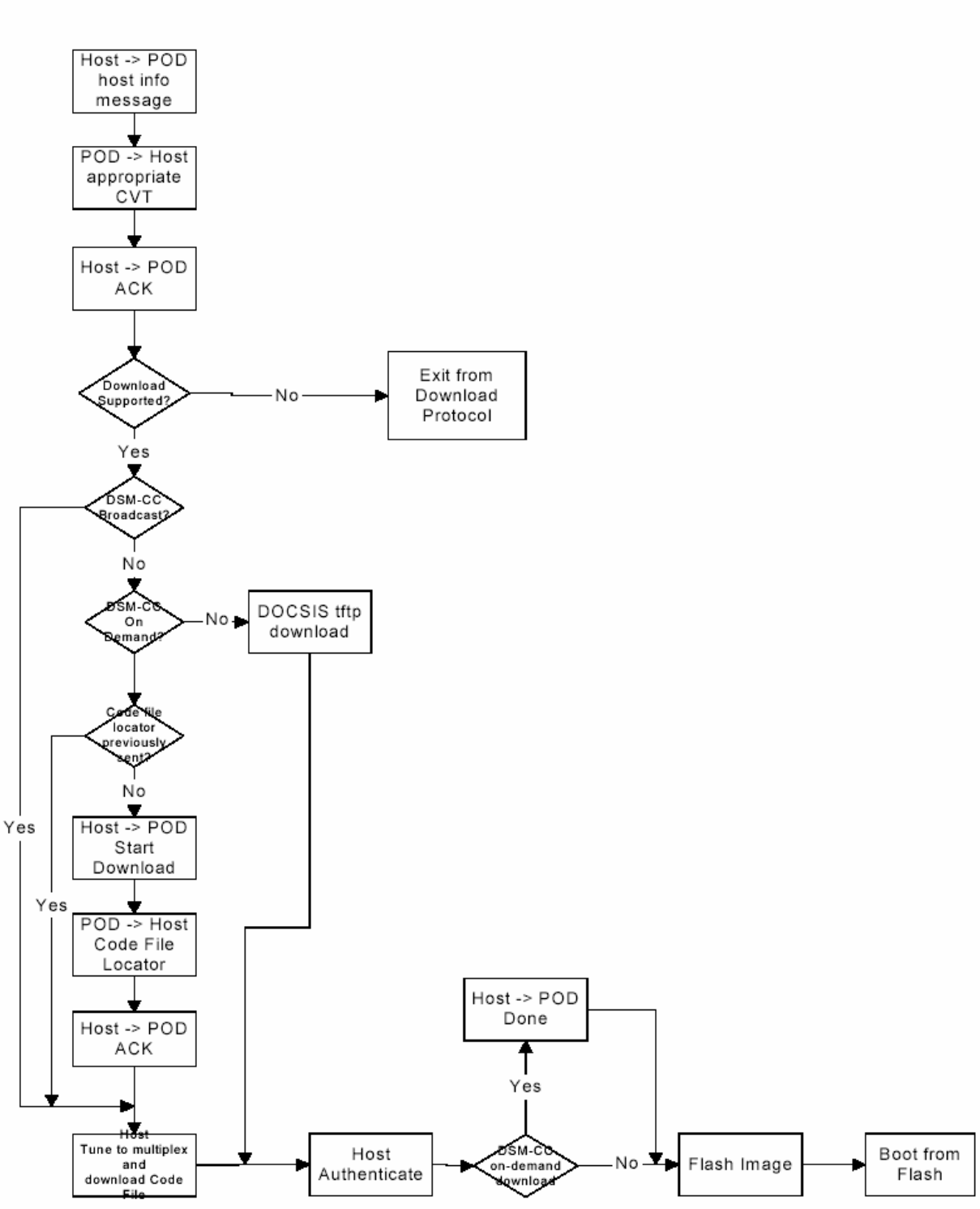

Figure 8.15-8 Flow chart summarizing download operations for OOB Forward Data Channel method

8.15.2.4.2 IB Forward Application Transport Channel Summary

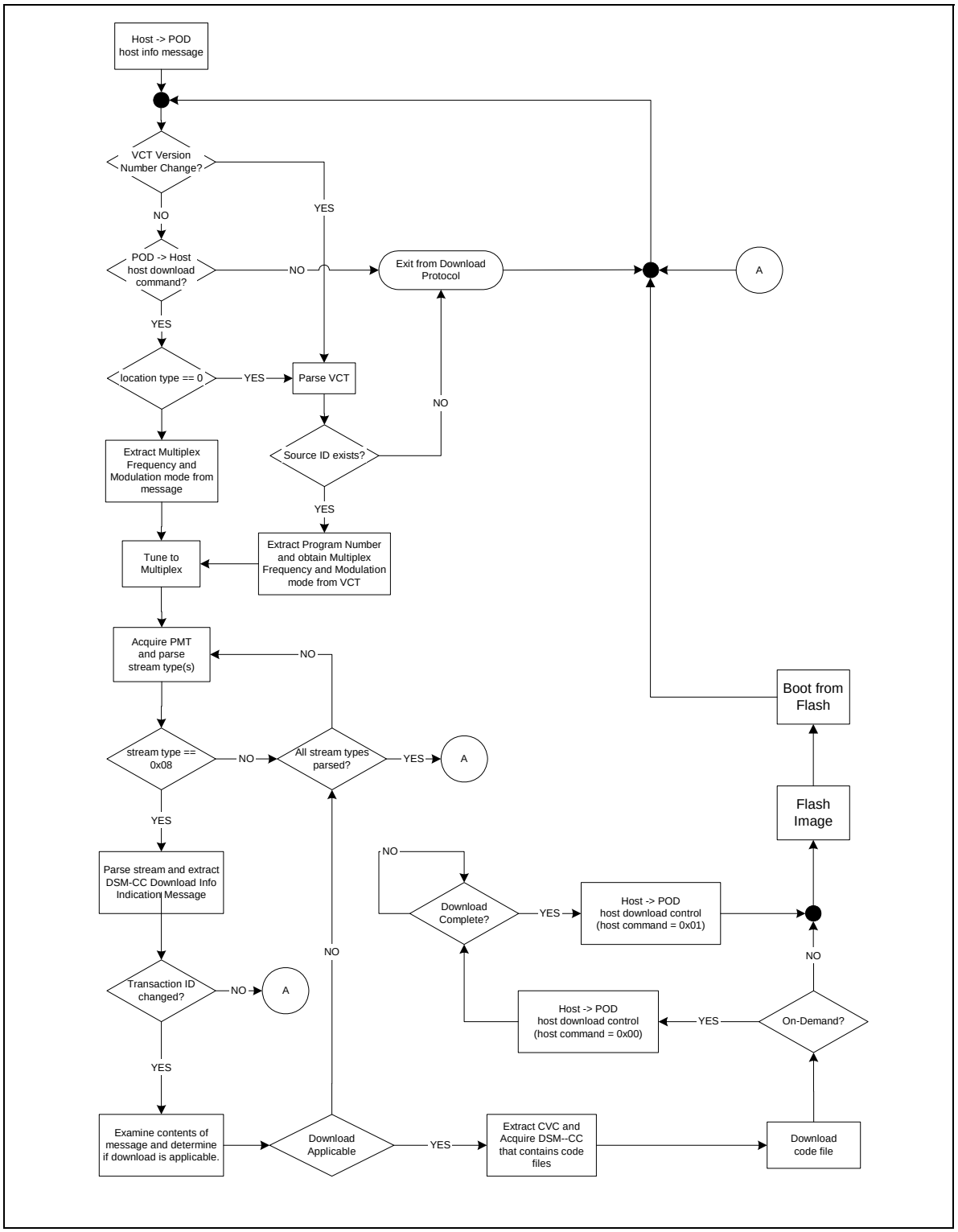

Figure 8.15-9 Flow chart summarizing broadcast download operations

8.15.2.5 Code Authentication

After a code image is downloaded into the set top box and before it is placed in permanent storage in non-volatile memory, the image is authenticated using the SCTE 23-2 2002 code authentication process regardless of the method used to download the file. This method specifies a particular structure to the code file (PKCS#7 compliant). The code file consists of the manufacturer's Code Verification Signature (CVS), and X.509 Code Verification Certificate (CVC) signed by the root CA, and the signed code image that is compatible with the target.

8.15.3 System Control Resource (Normative)

This section provides details of Host-POD messages. A new resource, the System Control resource, is introduced for handling revision control and download operations. Applications must exist in the POD to support this resource. New Application Protocol Data Units (APDU) are also introduced.

8.15.3.1 Resource Identifier

The following resource identifier associated with the System Control resource shall reside in the Host and is optional for use by the POD module. The POD shall open a session to this resource in the Host and shall never close it. Only one session is supported by the Host.

Table 8.15-B Resource Identifier

Resource	Class	Type	Version	Identifier
System Control	43	1	1	0x002B0041*

*proposed value

8.15.3.2 Application Objects (APDUs)

The following table is a list of the APDUs that are required for this specification. The host_info_request, host_info_response and the host_download_control APDUs are required for both download methods. The code_version_table and code_version_table_reply APDUs are exclusively utilized for the OOB Forward Data Channel download method. The host_download_command APDU is exclusively utilized for the IB FAT Channel download method.

Table 8.15-C Table of Application Protocol Data Units			
APDU_tag	**Tag value (hex)**	**Resource**	**Direction Host <-> POD**
Host_info_request	9F9C00	System Control	←
Host_info_response	9F9C01	System Control	→
Code_version_table	9F9C02	System Control	←
Code_version_table_reply	9F9C03	System Control	→
Host_download_control	9F9C04	System Control	→
Host_download_command	9F9C05	System Control	←

8.15.3.3 host_info_request

After the POD module opens a session to the System Control resource, the POD module shall query the Host to determine its vendor ID and hardware version ID and optional additional parameters. The POD also must inform the Host as to what type of download the Headend is going to use to update the Host. The POD module shall use at least the vendor ID and hardware version ID to filter the CVT. If a download is in progress, the Host shall terminate it.

Table 8.15-D host_info_request		
Syntax	**# of bits**	**Mnemonic**
host_info_request() {		
host_info_request_tag	24	Uimsbf
length_field()		
supported_download_type	8	Uimsbf
}		

host_info_request_tag Value = 0x9F9C00

supported_download_type Defines the type of Common Download method utilized by the Headend.

00 OOB Forward Data Channel method
01 IB FAT Channel method
02 DOCSIS only
03 – FF Reserved

Note: this document does not define DOCSIS download requirements. A headend need not use this ADPU to inform the Host that updates are performed via a DOCSIS download.

8.15.3.4 host_info_response

The Host shall respond to the POD module query with its vendor ID and hardware version ID.

Table 8.15-E host_info_response

Syntax	# of bits	Mnemonic
Host_info_response() {		
Host_info_response_tag	24	Uimsbf
Length_field()		
Vendor_id	24	Uimsbf
Hardware_version_id	32	Usmsbf
Number_of_descriptors	8	Uimsbf
for(I=0;i<number_of_descriptors;i++){		
Descriptor_tag	8	Uimsbf
Descriptor_len	8	Usmsbf
Descriptor_data()		
}		
}		

host_info_response_tag Value = 0x9F9C01

vendor_id Organizationally Unique Identifier (OUI) assigned to the Host device vendor by the IEEE. A value of 0x000000 is not valid.

hardware_version_id Unique Hardware identifier assigned to each type of hardware from a particular vendor.

number_of_descriptors Indicates the number of descriptors defined in the following fields

descriptor_tag
0 – descriptor_data is Host proprietary data. The maximum value for descriptor_len is 128.

1-127 – reserved for future standardization
128-255 – optional, for use by POD-Host pairs, where both POD and Host support the same implementation of the Specific Application Resource. Other POD-Host pairs shall skip these descriptors using descriptor_len value.

8.15.3.5 code_version_table

The Headend broadcasts all CVTs via the OOB-FDC. After the POD receives the host_info_response message from the Host, the POD module shall only then start filtering any CVT's it receives from the Headend and passing them to the Host. The POD module shall pass a CVT to the Host only if it meets all of the following criteria:

- CVT vendor_id matches Host vendor_id, and
- CVT hardware_version_id matches Host hardware_version_id

Only one code object shall be on the carousel at any given time for a given vendor_id and hardware_id.

The Host acknowledges the receipt of the CVT and responds with an OK or an appropriate error code message in the code_version_table_reply. The POD continues to transmit the CVT until it receives the ACK message. If a new, different CVT is received during this time, the POD module shall transmit it to the Host, if appropriate to the selection criteria described above.

It is up to the Host to determine if a download is required, the POD module shall not determine this. When the Host receives a valid CVT (the OUI and hardware_version_id match), the Host shall determine if the code file name in the CVT matches the code file name stored in non-volatile memory when the code was last updated. If the file names do not match, the action of the Host shall be according to the download_command parameter. If the length of the CVT software upgrade filename is different than the length of the Host software upgrade filename, then they shall be declared to be different, independent of their contents. If the download_command parameter equals 03 (download now, no exceptions), then the Host shall ignore the code file name that is stored in non-volatile memory and shall begin to download of the file referred to in the CVT.

It is also up to the Host design to handle error conditions without lockouts or wait states as well as to authenticate the vendor parameters and download code. The Host shall assume that the POD module is operating correctly.

The use of frequency vector and PID avoids the use of the virtual channel table, which assumes that the entire SI is being processed. When the download is on-demand (download_type parameter equal 01), the POD may send a CVT with frequency and PID equal to 0, to signify that the location of the code file is not

known. In this case, an additional CVT with the frequency and PID should be sent when the location is known.

In a program corruption case, the CVT may not be available.

Table 8.15-F code version table

Syntax	# of bits	Mnemonic
code_version_table() {		
code_version_table_tag	24	Uimsbf
length_field()		
number of descriptors	8	Uimsbf
for(i=0;i<number of descriptors;i++){		
descriptor_tag	8	Uimsbf
descriptor_len	8	Usmsbf
descriptor_data()		
}		
download_type	4	Uimsbf
download_command	4	Uimsbf
frequency_vector	16	Uimsbf
transport_value	8	Uimsbf
Reserved	3	Uimsbf
PID	13	Uimsnf
code_file_name_length	8	Uimsbf
for(i=0;i<software_filename_length;i++){		
code_file_name_byte	8	Uimsbf
}		
code_verification_certificate()		
}		

download_status_tag Value = 0x9F9C02

number_of_descriptors must be greater than 2, mandatory descriptors are vendor_id and hardware_version_id

descriptor_tag
0 – descriptor_data is vendor_id (mandatory, descriptor_len = 3). Unique Identifier (the vendor's OUID) assigned to each vendor.

1 – descriptor_data is hardware_version_id (mandatory, descriptor_len = 4), Unique Hardware identifier assigned to each type of hardware from a particular vendor.
2 – descriptor_data is Host proprietary data. The maximum value for descriptor_len is 128.
3-127 – reserved for future standardization
128-255 – optional, for use by POD-Host pairs, where both POD and Host support the same implementation of the Specific Application Resource. Other POD-Host pairs shall skip these descriptors using descriptor_len value

download _type Type of download (supplied by Headend) :
00 One-way, broadcast
01 Always On Demand
02 DOCSIS tftp
03 Download unsupported – no code object available. In this case, the code_file_name_length, frequency_vector and PID parameters must be equal to zero. NOTE: If the Host does not receive a CVT, then it shall assume that no code object is available.
04 Conditional On Demand

download_command
00 Download now
01 Deferred download
02 Download now, no exceptions
03 Reserved

frequency_vector Frequency of the download carousel. The frequency is coded as the number of 0.25 MHz intervals. If download_type parameter equals 02, this parameter shall be set to 0

transport_value

Value	Type
00	DOCSIS channel
01	FAT channel/QAM-64
02	FAT channel/QAM-256
03-FF	Reserved

PID Stream identifier of the code file. If download_type parameter equals 02, this parameter shall be set to 0.

code_file_name_length length of code file name

code_file_name_byte Name of software upgrade file on carousel. This is the name of the Code File (see SCTE 23-2 2002) that is on the broadcast carousel as well as in Host Flash

code_verification_certificate Authentication certificate(s) per SCTE 23-2 2002

8.15.3.6 code_version_table_reply

When the Host receives a CVT APDU, it shall respond with the code_version_table_reply APDU. This response serves as an acknowledgement to the receipt of the CVT and an error code if necessary. When the POD receives this APDU, it shall stop sending CVTs to the Host.

Table 8.15-G code_version_table_reply

Syntax	# of bits	Mnemonic
code_version_table_reply() {		
code_version_table_reply_tag	24	uimsbf
length_field()		
host_response	8	uimsbf
}		

download_status_reply_tag Value = 0x9F9C03

host_response host response to download status:

00	ACK, no error
01	Invalid vendor ID or hardware version ID.
02	Other parameter error
03-FF	Reserved

8.15.3.7 host_download_control

This APDU is used when the download_type parameter equals 01 (on-demand download) in CVT.

If the Host was told to do conditional_on_demand, the Host shall first determine if its new download image is already loaded on the DSM-CC carousel. If it is, then it shall send the host_download_control with the host_command as (00) start download.

If the Host was told to do conditional_on_demand and it does not find its download image, or it was told to do always_on_demand, then it shall send the host_download_control with the host_command as (02), notify_headend. The POD module shall then send a host_notification to the Headend along with the vendor_id and hardware_version_id. The Headend shall then send the host_download_control with the host_command as (00) start download.

When the Host has successfully authenticated the code file and it had request that the POD module send the host_notification, it shall send an additional host_download_control APDU with a host_command parameter equal to 01 (done).

The POD shall then send the done message to the Headend along with the vendor_id and hardware_version_id so that the code file can be unloaded from the carousel.

Table 8.15-H host_download_control table

Syntax	# of bits	Mnemonic
host_download_control() {		
host_download_control_tag	24	Uimsbf
length_field()		
host_command	8	Uimsbf
}		

host_download_control_tag Value = 0x9F9C04

host_command host command:
00 Start download
01 Download Completed – sent when done receiving data.
02 Notify headend
03-FF Reserved

8.15.3.8 host_download_command

The POD shall utilize the host_download_command APDU to command a Host to initiate a download when using the two-way Inband FAT channel commanded download method. The POD shall also utilize this APDU to command a Host to use the values defined within the APDU to locate CVDTs instead of the source ID when using the one-way Inband Forward Application Transport Channel broadcast download method

Table 8.15-I host_download_command

Syntax	# of bits	Mnemonic
host_download_command() {		
host_download_control_tag	24	Uimsbf
length_field()		
host_command	8	Uimsbf
location_type	8	uimsbf
if(location_type == 00){		
source_id	16	uimsbf
}		
if(location_type == 01){		
frequency_vector	16	uimsbf
transport_value	8	uimsbf
stream_ID	8	uimsbf
if(stream_ID == 00){		
Reserved	3	uimsbf
PID	13	uimsbf
}		
if(stream_ID == 01){		
program_number	16	uimsbf
}		
}		
}		

host_download_control_tag Value = 0x9F9C05

host_command Defines the priority of download.
00 Check for download during next cycle using source_id
01 Download now
02 Deferred download
03 Download now, no exceptions
04-FF Reserved

location_type Defines the method in which the Host device is to utilize to acquire the DSM-CC stream.
00 Transport Stream location is defined in the channel map and may be found via the defined source ID

01 Indicates that the Host device is to use the frequency, modulation type and PID or program number to acquire the DSM-CC stream.
02 – FF Reserved

source_id The VCT source ID that is associated with each program source. The source ID is utilized to locate the frequency that the DSC-CC data carousel is multiplexed on. A value of zero indicates that the download is located via the previously assigned source ID.

frequency_vector Frequency of the download carousel. The frequency is coded as the number of 0.25 MHz intervals. If download_type parameter equals 02, this parameter must be set to 0.

transport_value	Value Type
00	DOCSIS channel
01	FAT channel/QAM-64
02	FAT channel/QAM-256
03-FF	Reserved

stream_ID Defines the way in which the DSM-CC is to be located.
00 Indicates that the DSM-CC stream is to be located utilizing the defined PID.
01 Indicates that the stream is to be located utilizing the program number.
02 – FF Reserved

PID Packet identifier of the stream that contains the code file.

program_number Defines the program number in which the DSM-CC stream resides

APPENDIX A. Operational Modes (Informative)

A.1. Data Path Options

The Low Speed Communication resource allows the Host to share with the POD the different supported communication channels. This appendix describes data path options for various configurations.

Compliant Hosts must implement a "POD RX" channel (QPSK downstream modem), through which the POD Module expects to receive its messages. The following table describes standardized data paths with and without the availability of a "Host RX" channel.

Table A.1-A Table Downstream Data Paths

Host RX implemented (e.g. DOCSIS)	Standardized Downstream Data Paths
No	• POD RX → Extended Channel → Host • POD RX
Yes	• Host RX • POD RX • Host RX → Extended Channel → POD

When a "POD TX" channel is implemented, the POD Module expects to send its messages through its OOB channel. Other combinations are described in the following table.

Table A.1-B Upstream Data Paths		
POD TX implemented (e.g. QPSK upstream modem)	**Host TX implemented (e.g. DOCSIS)**	**Standardized Upstream Data Paths**
No	No	• None
No	Yes	• Host TX • POD → Extended Channel → Host TX
Yes	No	• Host → Extended Channel → POD TX • POD TX
Yes	Yes	• Host TX • POD TX

A.2. OOB TX Channel Available

When the Host includes POD TX support, it includes the RF circuitry that enables the POD Module to control the out-of-band. The POD Module demultiplexes from the OOB stream the Host's application messages and the POD Module's application messages and transmit the Host's application messages through the Extended Channel, and keeps its own application messages for its own use. On the return path, the POD Module multiplexes the Host's application messages with its own messaging and transmits them to the Cable System.

Figure A.2-1 shows the example case where the POD has offered a bi-directional IP packet flow across the interface. The Extended channel also supports MPEG sections uni-directionally from POD to Host.

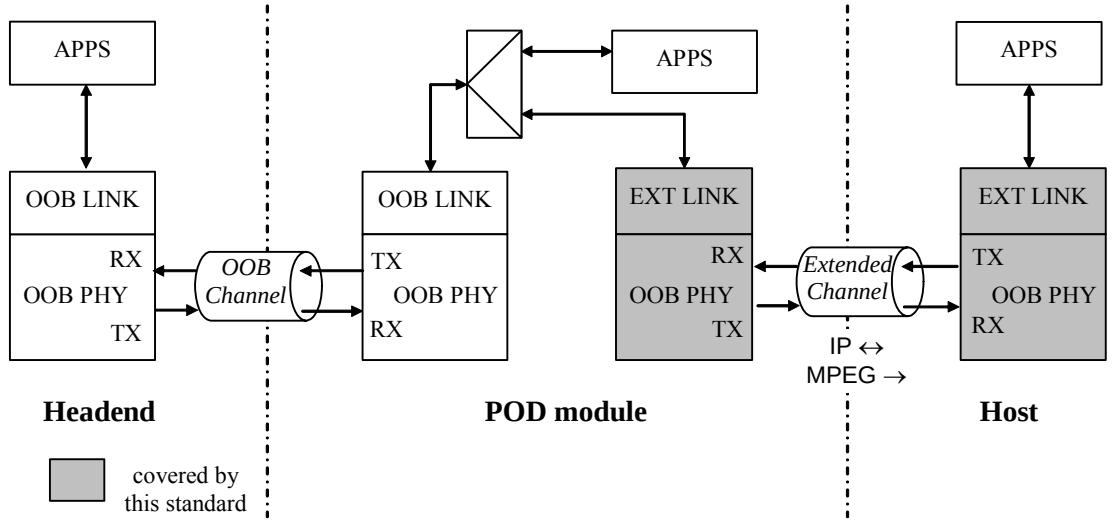

Figure A.2-1 OOB TX Channel Available

A.3. High Speed Modem Available

When the Host includes a High Speed Host Modem (e.g. DOCSIS), the POD Module still receives MPEG tables from the OOB channel, but relies on the HSHM for sending and receiving IP messages. The module OOB TX function may or may not be available.

A.3.1. OOB TX Channel Available

Figure A.3-1 shows the case that the OOB TX function is available. In this case, the POD Module uses it for upstream communications. Now, the POD has requested an IP flow from the Host, which must make its High Speed modem available for its use. It now has two possible paths for upstream data, the HSHM and the QPSK

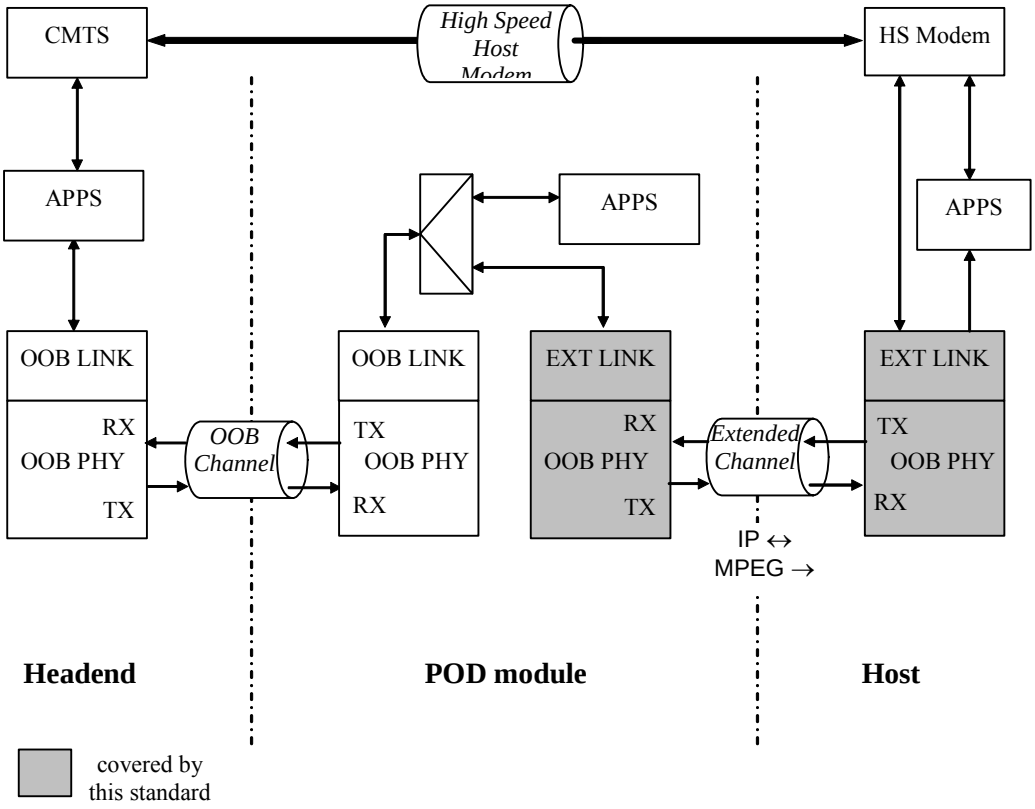

Figure A.3-1 High Speed Host Modem and OOB TX Channel Available

A.3.2. OOB TX Channel Not Available

Figure A.3-2 shows the case that the OOB TX function is not available. In this case, the POD Module uses the HSHM for upstream communications.

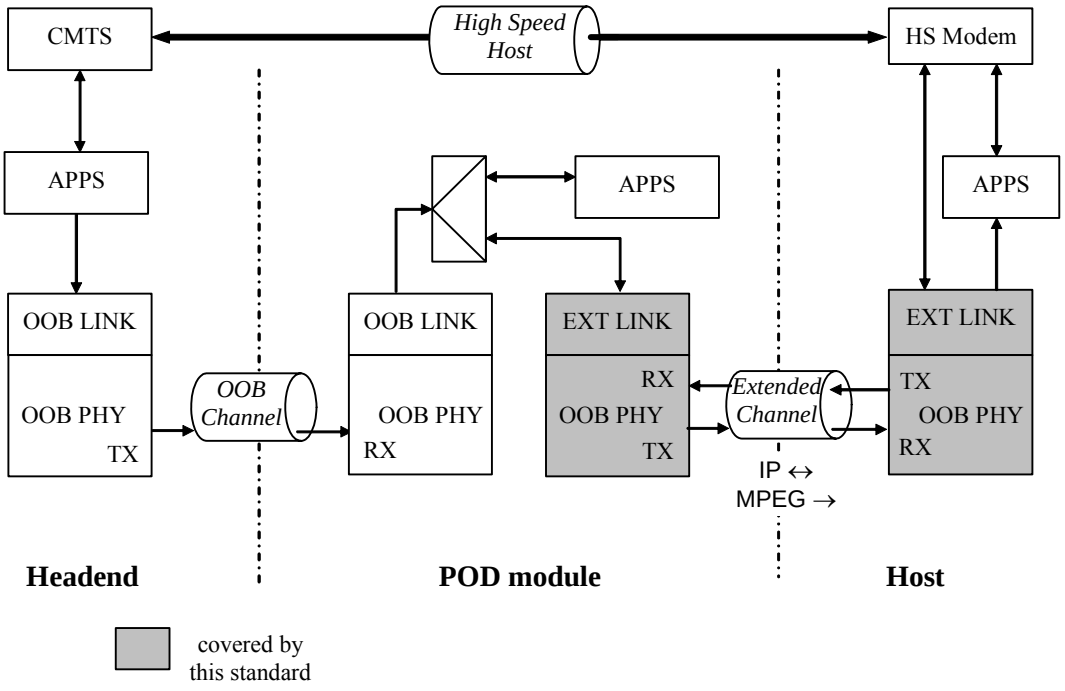

Figure A.3-2 High Speed Host Modem Available, OOB TX Channel Not Available

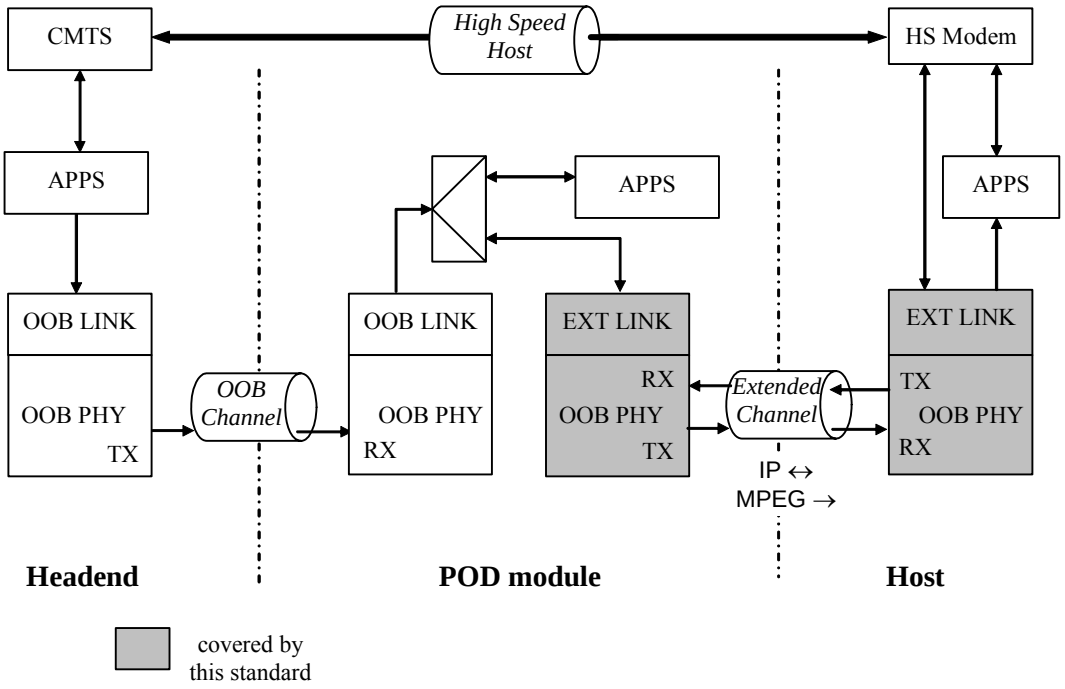

Figure A.3-3 High Speed Host Modem Available, OOB TX Channel Not Available

APPENDIX B. Glossary

Term	Definition
A1	Address Line 1
ANSI	American National Standards Institute
APDU (Application Protocol Data Unit)	A common structure to send application data between POD module and Host.
API (Application Program Interface)	The software interface to system services or software libraries. An API can consist of classes, function calls, subroutine calls, descriptive tags, etc.
ASCII (American Standard Code for Information Interchange)	Internationally recognized method for the binary representation of text
ATSC	Advanced Television System Committee.
AUT	AUT is a field in the status register that is set to 1 when the program event has been authorized.
BVD1	Chip pin acronym defined in CEA-679-C Part B.
BVD2	Chip pin acronym defined in CEA-679-C Part B.
CA	Conditional Access
CAN	CAN is a field in the status register that is set to 1 when the program event has been cancelled.
CD1#	Chip pin acronym defined in CEA-679-C Part B.
CD2#	Chip pin acronym defined in CEA-679-C Part B.
CE#2	Card Enable
CE1#	Chip pin acronym defined in CEA-679-C Part B.
CEA	Consumer Electronics Alliance
CIS (Card Information Structure)	Low-level configuration information contained in the POD module's Attribute Memory.
CMOS	Complementary Metal Oxide Silicon
Cnf	Chip pin acronym defined in CEA-679-C Part B.
Conditional Access and encryption	Conditional access and encryption is a system that provides selective access to programming to individual customers.
CPU Interface	The logical interface between the POD module and the Host comprised of the Data and Extended communications channels.

APPENDIX B. Glossary

Term	Definition
CRC	Cyclic Redundancy Check.
CRX	Chip pin acronym defined in CEA-679-C Part B.
CTX	Chip pin acronym defined in CEA-679-C Part B.
D1	Chip pin acronym defined in CEA-679-C Part B.
D3	Chip pin acronym defined in CEA-679-C Part B.
DA	Data Available
DAIE	Chip pin acronym defined in CEA-679-C Part B.
DHCP (Dynamic Host Configuration Protocol)	An Internet standard for assigning IP addresses dynamically to IP hosts.
DOCSIS	Data-Over-Cable Service Interface Specification
Downstream	Transmission from head-end to Host.
DRX	Chip pin acronym defined in CEA-679-C Part B.
DSM-CC	Digital Storage Medium – Command and Control.
DSM-CC-UU	Digital Storage Medium – Command and Control – User to User.
DVB	Digital Video Broadcast.
DVS	Digital Video Subcommittee
EAS	Emergency Alert System
EIA	Electronic Industries Alliance
EIT (Event Information Table)	An MPEG 2 table contained within the Program and System Information Protocol ("PSIP") which provides information for events on the virtual channels.
EMM (Entitlement Management Message)	A conditional access control message to a Host.
ETT (Extended Text Table)	An MPEG 2 table contained in the Program and System Information Protocol ("PSIP") which provides detailed descriptions of virtual channels and events.
ETX	Chip pin acronym defined in CEA-679-C Part B.
FAT (Forward Applications Transport) Channel	A data channel carried from the headend to the set-top terminal in a modulated channel at a rate of 27 or 36 Mbps. MPEG-2 transport is used to multiplex video, audio, and data into the FAT channel.

APPENDIX B. Glossary

Term	Definition
FDC (Forward Data Channel)	An out-of-band ("OOB") data channel from the headend to the Host.
Forward	See **Downstream**
FRE	FRE is a field in the status register that is set to 1 when the free preview of the program event has been viewed.
FRIE	Chip pin acronym defined in CEA-679-C Part B.
Gapped clock	A periodic signal in which some transitions are omitted creating gaps in a clock pattern.
GND	Ground
H	Chip pin acronym defined in CEA-679-C Part B.
HC	Header Check
Headend	The point at which all signals are collected and formatted for placement on the cable system.
Host	The consumer device used to access and navigate cable content. Typically a digital TV or set-top DTV receiver,
HTML (HyperText Markup Language)	A presentation language for the display of multiple media contents, typically used on the Internet.
HTTP (HyperText Transport Protocol)	The transport layer for HTML documents over the Internet Protocol ("IP").
I/O	Input or output
ID	Identifier
IIR (Initialize Interface Request)	Data bit in Status register
INPACK#	Chip pin acronym defined in CEA-679-C Part B.
In-band	Within the main Forward Applications Transport channel
IOIS6#	Chip pin acronym defined in CEA-679-C Part B.
IORD#	Chip pin acronym defined in CEA-679-C Part B.
IOWR#	Chip pin acronym defined in CEA-679-C Part B.
IP (Internet Protocol)	Network layer protocol in the TCP/IP stack offering a connectionless internetwork service.
IP_U (IP Unicast)	Point-to-Point Internet Protocol datagram service

APPENDIX B. Glossary

Term	Definition
IP_M(IP Multicast)	Point to multi-point Internet Protocol datagram service
IPG	Interactive Program Guide
IPPV	Impulse Pay-per-View
IREQ#	Chip pin acronym defined in CEA-679-C Part B.
ITX	Chip pin acronym defined in CEA-679-C Part B.
KHz	Kilo Hertz
Low Speed Communications Resource	Communications protocol defined in CEA-679-C Part B
LSB	Least Significant Bit or Byte
M/L	Chip pin acronym defined in CEA-679-C Part B.
MA	Milli-Amps
MAC	Media Access Control
Master Guide Table	An MPEG 2 table which provides version, size and PIDs of other tables contained within the transport stream.
MCLKO	Chip pin acronym defined in CEA-679-C Part B.
MCLKI	Chip pin acronym defined in CEA-679-C Part B.
MDO1	Chip pin acronym defined in CEA-679-C Part B.
MISTRT	Chip pin acronym defined in CEA-679-C Part B.
MIVAL	Chip pin acronym defined in CEA-679-C Part B.
MMI	Man Machine Interface
MOSTRT	Chip pin acronym defined in CEA-679-C Part B.
MOVAL	Chip pin acronym defined in CEA-679-C Part B.
MPEG (Moving Picture Experts Group)	Colloquial name for ISO-IEC SC29/WG11, which develops standards for compressed full-motion video, still image, audio, and other associated information.
MPEG-2 Video	ISO-IEC 13818-2, international standard for the compression of video.
MPEG-2 Transport	ISO-IEC 13818-1, international standard for the transport of compressed digital media.

APPENDIX B. Glossary

Term	Definition
Ms	Milli-second
MSB	Most Significant Bit or Byte
Ns	Nano-second
NTSC (National Television Systems Committee)	An entity that developed the analog television system used in North America and elsewhere.
OE#	Chip pin acronym defined in CEA-679-CPart B.
OOB (Out-of-Band) Channel	The combination of the Forward and Reverse Data Channels. The OOB channel provides a data communication channel between the cable system and the Host.
OPU	OPU is a field in the status register that is set to 1 when the program event has been purchased for taping once.
PC Card	A device that complies with the PC Card Standard, as specified in Section 2.3.1 of this document.
PCMCIA	Personal Computer Memory Card International Association
PDU (Protocol Data Unit)	A packet of data passed across a network or interface
PF	Pico-farad
PHY	Physical Layer
PID	Packet Identifier.
PIN	Personal Identification Number
PNG	Portable Network Graphics.
POD	Point-of-Deployment; The Point-of-Deployment module is also known as a CableCARD™ device.
PPV	Pay-Per-View.
PSI	Program Specific Information.
QAM (Quadrature Amplitude Modulation)	A digital modulation method in which the value of a symbol consisting of multiple bits is represented by amplitude and phase states of a carrier. Typical types of QAM include 16-QAM (four bits per symbol), 32-QAM (five bits), 64-QAM (six bits), and 256-QAM (eight bits).
QPSK (Quadrature Phase Shift Key)	A digital modulation method in which the state of a two-bit symbol is represented by one of four possible phase states.
QTX	Chip pin acronym defined in CEA-679-C Part B.

APPENDIX B. Glossary

Term	Definition
R	Chip pin acronym defined in CEA-679-C Part B.
RDC (Reverse Data Channel)	An out-of-band ("OOB") data channel from the host to the headend.
REG#	Chip pin acronym defined in CEA-679-C Part B.
REP	Chip pin acronym defined in CEA-679-C Part B.
RESET	Chip pin acronym defined in CEA-679-C Part B.
Resource	A unit of functionality provided by the host for use by a module. A resource defines a set of objects exchanged between module and host by which the module uses the resource.
Reverse	See **Upstream**
RF	Radio Frequency
RFC	Request For Comments
RPC (Remote Procedure Call)	The ability for client software to invoke a function or procedure call on a remote server machine.
RX	Receive
SCTE	Society of Cable Telecommunications Engineers
Smart Card	A ISO 7816-compliant card with embedded electronics used to store or process data
SPKR#	Chip pin acronym defined in CEA-679-C Part B.
STSCHG#	Chip pin acronym defined in CEA-679-C Part B.
STCI_IFN	The interface ID number defined by PCMCIA to be included in the Custom Interface Subtuple.
Subtuple	Subset of a Tuple
System Time Table	An MPEG 2 table which provides date and time.
Td	Time interval for delay
Th	Time interval for holding
Tsu	Time interval for setup
Tuple	Data stored within a PC Card that can be used to determine the capabilities of the card
TX	Transmit

APPENDIX B. Glossary

Term	Definition
UDP (User Datagram Protocol)	Connectionless Transport layer protocol in the TCP/IP stack.
Uimsbf	Unsigned Integer Most Significant Bit First
UPU	UPU is a field in the status register that is set to 1 when the program event has been purchased for unlimited taping.
Upstream	Transmission from host to head-end
URL (Uniform Resource Locator)	A standard method of specifying the location of an object or file.
V	Volt
VCC	Chip pin acronym defined in CEA-679-C Part B.
VIE	VIE is a field in the status register that is set to 1 when the program event has been viewed.
VCT (Virtual Channel Table)	An MPEG 2 table which contains a list of all the channels that are or will be on plus their attributes.
VOD	Video on Demand.
VPP-1	Chip pin acronym defined in CEA-679-C Part B.
VPP-2	Chip pin acronym defined in CEA-679-C Part B.
VPU	VPU is a field in the status register that is set to 1 when the program event has been purchased for viewing once.
VS1	Chip pin acronym defined in CEA-679-C Part B.
WAIT#	Chip pin acronym defined in CEA-679-C Part B.
WE#	Chip pin acronym defined in CEA-679-C Part B.

APPENDIX C. Baseline HTML Profile Requirements

This appendix describes HTML keywords that shall be supported by the Baseline HTML Profile and gives for each keyword the requirements foreseen on the Host.

The Baseline HTML Profile 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 POD can supply HTML pages that take advantage of these enhanced features.

This extended mode of operation is not described in Appendix C.

C.1. Format

C.1.1. Display

Baseline HTML Profile 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.

MMI messages from the POD shall be limited to a maximum of 16 lines of 32 characters each. If the MMI message is longer than 16 lines, the message may include up to 16 lines of text including a hyperlink pointing to an additional page.

Under the Baseline HTML Profile the Uni-Directional Receiving Device shall support minimum of 32 characters per line and a minimum of 16 lines of characters in a window without vertical and horizontal scrolling. The vertical height of the characters shall be approximately 1/18 of the screen height.

The Host shall display all text of every page visibly on the screen.The Host device may use screen space around the MMI message for navigation buttons such as “Press MENU to Exit” and/or status information. Host-added navigation buttons and status information, if added, shall not obscure any MMI text.

If the HTML from the POD contains a hyperlink, the Host manufacturer shall provide instructions on how to navigate to any links contained in the POD’s HTML message. Instructions may be provided by on-screen display or by instructions in the owner’s manual.

The Baseline HTML Profile requires that MMI windows shall be opaque.

C.1.2. Font

The Baseline HTML Profile font shall support a minimum of 32 characters per line, and a minimum of 16 lines of characters.

C.1.3. Text and Background Colors

Under the Baseline HTML Profile, the Host may render text color as requested in the HTML data from the POD.

Under the Baseline HTML Profile, the Host may render the background color as requested in the HTML data from the POD.

If the HTML data does not include a background and/or text color command, or the Host does not support the background and/or color command, the Host shall use either

1) black (#000000) text on a light gray (#C0C0C0) background or

2) white (#FFFFFF) text on a black (#000000) background.

If the Host device supports either the background color or text color command then it shall support both of the commands. It should not support only one of the commands. (Footnote: Supporting only one of the commands could lead to unreadable messages, for example if the POD requests gray text on a white background and the Host supports the text color command but uses the default background color, the result would be gray text on a gray background).

C.1.4. Unvisited Link Color

Under the Baseline HTML Profile, the Host may render the unvisited link color as requested in the HTML data from the POD. If the HTML data does not include an unvisited link color command, or the Host does not support the unvisited link color command, the Host shall use blue (#0000FF).

C.1.5. Paragraph

Under the Baseline HTML Profile, the Host may align paragraphs as requested by the HTML data from the POD. If the HTML data does not include a paragraph alignment command, or the Host does not support the paragraph alignment command, the Host shall use a LEFT paragraph alignment.

C.1.6. Image

The Baseline HTML Profile does not require support for images.

C.1.7. Table

The Baseline HTML Profile does not require support for tables.

C.1.8. Forms

The Baseline HTML Profile does not require support for forms.

C.2. Supported User Interactions

C.2.1. Navigation and Links

The Baseline HTML Profile 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 POD module using the server_query object. It is up to the POD module manufacturer to determine how results are returned to the POD module through the URL of the server_query object. . The Host shall provide a method of user navigation to the hyperlink in the MMI message if one is present.

C.3. HTML Keywords

The following table lists HTML keywords used in the Baseline HTML Profile (R=required or 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 C.3-A Keyword List	
STRUCTURE	
<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) • **text:** color of text, default: black (#000000) • **link:** color of unvisited links, default: blue (#0000FF)	 O O O
<A href= ....>...</A>	R
Begin and end an anchor • **href:** URL targeted by this anchor	 R
Style Element	
<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	

C.4. Characters

An HTML page can refer to all characters by their numeric value by enclosing them between the & and ; symbols. For example, the quotation mark " can be expressed as " in an HTML page. The characters also have mnemonic names as well. Thus, the following 3 expressions are interpreted as a character ":

"
"
"

Note: Mnemonic expressions are case sensitive.

Table C.5-A defines characters, their numeric and mnemonic expressions that Baseline HTML viewer shall support. Any baseline HTML page shall not use the characters, numeric or mnemonic expressions which are not defined in the table C.5-A. The host device may ignore the characters which are not defined in the table C.5-A.

This list is taken from the HTML 4 Character entity references found at:

http://www.w3.org/TR/1999/REC-html401-19991224/sgml/entities.html.

Table C.4-A Characters

Character	Name	Numeric Expression	Mnemonic Expression
	Horizontal tab		
	Line feed
	Space	&#32	
!	Exclamation mark	!	
"	Quotation mark	"	"
#	Number sign	#	
$	Dollar sign	$	
%	Percent sign	%	
&	Ampersand	&	&
'	Apostrophe	'	
(	Left parenthesis	(	
)	Right parenthesis	)	
*	Asterisk	*	
+	Plus sign	+	
,	Comma	,	
-	Hyphen	-	
.	Period	.	
/	Solidus (slash)	/	
0		0	
1		1	
2		2	
3		3	
4		4	
5		5	
6		6	
7		7	
8		8	
9		9	
:	Colon	:	
;	Semicolon	;	
<	Less than	<	<
=	Equals sign	=	
>	Greater than	>	>
?	Question mark	?	
@	Commercial at	@	
A		A	
B		B	
C		C	

Table C.4-A Characters			
D		&#68;	
E		&#69;	
F		&#70;	
G		&#71;	
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;	

Table C.4-A Characters			
z		z	
{	Left curly brace	{	
|	Vertical bar			
}	Right curly brace	}	
~	Tilde	~	
	Non-breaking space		
¡	Inverted exclamation	¡	¡
¢	Cent	¢	¢
£	Pound	£	£
¤	Currency	¤	¤
¥	Yen	¥	¥
¦	Broken vertical	¦	¦
§	Section sign	§	§
¨	Umlaut/diaeresis	¨	¨
©	Copyright	©	©
ª	Feminine	ª	ª
«	Left angle quote	«	«
¬	No sign	¬	¬
-	Hyphen		
®	Reg. trade mark	®	®
¯	Macron	¯	¯
°	Degrees	°	°
±	Plus/Minus	±	±
²	Superscript 2	²	²
³	Superscript 3	³	³
´	Acute accent	´	´
µ	Micron	µ	µ
¶	Paragraph sign	¶	¶
·	Middle dot	·	·
¸	Cedilla	¸	¸
¹	Superscript 1	¹	¹
º	Masculine	º	º
»	Right angle quote	»	»
¼	One quarter	¼	¼
½	One half	½	½
¾	Three quarters	¾	¾
¿	Inverted question mark	¿	¿
À	A Grave	À	À
Á	A Acute	Á	Á
Â	A Circumflex	Â	Â
Ã	A Tilde	Ã	Ã
Ä	A Diaeresis	Ä	Ä
Å	A Ring	Å	Å
Æ	AE Diphthong	Æ	Æ
Ç	C Cedilla	Ç	Ç
È	E Grave	È	È
É	E Acute	É	É
Ê	E Circumflex	Ê	Ê
Ë	E Diaeresis	Ë	Ë
Ì	I Grave	Ì	Ì
Í	I Acute	Í	Í
Î	I Circumflex	Î	Î
Ï	I Diaeresis	Ï	Ï
Ð	Icelandic eth	Ð	Ð

Table C.4-A Characters			
Ñ	N Tilde	Ñ	Ñ
Ò	O Grave	Ò	Ò
Ó	O Acute	Ó	Ó
Ô	O Circumflex	Ô	Ô
Õ	O Tilde	Õ	Õ
Ö	O Diaeresis	Ö	Ö
×	Multiplication	×	×
Ø	O Slash	Ø	Ø
Ù	U Grave	Ù	Ù
Ú	U Acute	Ú	Ú
Û	U Circumflex	Û	Û
Ü	U Diaeresis	Ü	Ü
Ý	Y Acute	Ý	Ý
Þ	Icelandic Thorn	Þ	Þ
ß	Small sharp S	ß	ß
à	a Grave	à	à
á	a Acute	á	á
â	a Circumflex	â	â
ã	a Tilde	ã	ã
ä	a Diaeresis	ä	ä
å	a Ring	å	å
æ	ae Diphthong	æ	æ
ç	c Cedilla	ç	ç
è	e Grave	è	è
é	e Acute	é	é
ê	e Circumflex	ê	ê
ë	e Diaeresis	ë	ë
ì	i Grave	ì	ì
í	i Acute	í	í
î	i Circumflex	î	î
ï	i Diaeresis	ï	ï
ð	Icenlandic eth	ð	ð
ñ	n Tilde	ñ	ñ
ò	o Grave	ò	ò
ó	o Acute	ó	ó
ô	o Circumflex	ô	ô
õ	o Tilde	õ	õ
ö	o Diaeresis	ö	ö
÷	Division	÷	÷
ø	o Slash	ø	ø
ù	u Grave	ù	ù
ú	u Acute	ú	ú
û	u Circumflex	û	û
ü	u Diaeresis	ü	ü
ý	y Acute	ý	ý
þ	Icenlandic thorn	þ	þ
ÿ	y Diaeresis	ÿ	ÿ

APPENDIX D. POD Module Attribute and Configuration Registers

D.1. General

This appendix was originally documented in SCTE –DVS/222 and has been included in this document.

This appendix is a detailed map of the attribute registers and configuration option register of the SCTE Point of Deployment (POD) module. It is assumed that the reader is familiar with the PC Card tuple arrangement for the attribute registers.

D.2. Attribute Tuples

The following is a list of the attribute tuples which must be implemented in the POD module.

CISTPL_LINKTARGET
CISTPL_DEVICE_OA
CISTPL_DEVICE_OC
CISTPL_VERS_1
CISTPL_MANFID
CISTPL_CONFIG
CCST_CIF
CISTPL_CFTABLE_ENTRY
STCE_EV
STCE_PD
CISTPL_NO_LINK
CISTPL_END

D.2.1. CISTPL_LINKTARGET

Defined in section 3.1.4 of PC Card Metaformat [10], this is recommended by the PC Card standard for low voltage PC Cards for robustness. This would be in addition to the tuples defined in [1] and would be the first tuple.

Table D.2-A CISTPL_LINKTARGET									
Byte	**Address (hex)**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
0	00	TPL_CODE = CISTPL_LINKTARGET (0x13)							
1	02	TPL_LINK = 0x03							
2	04	TPL_TAG (3 bytes) = 0x43 (C)							
3	06	0x49 (I)							
4	08	0x53 (S)							

D.2.2. CISTPL_DEVICE_0A

Defined in section 3.2.3 of PC Card Metaformat [10] , this tuple is used to define the attribute memory operation.

Table D.2-B CISTPL_DEVICE_0A									
Byte	**Address (hex)**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
0	00	TPL_CODE = CISTPL_DEVICE_0A (0x1D)							
1	02	TPL_LINK = 0x04							
2	04	Other_Conditions_Info = 0x02							
3	06	Device_ID_1 = 0x08							
4	08	Device_Size = 0x00							
5	0A	0xFF							

D.2.3. CISTPL_DEVICE_0C

Defined in section 3.2.3 of PC Card Metaformat [10] , this tuple is used to define the common memory operation.

Table D.2-C CISTPL_DEVICE_0C									
Byte	**Address (hex0**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
0	00	TPL_CODE = CISTPL_DEVICE_0C (0x1C)							
1	02	TPL_LINK = 0x04							
2	04	Other_Conditions_Info = 0x02							
3	06	Device_ID_1 = 0x08							
4	08	Device_Size = 0x00							
5	0A	TPL_END = 0xFF							

D.2.4. CISTPL_VERS_1

Defined in section 3.2.10 of PC Card Metaformat [10] . Section A.5.6 of [1] requires that TPLLV1_MAJOR be 0x05 and that TPLLV1_MINOR = 0x00. The field name of the product shall be "OPENCABLE POD Module".

Table D.2-D CISTPL_VERS_1

Byte	Address (hex)	7	6	5	4	3	2	1	0
0	00	TPL_CODE = CISTPL_VERS_1 (0x15)							
1	02	TPL_LINK = 26+n+m							
2	04	TPLLV1_MAJOR = 0x05							
3	06	TPLLV1_MINOR = 0x00							
4	08	TPPLV1_INFO = {Name of manufacturer (n bytes)							
4+n	08+(2*n)	TPLLV1_INFO (multiple bytes) ox00 (Null)							
5+n	0A+(2*n)	0x4F (O)							
6+n	0C+(2*n)	0x50 (P)							
7+n	0E+(2*n)	0x45 (E)							
8+n	10+(2*n)	0x4E (N)							
9+n	12+(2*n)	0x43 (C)							
10+n	14+(2*n)	0x41 (A)							
11+n	16+(2*n)	0x42 (B)							
12+n	18+(2*n)	0x4C (L)							
13+n	1A+(2*n)	0x45 (E)							
14+n	1C+(2*n)	0x20 ( )							
15+n	1E+(2*n)	0x50 (P)							
16+n	20+(2*n)	0x4F (O)							
17+n	22+(2*n)	0x44 (D)							
18+n	24+(2*n)	0x20 ( )							
19+n	26+(2*n)	0x4D (M)							
20+n	28+(2*n)	0x6F (o)							
21+n	2A+(2*n)	0x64 (d)							
22+n	2C+(2*n)	0x75 (u)							
23+n	2E+(2*n)	0x6C (l)							
24+n	30+(2*n)	0x65 (e)							
25+n	32+(2*n)	0x00 (Null)							
26+n	34+(2*n)	Additional Product Information (m bytes)							
27+n	36+(2*n)	0x00 (Null)}							
27+n+m	36+(2*n)+m	TPL_END = 0xFF							

D.2.5. CISTPL_CONFIG

Defined in section 3.3.4 of PC Card Metaformat [10] with requirements in [1].

Table D.2-E CISTPL_CONFIG

Byte	Address (hex)	7	6	5	4	3	2	1	0
0	00	TPL_CODE = CISTPL_CONFIG (0x1A)							
1	02	TPL_LINK = 5+n+m+p							
2	04	0		TPCC_RMSZ				TPCC_RASZ	
3	06	0		TPCC_LAST					
4	08	n bytes of TPCC_RADR							
5+n	0A+(2*n)	m bytes of TPCC_RMSK							
6+n+m	0C+(2*(n+m))	19 bytes of TPCC_SBTPL							
25+n+m	32+(2*(n+m+p))	TPL_END = 0xFF							

TPCC_RMSZ — The number of bytes in the configuration registers Base Address in Attribute Memory Space field (TPCC_RMSK) of this tuple is the value of this field plus 1. For the POD module, this value will depend on the manufacturer.

TPCC_RASZ — The number of bytes in the Configuration Register presence mask field (TPCC_RADR field) of the tuple is this value plus 1. For the POD module, this value will depend on the manufacturer.

TPCC_LAST — One byte field which contains the Configuration Index Number of the last configuration described in the Card Configuration Table. Once the Host encounters this configuration, when scanning for valid configurations, it shall have processed all valid configurations. For the POD module, this value will depend on the manufacturer.

TPCC_RADR — The Base Address of the Configuration Registers, in an even byte of Attribute Memory (address of Configuration Register 0), is given in this field.

TPCC_RMSK — The presence mask for the Configuration Registers is given in this field. Each bit represents the presence (1) or absence (0) of the corresponding Configuration Register.

TPCC_SBTPL — The sub-tuple allows for additional configuration sub-tuples. The CCST_CIF sub-tuple must be implemented.

D.2.6. CCST_CIF

Defined in section 3.3.4.5.1 of PC Card Metaformat [10] . The interface ID number (STCI_IFN) is 0x41. STCI_STR is defined to be ‘POD_V1.00’.

Table D.2-F CCST_CIF									
Byte	**Address$_H$**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
0	00	ST_CODE = CCST_CIF (0xC0)							
1	02	ST_LINK = 0x0B							
2	04	STCI_IFN = 0x41							
3	06	STCI_IFN_1 = 0x03							
4	08	STCI_STR (multiple bytes) 0x50 (P)							
5	0A	0x4F (O)							
6	0C	0x44 (D)							
7	0E	0x5F (_)							
8	10	0x56 (V)							
9	12	0x31 (1)							
10	14	0x2E (.)							
11	16	0x30 (0)							
12	18	0x30 (0)							
13	1A	0x00 (Null)							
14	1C	TPL_END 0xFF							

D.2.7. CISTPL_CFTABLE_ENTRY

Defined in section 3.3.2 of PC Card Metaformat [10]. For the first entry TPCE_INDX has both bits 6 (Default) and 7 (Intface) set. The Configuration Entry Number is selected by the manufacturer. TPCE_IF = 0x04 – indicating Custom Interface 0. TPCE_FS shall indicate the presence of both I/O and power configuration entries. TPCE_IO is a 1-byte field with the value 0x22. The information means: 2 address lines are decoded by the module and it uses only 8-bit accesses. The power configuration entry – required by this specification, shall follow the PC Card Specification." Additionally, two sub-tuples, STCE_EV and STCE_PD shall be included.

The power descriptor for Vcc is modified to 1 A.

Table D.2-G CISTPL_CFTABLE_ENTRY									
Byte	**Address (hex)**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
0	00	TPL_CODE = CISTPL_CFTABLE_ENTRY (0x1B)							
1	02	TPL_LINK == 0x33							
2	04	TPCE_INDX = 0xC0 LOGICAL OR Config. Entry Number$_H$							
3	06	TPCE_IF = 0x04							
4	08	TPCE_FS = 0x0A							
5	0A	TPCE_PD Vcc Parameter Selection Byte = 0x38							
6	0C	TPCE_PD Vcc Static Current = Manufacturer value							
7	0E	TPCE_PD Vcc Average Current = 0x07							
8	10	TPCE_PD Vcc Peak Current = 0x07							
9	12	TPCE_PD Vpp Parameter Selection Byte = 0x78							
10	14	TPCE_PD Vpp Static Current = Manufacturer value							
11	16	TPCE_PD Vpp Average Current = 0x26							
12	18	TPCE_PD Vpp Peak Current = 0x26							
13	1A	TPCE_PD Vpp Power Down Current = Manufacturer value							
14	1C	TPCE_IO = 0x22							
15	1E	ST_CODE = STCE_EV (0xC0)							
16	20	ST_LINK = 0x10							
17	22	STEV_STRS = “NRSS_HOST” 0x4F (O)							
18	24	0x50 (P)							
19	26	0x45 (E)							
20	28	0x4E (N)							
21	2A	0x43 (C)							
22	2C	0x41 (A)							
23	2E	0x42 (B)							
24	30	0x4C (L)							
25	32	0x45 (E)							
26	34	0x5F (_)							
27	36	0x48 (H)							
28	38	0x4F (O)							
29	3A	0x53 (S)							
30	3C	0x54 (T)							
31	3E	0x00 (Null)							
32	40	0xFF							
33	42	ST_CODE = STCE_PD (0xC1)							
34	44	ST_LINK = 0x12							
35	46	STPD_STRS = “NRSS_CI_MODULE” 0x45 (O)							
36	48	0x50 (P)							
37	4A	0x45 (E)							
38	4C	0x4E (N)							
39	4E	0x43 (C)							
40	50	0x41 (A)							
41	52	0x42 (B)							
42	54	0x4C (L)							
43	56	0x45 (E)							
44	58	0x5F(_)							

Table D.2-G CISTPL_CFTABLE_ENTRY									
Byte	**Address (hex)**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
45	5A	0x4D (M)							
46	5C	0x4F (O)							
47	5E	0x44 (D)							
48	60	0x55 (U)							
49	62	0x4C (L)							
50	64	0x45 (E)							
51	66	0x00 (Null)							
52	68	0xFF							
53	6A	0xFF							

D.2.8. CISTPL_END

Defined in section 3.1.2 of PC Card Metaformat [10] .

Table D.2-H CISTPL_END									
Byte	**Address (hex)**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
0	00	TPL_CODE = CISTPL_END(0xFF)							

D.2.9. Configuration Option Register

Defined in section 4.15.1 of PC Card Electrical [9].

Table D.2-I Configuration Option Register									
Byte	**Address ,(hex)**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
0	00	SRESET	Levl REQ	Function Configuration Index					

D.2.9.1. Values to Enable POD Personality Change

SRESET – 0 (Do not soft reset (POD reset) the POD module)

LevIREQ – 1 (POD module generates Level Mode interrupts.

Function Configuration Index – Lower 6 bits of TPCE_INDX.

D.2.9.2. Operation After Invoking POD Personality Change

After the correct value is written into the configuration register, the POD module shall wait a minimum of 10 usec before switching from the PCMCIA to the POD interface.

APPENDIX E. POD Error Handling

E.1. Error Handling

When error handling requires action by both the Host and the POD module, the action by the first is designated with a "(1)". It is suggested that the POD module create a diagnostic user interface which registers with the application info resource to allow it to report any error conditions, especially in a broadcast (one-way) scenario.

Table E.1-A Error Handling

	Error condition	Failure mechanism	Host action	SCTE POD module action	Comments
1	POD READY signal does not go active.	POD	Host either 1) reports error using screen in figure (figure E.1-1), 2) retry PCMCIA resets up to two times and then report error using screen in figure (figure E-1), or 3) report error but continue to perform PCMCIA resets	None	Host reports error to user.
2	Host reads incorrect CIS values	POD	Host reports error using screen in figure E.1-1 Error Display	None	Host reports error to user.[1]
3	Host writes incorrect TPCE_INDX value to POD configuration register	Host	None	POD cannot perform any action.	Host detects as failure #4 and reports error to user.[1]

Table E.1-A Error Handling					
	Error condition	**Failure mechanism**	**Host action**	**SCTE POD module action**	**Comments**
4	Host sets data channel RS bit but POD fails to set FR bit within 5 second timeout.	POD	Host either 1) reports error using screen in figure Figure E.1-1 Error Display 2) retry PCMCIA resets up to two times and then report error using screen in figure Figure E.1-1 Error Display, or 3) reports error and continue to perform PCMCIA resets	None	Host reports error to user.[1]
5	Host sets extended channel RS bit but POD fails to set FR bit within 5 second timeout.	POD	Host either 1) reports error using screen in figure Figure E.1-1 Error Display 2) retry PCMCIA resets up to two times and then report error using screen in figure Figure E.1-1 Error Display, or 3) reports error and continue to perform PCMCIA resets	None	Host reports error to user.[1]

Table E.1-A Error Handling					
	Error condition	**Failure mechanism**	**Host action**	**SCTE POD module action**	**Comments**
6	Invalid buffer negotiation - POD data channel (buffer size < 16)	POD	Host either 1) reports error using screen in figure Figure E.1-1 Error Display 2) retry PCMCIA resets up to two times and then report error using screen in figure Figure E.1-1 Error Display, or 3) operate with smaller size	None	Host reports error to user.[1]
7	Invalid buffer negotiation - Host data channel (buffer size < 16 or greater than POD data channel buffer size)	Host	None	Minimum – POD sets IIR flag and stops responding to polls. Preferred – POD works with Host buffer size	Host reports error to user.[1]
8	Invalid buffer negotiation – POD extended channel (buffer size < 16)	POD	Host either 1) reports error using screen in figure Figure E.1-1 Error Display 2) retry PCMCIA resets up to two times and then report error using screen in figure Figure E.1-1 Error Display, or 3) operate with smaller size	None	Host reports error to user.[1]
9	Invalid buffer negotiation - Host extended channel (buffer size < 16 or greater than POD data channel buffer size)	Host	None	Minimum – POD sets IIR flag and stops responding to polls. Preferred – POD works with Host buffer size	Host reports error to user.[1]

Table E.1-A Error Handling					
	Error condition	**Failure mechanism**	**Host action**	**SCTE POD module action**	**Comments**
10	POD does not respond to Hosts open transport request within 5 seconds.	POD	Host either 1) reports error using screen in figure Figure E.1-1 Error Display 2) retry PCMCIA resets up to two times and then report error using screen in figure Figure E.1-1 Error Display, or 3) reports error and continue to perform PCMCIA resets	None	Host reports error to user.[1]
11	Host does not respond to POD request to open resource manager session within 5 seconds.	Host	None	Minimum – POD sets IIR flag and stops responding to polls.	Host reports error to user.[1]
12	Host response to open resource manager session response - resource manager non-existent	Host	None	Minimum – POD sets IIR flag and stops responding to polls.	Host reports error to user.[1]
13	Host response to open resource manager session response - resource manager unavailable	Host	None	Minimum – POD sets IIR flag and stops responding to polls.	Host reports error to user.[1]
14	Host response to open resource manager session response - incorrect version of resource manager	Host	None	Minimum – POD sets IIR flag and stops responding to polls.	Host reports error to user.[1]
15	Host response to open resource manager session response - resource manager busy	Host	None	Minimum – POD sets IIR flag and stops responding to polls.	Host reports error to user.[1]
16	Host response to open resource manager session response - invalid status byte	Host	None	Minimum – POD sets IIR flag and stops responding to polls.	Host reports error to user.[1]

Table E.1-A Error Handling					
	Error condition	**Failure mechanism**	**Host action**	**SCTE POD module action**	**Comments**
17	POD fails to respond to profile_inq within 5 seconds.	POD	Host either 1) reports error using screen in figure Figure E.1-1 Error Display 2) retry PCMCIA resets up to two times and then report error using screen in figure Figure E.1-1 Error Display, or 3) reports error and continue to perform PCMCIA resets	None	Host reports error to user.[1]
18	Host resource response - no application information resource	Host	None	Minimum – POD sets IIR flag and stops responding to polls.. Preferred – POD continues operation and will not open a session to the application info resource.	Minimum – Host reports error to user. Preferred - Applications on the POD may not operate correctly, including MMI.[1]
19	Host resource response - no Host control resource	Host	None	Minimum – POD sets IIR flag and stops responding to polls.	POD may not be able to do conditional access properly. NOTE: There is a discussion ongoing about DOCSIS only operation.[1]
20	Host resource response - no system time resource	Host	None	Minimum – POD continues operation and will not open a session to the system time resource. Preferred – Same as minimum but also reports this in its MMI diagnostics application.	POD operations which require system time will not operate.[1]

Table E.1-A Error Handling					
	Error condition	**Failure mechanism**	**Host action**	**SCTE POD module action**	**Comments**
21	Host resource response - no MMI resource	Host	None	Minimum – POD continues operation and will not open a session to the MMI resource.	POD cannot utilize MMI for applications or to report error conditions.[1]
22	Host resource response - no low speed communications	Host	None	Minimum – POD continues operation and will not open a session to the low speed communication resource. Preferred – Same as minimum but also reports this in its MMI diagnostic application.	If OOB reverse path not available, then some applications will be unavailable.[1]
23	Host resource response - no homing resource[1]	Host	None	Minimum – POD continues operation and will not open a session to the homing resource. Preferred – Same as minimum but also reports this in its MMI diagnostic application.	POD may have some operational problems (i.e. downloading software).[1]
24	Host resource response - no copy protection resource	Host	None	Minimum – POD continues operation, disables descrambling of all conditional access channels, it will not open a session to the copy protection resource, reports to headend if possible, reports error to user, and reports this in its MMI diagnostic application.	All CA channels will not be descrambled, only clear channels may be viewed.[1]
25	Host resource response - unknown resource identifier	Host	None	Minimum – POD continues operation.	Not a failure condition
26	Host fails to respond to open session request within 5 seconds.	Host	None	Minimum – POD sets IIR flag and stops responding to polls.	Host reports error to user.[1]

Table E.1-A Error Handling					
	Error condition	**Failure mechanism**	**Host action**	**SCTE POD module action**	**Comments**
27	Host response to open application info resource session - application info non-existent	Host	None	Minimum – POD sets IIR flag and stops responding to polls.. Preferred – POD continues operation and will not open a session to the application info resource.	Minimum – Host reports error to user. Preferred - Applications on the POD may not operate correctly, including MMI.[1]
28	Host response to open application info resource session - application info unavailable	Host	None	Minimum – POD sets IIR flag and stops responding to polls.. Preferred – POD continues operation and will not open a session to the application info resource.	Minimum – Host reports error to user. Preferred - Applications on the POD may not operate correctly, including MMI.[1]
29	Host response to open application info resource session - incorrect version of application info	Host	None	Minimum – POD sets IIR flag and stops responding to polls.. Preferred – POD continues operation and will not open a session to the application info resource.	Minimum – Host reports error to user. Preferred - Applications on the POD may not operate correctly, including MMI.[1]
30	Host response to open application info resource session - application info busy	Host	None	Minimum – POD sets IIR flag and stops responding to polls.. Preferred – POD continues operation and will not open a session to the application info resource.	Minimum – Host reports error to user. Preferred - Applications on the POD may not operate correctly, including MMI.[1]
31	Host response to open application info resource session - invalid status byte	Host	None	Minimum – POD sets IIR flag and stops responding to polls.. Preferred – POD continues operation and will not open a session to the application info resource.	Minimum – Host reports error to user. Preferred - Applications on the POD may not operate correctly, including MMI.[1]

Table E.1-A Error Handling					
	Error condition	**Failure mechanism**	**Host action**	**SCTE POD module action**	**Comments**
32	POD module requests to open conditional access session to the Host times out after 5 seconds.	Host	None	Minimum – POD sets IIR flag and stops responding to polls.	Host reports error to user.[1]
33	POD response to conditional access resource session - conditional access non-existent	Host	None	Minimum – POD sets IIR flag and stops responding to polls. Preferred – POD will not descramble but will continue other operation and reports this in its MMI diagnostic application.	Minimum - Host reports error to user. Preferred – Scrambled channels are not viewed.[1]
34	POD response to conditional access resource session - conditional access unavailable	Host	None	Minimum – POD sets IIR flag and stops responding to polls. Preferred – POD will not descramble but will continue other operation and reports this in its MMI diagnostic application.	Minimum - Host reports error to user. Preferred – Scrambled channels are not viewed.[1]
35	POD response to conditional access resource session - incorrect version of conditional access	Host	None	Minimum – POD sets IIR flag and stops responding to polls. Preferred – POD will not descramble but will continue other operation and reports this in its MMI diagnostic application.	Minimum - Host reports error to user. Preferred – Scrambled channels are not viewed.[1]
36	POD response to conditional access resource session - conditional access busy	Host	None	Minimum – POD sets IIR flag and stops responding to polls. Preferred – POD will not descramble but will continue other operation and reports this in its MMI diagnostic application.	Minimum - Host reports error to user. Preferred – Scrambled channels are not viewed.[1]

Table E.1-A Error Handling					
	Error condition	**Failure mechanism**	**Host action**	**SCTE POD module action**	**Comments**
37	POD response to conditional access resource session - invalid status byte	Host	None	Minimum – POD sets IIR flag and stops responding to polls. Preferred – POD will not descramble but will continue other operation and reports this in its MMI diagnostic application.	Minimum - Host reports error to user. Preferred – Scrambled channels are not viewed.[1]
38	POD fails to respond to ca_info_inq within 5 seconds.	POD	Host either 1) reports error using screen in figure Figure E.1-1 Error Display 2) retry PCMCIA resets up to two times and then report error using screen in figure Figure E.1-1 Error Display, or 3) reports error and continue to perform PCMCIA resets	None	Host reports error to user.[1]
39	POD module requests to open copy protection resource session to the Host times out after 5 seconds.	Host	None	Minimum – POD continues operation, disables descrambling of all conditional access channels, reports to headend if possible, reports this to user, and reports this in its MMI diagnostic application.	All CA channels will not be descrambled, only clear channels may be viewed.[1]

Table E.1-A Error Handling					
	Error condition	**Failure mechanism**	**Host action**	**SCTE POD module action**	**Comments**
40	Host response to open copy protection resource session - copy protection non-existent	Host	None	Minimum – POD continues operation, disables descrambling of all conditional access channels, reports to headend if possible, reports this to user, and reports this in its MMI diagnostic application.	All CA channels will not be descrambled, only clear channels may be viewed.[1]
41	Host response to open copy protection resource session - copy protection unavailable	Host	None	Minimum – POD continues operation, disables descrambling of all conditional access channels, reports to headend if possible, reports this to user, and reports this in its MMI diagnostic application.	All CA channels will not be descrambled, only clear channels may be viewed.[1]
42	Host response to open copy protection resource session - copy protection busy	Host	None	Minimum – POD continues operation, disables descrambling of all conditional access channels, reports to headend if possible, reports this to user, and reports this in its MMI diagnostic application.	All CA channels will not be descrambled, only clear channels may be viewed.[1]
43	Host response to open copy protection resource session - invalid status byte	Host	None	Minimum – POD continues operation, disables descrambling of all conditional access channels, reports to headend if possible, reports this to user, and reports this in its MMI diagnostic application.	All CA channels will not be descrambled, only clear channels may be viewed.[1]

Table E.1-A Error Handling					
	Error condition	**Failure mechanism**	**Host action**	**SCTE POD module action**	**Comments**
44	Host does not support POD's copy protection system.	Host/POD incompatibility	None	Minimum – POD continues operation, disables descrambling of all conditional access channels, reports to headend if possible, reports this to user, and reports this in its MMI diagnostic application.	All CA channels will not be descrambled, only clear channels may be viewed.[1]
45	Host and POD do not mate	Host/POD incompatibility	None	Minimum – POD continues operation, disables descrambling of all conditional access channels, reports to headend if possible, reports this to user, and reports this in its MMI diagnostic application.	All CA channels will not be descrambled, only clear channels may be viewed.[1]
46	Host response to CP_sync - Host busy	Host	None	Minimum – POD will cease descrambling of copy protected channels.	A copy protected channel will stop being descrambled.
47	Host response to CP_sync - no CP support	Host	None	Minimum – POD will cease descrambling of copy protected channels.	A copy protected channel will stop being descrambled.
48	Host response to CP_sync - invalid status	Host	None	Minimum – POD will cease descrambling of copy protected channels.	A copy protected channel will stop being descrambled.
49	Host fails to respond to cp_open_req.	Host	None	Minimum – POD will cease descrambling of copy protected channels.	A copy protected channel will stop being descrambled.

Table E.1-A Error Handling					
	Error condition	**Failure mechanism**	**Host action**	**SCTE POD module action**	**Comments**
50	Invalid Host certificate	Host	None	Minimum – POD continues operation, disables descrambling of all conditional access channels, reports to headend if possible, reports this to user, and reports this in its MMI diagnostic application.	All CA channels will not be descrambled, only clear channels may be viewed.[1]
51	Write Error (WE) occurs after completion of any transfer from Host to POD	POD or Host	Host performs POD reset.	None	User may see frozen picture on scrambled channels.[1]
52	Read Error (RE) occurs after completion of any transfer from POD to Host	POD or Host	Host performs POD reset.	None	User may see frozen picture on scrambled channels.[1]
53	POD fails to respond to any request within 5 seconds, other than described by error conditions 17 and 38.	POD	Host performs PCMCIA reset up to two times and then reports error using screen in figure Figure E.1-1 Error Display.	None	User may see frozen picture on scrambled channels.[1]
54	Invalid session APDU from Host	Host	None	No action	Not a failure condition
55	Invalid session APDU from POD	POD	Host ignores invalid sessions.	None	Not a failure condition
56	Invalid SPDU tag from Host	Host	None	No action	Not a failure condition
57	Invalid SPDU tag from POD	POD	Host ignores invalid SPDU tags.	None	Not a failure condition
58	Invalid APDU tag from Host	Host	None	No action	Not a failure condition
59	Invalid APDU tag from POD	POD	Host ignores invalid APDU tags.	None	Not a failure condition
60	Transport ID from Host that has not been created and confirmed by POD	Host	None	No action	Not a failure condition

Table E.1-A Error Handling					
	Error condition	**Failure mechanism**	**Host action**	**SCTE POD module action**	**Comments**
61	Transport ID from POD that has not been created by Host.	POD	Host ignores transport ID's that have not been created	None	Not a failure condition
62	Session ID from Host that has not been created and confirmed by POD	Host	None	No action	Not a failure condition
63	Session ID from POD that has not been created by Host.	POD	Host ignores session ID's that have not been created	None	Not a failure condition

NOTE: A POD reset is defined that the Host shall set the RS bit in the command interface control register. A PCMCIA reset is defined that the Host shall set the RESET signal active on the PCMCIA interface.

1 - If the error is caused by an issue with the design of the Host or POD module, this should be detected during certification.

In the even that an error occurs in which the Host must display an error message, the following message, or its equivalent, shall be displayed:

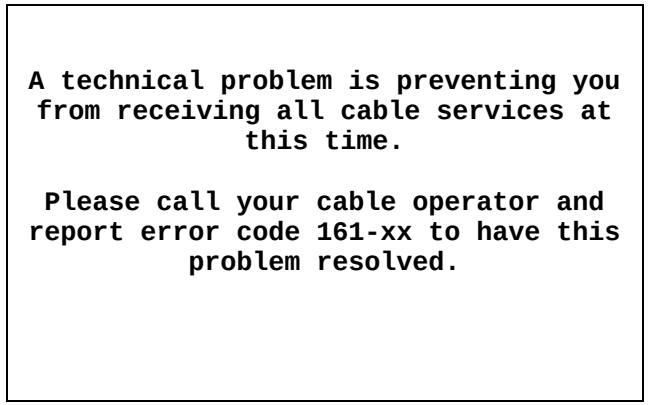

Figure E.1-1 Error Display

The “xx” after the error code 161 shall be the item number of the above table which has failed.