



**Society of Cable
Telecommunications
Engineers**

**ENGINEERING COMMITTEE
Data Standards Subcommittee**

AMERICAN NATIONAL STANDARD

ANSI/SCTE 23-3 2010

**DOCSIS 1.1 Part 3:
Operations Support System Interface**

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 non-member 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, and accepts full responsibility for any damage and/or claims arising from the adoption of such Standards.

Attention is called to the possibility that implementation of this standard may require the 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 inquiries 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>.

All Rights Reserved

© Society of Cable Telecommunications Engineers, Inc. 2010
140 Philips Road
Exton, PA 19341

DOCSIS® is a trademark of Cable Television Laboratories, Inc. (CableLabs) and is used in this document with permission

Table of Contents

1	Scope and Purpose.....	1
1.1	SCOPE.....	1
1.2	REQUIREMENTS	1
2	SNMP Protocol	2
2.1	SNMP MODE FOR DOCSIS 1.1 COMPLIANT CMTS.....	2
2.1.1	KEY CHANGE MECHANISM	3
2.2	SNMP MODE FOR DOCSIS 1.1 COMPLIANT CMS.....	3
2.2.1	SNMPV3 INITIALIZATION AND KEY CHANGES	5
2.2.2	SNMPV3 INITIALIZATION.....	5
2.2.3	DH KEY CHANGES	6
2.2.4	VACM PROFILE.....	7
3	Management Information Bases (MIBs).....	9
3.1	IPCDN DRAFTS AND OTHERS	9
3.2	IETF RFCS.....	10
3.3	MANAGED OBJECTS REQUIREMENTS	10
3.3.1	CMTS MIB REQUIREMENTS	10
3.3.2	REQUIREMENTS FOR RFC-2669.....	11
3.3.3	REQUIREMENTS FOR DOCS-IF-MIB.....	11
3.3.4	REQUIREMENTS FOR RFC-2863.....	11
3.3.5	INTERFACE MIB AND TRAP ENABLE.....	13
3.3.6	REQUIREMENTS FOR RFC-2665.....	14
3.3.7	REQUIREMENTS FOR RFC-1493.....	14
3.3.8	REQUIREMENTS FOR RFC-2011.....	14
3.3.9	REQUIREMENTS FOR RFC-2013.....	14
3.3.10	REQUIREMENTS FOR RFC-3418.....	14
3.3.11	REQUIREMENTS FOR DOCS-QOS-MIB	15
3.3.12	REQUIREMENTS FOR "DRAFT-IETF-IPCDN-IGMP-MIB-01.TXT"	15
3.3.13	REQUIREMENTS FOR RFC-2933.....	15
3.3.14	REQUIREMENTS FOR DOCS-BPI2-MIB	15
3.3.15	REQUIREMENTS FOR USB-MIB	15
3.3.16	REQUIREMENTS FOR DOCS-SUBMGT-MIB.....	15
3.3.17	REQUIREMENTS FOR RFC-2786.....	15
3.3.18	REQUIREMENTS FOR RFC-3083.....	16
3.3.19	REQUIREMENT FOR DOCS-IF-EXT-MIB	16
3.3.20	REQUIREMENTS FOR DOCS-CABLE-DEVICE-TRAP-MIB	16
3.3.21	REQUIREMENTS FOR SNMPV3 MIBS.....	16
3.4	CM CONFIGURATION FILES, TLV-11 AND MIB OIDS/VALUES.....	17
3.4.1	CM CONFIGURATION FILE TLV-11 ELEMENT TRANSLATION (TO SNMP PDU).....	17
3.4.2	IGNORE CM CONFIGURATION TLV-11 ELEMENTS WHICH ARE NOT SUPPORTED BY CM	17
3.4.3	CM STATE AFTER CM CONFIGURATION FILE PROCESSING SUCCESS.....	18
3.4.4	CM STATE AFTER CM CONFIGURATION FILE PROCESSING FAILURE.....	18
3.5	TREATMENT AND INTERPRETATION OF MIB COUNTERS ON THE CM	18
3.6	CONFIG FILE ELEMENT – SNMP V3NOTIFICATION RECEIVER	18
3.6.1	MAPPING OF TLV FIELDS INTO CREATED SNMP V3 TABLE ROWS	19
4	OSSI for Radio Frequency Interface.....	26
4.1	SUBSCRIBER ACCOUNT MANAGEMENT INTERFACE SPECIFICATION	26
4.1.1	SERVICE FLOWS, SERVICE CLASSES, AND SUBSCRIBER USAGE BILLING	26
4.1.2	IP DETAIL RECORD (IPDR) STANDARD	27

4.1.3	HIGH-LEVEL REQUIREMENTS FOR SUBSCRIBER USAGE BILLING RECORDS	28
4.1.4	BILLING COLLECTION INTERVAL	30
4.1.5	BILLING FILE RETRIEVAL MODEL	31
4.1.6	BILLING FILE SECURITY MODEL	31
4.1.7	IPDR RECORD STRUCTURE	32
4.2	CONFIGURATION MANAGEMENT	36
4.2.1	VERSION CONTROL	37
4.2.2	SYSTEM INITIALIZATION AND CONFIGURATION	37
4.2.3	SECURE SOFTWARE UPGRADES	37
4.3	PROTOCOL FILTERS	42
4.3.1	LLC FILTER	42
4.3.2	SPECIAL FILTER	42
4.3.3	IP SPOOFING FILTER	42
4.3.4	SNMP ACCESS FILTER	43
4.3.5	IP FILTER	43
4.4	FAULT MANAGEMENT	43
4.4.1	SNMP USAGE	44
4.4.2	EVENT NOTIFICATION	46
4.4.3	THROTTLING, LIMITING AND PRIORITY FOR EVENT, TRAP AND SYSLOG	53
4.4.4	NON-SNMP FAULT MANAGEMENT PROTOCOLS	54
4.5	PERFORMANCE MANAGEMENT	54
4.5.1	ADDITIONAL MIB IMPLEMENTATION REQUIREMENTS	55
4.6	COEXISTENCE	55
4.6.1	COEXISTENCE AND MIBS	56
4.6.2	COEXISTENCE AND SNMP	57
5	OSS for BPI+	58
5.1	DOCSIS ROOT CA	58
5.2	DIGITAL CERTIFICATE VALIDITY PERIOD AND RE-ISSUANCE	58
5.2.1	DOCSIS ROOT CA CERTIFICATE	58
5.2.2	DOCSIS MANUFACTURER CA CERTIFICATE	58
5.2.3	DOCSIS CM CERTIFICATE	58
5.2.4	DOCSIS CODE VERIFICATION CERTIFICATE	59
5.3	CM CODE FILE SIGNING POLICY	59
5.3.1	MANUFACTURER CM CODE FILE SIGNING POLICY	59
6	OSSI for CMCI	60
6.1	SNMP ACCESS VIA CMCI	60
6.2	CONSOLE ACCESS	60
6.3	CM DIAGNOSTIC CAPABILITIES	60
6.4	PROTOCOL FILTERING	61
6.5	MANAGEMENT INFORMATION BASE (MIB) REQUIREMENTS	61
7	CM Operational Status Visualization	62
7.1	CM LEDS REQUIREMENTS AND OPERATION	62
7.1.1	POWER AND SELF TEST	62
7.1.2	SCANNING AND SYNCHRONIZATION TO DOWNSTREAM	62
7.1.3	DOCSIS UPSTREAM OBTAINING PARAMETERS	63
7.1.4	BECOMING OPERATIONAL	63
7.1.5	DATA LINK AND ACTIVITY	63
7.2	ADDITIONAL CM OPERATIONAL STATUS VISUALIZATION FEATURES	63
7.2.1	SOFTWARE DOWNLOAD	63

Appendix A. Detailed MIB Requirements	64
Appendix B. RFC-2670 ifTable MIB-Object details.....	100
Appendix C. RFC-1493 and RFC-2570 MIB-Object Details for CCCM	112
C.1 RFC-1493 MIB-OBJECT DETAILS.....	112
C.2 IMPLEMENTATION OF RFC-1493 MIB FOR CCCM.....	114
C.2.1 RFC-2670 IFTABLE MIB-OBJECT DETAILS FOR CCCM.....	115
Appendix D. Business Process Scenarios For Subscriber Account Management.....	117
D.1 THE OLD SERVICE MODEL: “ONE CLASS ONLY” & “BEST EFFORT” SERVICE	117
D.2 THE OLD BILLING MODEL: “FLAT RATE” ACCESS.....	117
D.3 A SUCCESSFUL NEW BUSINESS PARADIGM	117
D.3.1 INTEGRATING “FRONT END” PROCESSES SEAMLESSLY WITH “BACK OFFICE” FUNCTIONS	117
D.3.2 DESIGNING CLASS OF SERVICES.....	118
D.3.3 USAGE-BASED BILLING	119
D.3.4 DESIGNING USAGE-BASED BILLING MODELS.....	119
Appendix E. IPDR.org NDM-U 3.1 Service Specification Submission for Cable Data Systems Subscriber Usage Billing Records	120
E.1 SERVICE DEFINITION	120
E.1.1 SERVICE REQUIREMENTS	120
E.1.2 SERVICE USAGE ATTRIBUTE LIST	121
E.2 XML SCHEMA SUBSCRIBER USAGE BILLING RECORDS.....	124
E.2.1 SCHEMA.....	124
E.2.2 EXAMPLE IPDRDOC XML FILE CONTAINING SUBSCRIBER USAGE IPDRS	128
Appendix F. SNMPv2c INFORM Request Definition for Subscriber Account Management (SAM).....	129
Appendix G. Summary of the CM Authentication and the Code File Authentication.....	130
G.1 AUTHENTICATION OF THE DOCS 1.1 COMPLIANT CM	130
G.1.1 RESPONSIBILITY OF THE DOCS ROOT CA	130
G.1.2 RESPONSIBILITY OF THE CM MANUFACTURERS	130
G.1.3 RESPONSIBILITY OF THE OPERATORS.....	131
G.2 AUTHENTICATION OF THE CODE FILE FOR THE DOCS 1.1 COMPLIANT CM	131
G.2.1 RESPONSIBILITY OF THE DOCS ROOT CA	132
G.2.2 RESPONSIBILITY OF THE CM MANUFACTURER	132
G.2.3 RESPONSIBILITY OF CABLELABS	132
G.2.4 RESPONSIBILITY OF THE OPERATORS.....	132
Appendix H. Format and Content for Event, SYSLOG and SNMP Trap.....	134
Appendix I. Trap Definitions for Cable Device	157
I.1 DOCS-CABLE-DEVICE-TRAP-MIB.....	157
Appendix J. Application of RFC-2933 to DOCS 1.1 active/passive IGMP devices.....	179
J.1 DOCS 1.1 IGMP MIBS.....	179
J.1.1 IGMP CAPABILITIES: ACTIVE AND PASSIVE MODE.....	179
J.1.2 IGMP INTERFACES.....	179

J.2 DOCSIS 1.1 CM SUPPORT FOR THE IGMP MIB	179
J.2.1 IGMPINTERFACETABLE- IGMPINTERFACEENTRY	179
J.2.2 IGMPCACHETABLE - IGMPCACHEENTRY.....	182
J.3 DOCSIS 1.1 CMTS SUPPORT FOR THE IGMP MIB	183
J.3.1 IGMPINTERFACETABLE- IGMPINTERFACEENTRY	184
J.3.2 IGMPCACHETABLE - IGMPCACHEENTRY.....	187
J.3.3 IGMP MIB COMPLIANCE	188
J.3.4 MIB GROUPS.....	189
Appendix K. Expected Behaviors for DOCSIS 1.1 modem in 1.0 and 1.1 modes in OSS area.....	191
Appendix L. DOCS-IF-EXT-MIB.....	194
Appendix M. DOCSIS Quality of Service MIB	197
Appendix N. Baseline Privacy Plus MIB	272
Appendix O. USB MIB.....	339
Appendix P. Subscriber Management MIB	369
Appendix Q. draft-ietf-magma-igmp-proxy-00.txt	393
Appendix R. RF Interface MIB.....	400
Appendix S. References.....	487

Figures

Figure 1. Ifindex Example for CMTS.....	12
Figure 2. Basic Network Model (ref. NDM-U 3.1 from www.ipdr.org)	28
Figure 3. Billing Collection Interval Example.....	30
Figure 4. IPDRDoc 3.1 Generic Schema	32
Figure 5. DOCSIS IPDR 3.1 Schema	33
Figure 6. Manufacture control scheme	38
Figure 7. Operator control scheme	38
Figure 8. Coexistent (DOCSIS 1.0 mode VS DOCSIS 1.1 mode).....	55
Figure 9. CM DOCSIS Mode and MIBs Requirement	56
Figure 10. Authentication of the DOCS 1.1 compliant CM	130
Figure 11. Authentication of the code file for the DOCS 1.1 compliant CM.....	132

Tables

Table 1. IPCDN Drafts	9
Table 2. IETF RFCs	10
Table 3. CM Interface numbering	12
Table 4. docsIfCmStatusValue and ifOperStatus Relationship	13
Table 5. snmpNotifyTable	19
Table 6. snmpTargetAddrTable	20
Table 7. snmpTargetAddrExtTable	20
Table 8. snmpTargetParamsTable for <Trap type> 1, 2, or 3	21
Table 9. snmp TargetParamsTable for <Trap type> 4 or 5	22
Table 10. snmpNotifyFilterProfileTable	22
Table 11. snmpNotifyFilterTable	23
Table 12. snmpCommunityTable	23
Table 13. usmUserTable	24
Table 14. vacmSecurityToGroupTable	24
Table 15. vacmAccessTable	25
Table 16. vacmViewTreeFamilyTable	25
Table 17. Default event priorities for the Cable Modem Device'	50
Table 18. Default Event priorities for CMTS supporting only local-log non-volatile	51
Table 19. Default Event priorities for CMTS supporting only local-log volatile	51
Table 20. Default Event priorities for CMTS supporting both local-log non-volatile and local-log volatile	52
Table 21. Event Priorities Assignment For CM and CMTSS	52
Table 22. Maximum Level of Support for CM Events	53
Table 23. Maximum Level of Support for CMTS Events	53
Table 24. Detailed MIB Requirements	64
Table 25. RFC-2670 ifTable MIB-Object details	100
Table 26. RFC-1493 MIB-Object Details	112
Table 27. The dot1dBase Group	114
Table 28. Dot1dBasePortTable	114
Table 29. The dot1dTp Group	115
Table 30. dot1dFdbTable	115
Table 31. dot1dTpPortTable	115
Table 32. RFC-2670 ifTable MIB-Object details for CCCM	116
Table 33. Service Usage Attribute Value Names	122
Table 34. Format and Content for Event, SYSLOG and SNMP Trap	135
Table 35. Expected Behaviors for DOCS 1.1 modem in 1.0 and 1.1 modes in OSS area	191

This page intentionally left blank

1 Scope and Purpose

1.1 Scope

This standard defines the Network Management requirements for support a DOCSIS® 1.1 environment. More specifically, the specification details the SNMP v3 protocol and how it coexists with SNMP V1/V2. The RFCs and Management Information Base (MIB) requirements are detailed as well as interface numbering, filtering, event notifications, etc. Basic network management principals such as account, configuration, fault, and performance management are incorporated in this specification for better understanding of managing a high-speed cable modem environment.

1.2 Requirements

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

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

This document defines many features and parameters and a valid range for each parameter is usually specified. Equipment (CM and CMTS) requirements are always explicitly stated. Equipment must comply with all mandatory (MUST and MUST NOT) requirements to be considered compliant with this specification. Support of non-mandatory features and parameter values is optional.

2 SNMP Protocol

The SNMPV3 protocol has been selected as the communication protocol for management of data-over-cable Services and MUST be implemented. Although SNMPv3 offers advantages, many management systems may not be capable of supporting SNMPV3 agents; therefore, support of SNMPv1 and SNMPv2c is also required and MUST be implemented.

The following IETF SNMP related RFCs MUST be implemented:

RFC-3410	Introduction and Applicability Statements for Internet Standard Management Framework
RFC-3411	An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks
RFC-3412	Message Processing and Dispatching for the Simple Network Management Protocol (SNMP)
RFC-3413	Simple Network Management Protocol (SNMP) Applications
RFC-3414	User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)
RFC-3415	View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP)
RFC-3416	Version 2 of the Protocol Operations for the Simple Network Management Protocol (SNMP)
RFC-3417	Transport Mappings for the Simple Network Management Protocol (SNMP)
RFC-3418	Management Information Base (MIB) for the Simple Network Management Protocol (SNMP)
RFC-2576	Coexistence between Version 1, Version 2, and Version 3 of the Internet-standard Network Management Framework
RFC-1901	Introduction to Community-based SNMPv2
RFC-1157	A Simple Network Management Protocol

For support of SMIv2 the following IETF SNMP related RFCs MUST be implemented:

RFC-2578	Structure of Management Information Version 2 (SMIv2)
RFC-2579	Textual Conventions for SMIv2
RFC-2580	Conformance Statements for SMIv2

For support of Diffie-Helman Key exchange for the User Based Security Model, the follow IETF SNMP related RFC MUST be implemented:

RFC-2786	Diffie-Helman USM Key Management Information Base and Textual Convention
----------	--

2.1 SNMP Mode for DOCSIS 1.1 compliant CMTS

DOCSIS 1.1 compliant CMTS MUST support SNMPv1, SNMPv2c, and SNMPv3 and SNMP coexistence as described by RFC-3411-RFC-2576 and MAY support SNMPv1, SNMPv2c vendor proprietary solutions, including SNMP v1/v2c NmAccess mode, with the following requirements:

- a) DOCSIS 1.1 compliant CMTS MUST operate in SNMP coexistence mode (not using docsDevNmAccessTable); additionally, SNMP coexistence mode MAY be disabled, by vendor proprietary configuration control, to allow the CMTS to support SNMPv1, SNMPv2c vendor proprietary solutions, including SNMP v1/v2c NmAccess mode (using docsDevNmAccessTable).
- b) CMTS in SNMPv1/v2c NmAccess mode (using DOCS-CABLE-DEVICE-MIB docsDevNmAccess Table) MUST operate with the following requirements/limitations:
 - Only SNMPv1/v2c packets are processed
 - SNMPv3 packets are dropped

- docsDevNmAccessTable controls SNMP access and SNMP trap destinations as described in RFC-2669
 - None of the SNMPv3 MIBs as defined in [RFC-3411-3415] and [RFC-2576] are accessible.
- c) CMTS SNMPv1, SNMPv2c vendor proprietary solutions MUST operate with the following requirements/limitations:
- Only SNMPv1/v2c packets are processed
 - SNMPv3 packets are dropped
 - Vendor proprietary solution MUST control SNMP access and SNMP trap destinations
 - None of the SNMPv3 MIBs as defined in [RFC-3411-3415] and [RFC-2576] are accessible.
- d) CMTS SNMP Coexistence Mode MUST operate with the following requirements/limitations:
- SNMP v1/v2c/v3 Packets are processed as described by RFC-3411-3414 and RFC-2576.
 - docsDevNmAccessTable is not accessible. (If the CMTS also support DOCS-CABLE-DEVICE-MIB)
 - Access control and trap destinations are determined by the SNMP-COMMUNITY-MIB, SNMP-NOTIFICATION-MIB, SNMP-TARGET-MIB, SNMP-VIEW-BASED-ACM-MIB, SNMP-COMMUNITY-MIB, and SNMP-USER-BASED-SM-MIB.
 - The SNMP-COMMUNITY-MIB controls the translation of SNMPv1/v2c packet community string into securityName which select entries in the SNMP-USER-BASED-SM-MIB. Access control is provided by the SNMP-VIEW-BASED-ACM-MIB.
 - The SNMP-USER-BASED-SM-MIB and SNMP-VIEW-BASED-ACM-MIB control SNMPv3 packets.
 - Trap destinations are specified in the SNMP-TARGET-MIB and SNMP-NOTIFICATION-MIB.

2.1.1 Key Change Mechanism

DOCSIS 1.1 compliant CMTS SHOULD use the key-change mechanism specified in the RFC-2786. CMTS MUST always support the key-change mechanism described in the RFC-3414 to comply with industry-wide SNMP V3 standard.

2.2 SNMP Mode for DOCSIS 1.1 compliant CMs

DOCSIS 1.1 compliant CMs (in 1.1 and 1.0 mode) MUST support SNMPv1, SNMPv2c and SNMPv3 as well as SNMP-coexistence (RFC-2576) with the following requirements:

- a) Before completion of registration, the CM MUST operate as follows (in some CCCM implementations, SNMP access MAY be made inaccessible from the CPE for security reasons; in such implementation, the access to similar set of MIB objects SHOULD be provided by a diagnostic utility as described in section 6.3):
- IP connectivity between the CM and the SNMP management station MUST implemented as described in section 6.1
 - The CM MUST provide read-only access to the following MIB objects:
 - docsIfDownChannelFrequency
 - docsIfDownChannelPower
 - docsIfCmStatusValue
 - docsDevServerBootState
 - docsDevEventTable
 - The CM MAY provide read-only access to the following MIB objects
 - sysDescr
 - sysUptime
 - ifTable
 - ifXtable

docsIfUpChannelFrequency
 docsIfSigQSignalQualityTable
 docsIfCmCmtsAddress
 docsIfCmStatusTxPower
 docsDevSwCurrentVers

- The CM MAY provide access to additional information, but MUST NOT reveal:
 - CoS and QoS service flow information
 - configuration file contents
 - Secure Software Download information
 - Key authentication and encryption material
 - SNMP management and control
 - DOCSIS functional modules statistics and configuration
 - Network provisioning hosts and servers IPs addresses.
 - Access from the RF interface MUST NOT be allowed
 - SNMPv1/v2c packets are accepted which contain any community string
 - All SNMPv3 packets are dropped
 - The registration request MUST be sent and registration MUST be completed after successful processing of all MIB elements in the config file, but before beginning the calculation of the public values in the USMDHKickstart Table.
- b) The content of the CM config file determines the CM SNMP mode after registration
- CM is in SNMPv1/v2c docsDevNmAccess Mode if the CM configuration file contains ONLY docsDevNmAccessTable setting for SNMP access control.
 - If configuration file does not contain SNMP access control items (docsDevNmAccessTable or snmpCommunityTable or TLV 34.1/34.2 or TLV38), then the CM is in NmAccess mode.
 - CM is in SNMP coexistence mode if the CM configuration file contains
 - snmpCommunityTable setting and/or
 - TLV type 34.1 and 34.2. and/or
 - TLV type 38
 - In this case, any entries made to the docsDevNmAccessTable are ignored.
- c) After completion of registration - Modem operates in one of 2 modes. The operating mode is determined by the contents of the config file as described above.

SNMP V1/V2c NmAccess Mode (using docsDevNmAccess Table)

- Only SNMP V1/V2c packets are processed
- SNMP V3 packets are dropped
- docsDevNmAccessTable controls access and trap destinations as described in RFC-2669
- None of the SNMP V3 MIBs as defined in [RFC-3411-3415] and [RFC-2576] are accessible.

SNMP Coexistence Mode

During calculation of USMDHKickstartTable public values:

- The modem MUST NOT allow any SNMP access from the RF port
- The modem MAY continue to allow access from the CPE port with the limited access as configured by the SNMP-COMMUNITY-MIB, SNMP-TARGET-MIB, SNMP-VIEW-BASED-ACM-MIB and SNMP-USER-BASED-SM-MIB.

After calculation of USMDHKickstartTable public values:

- The modem MUST send the cold start or warm start trap to indicate that the modem is now fully SNMPv3 manageable.
- SNMP V1/V2c/V3 Packets are processed as described by RFC-3411-3415 and RFC-2576.

- docsDevNmAccessTable is not accessible.
 - Access control and trap destinations are determined by the SNMP-COMMUNITY-MIB, SNMP-NOTIFICATION-MIB, SNMP-TARGET-MIB, SNMP-VIEW-BASED-ACM-MIB, SNMP-COMMUNITY-MIB and SNMP-USER-BASED-SM-MIB.
 - The SNMP-COMMUNITY-MIB controls the translation of SNMPv1/v2c packet community string into security name which select entries in the SNMP-USER-BASED-SM-MIB. Access control is provided by the SNMP-VIEW-BASED-ACM-MIB.
 - SNMP-USER-BASED-SM-MIB and SNMP-VIEW-BASED-ACM-MIB controls SNMPv3 packets.
 - Trap destinations are specified in the SNMP-TARGET-MIB and SNMP-NOTIFICATION-MIB.
- d) In case of failure to complete SNMPv3 initialization (i.e. NMS can not access CM via SNMPv3 PDU), the CM is in the co-existence mode and will allow SNMPv1/v2c access if and only if the SNMP-COMMUNITY-MIB entries (and related entries) are configured.

2.2.1 SNMPv3 Initialization and Key changes

DOCSIS 1.1 compliant CM MUST support the “SNMPv3 Initialization” and “DH Key Changes” requirements specified in the following sections.

The DOCSIS 1.1 cable modem is designated as having "very-secure" security posture in the context of RFC-3414 Appendix A and RFC-3415 Appendix A. This means that default usmUser and vacmAccess entries defined in RFC-3414 Appendix A and RFC-3415 Appendix A MUST NOT be present.

2.2.2 SNMPv3 Initialization

1. For each of up to 5 different security names, the Manager generates a pair of numbers:

- a. Manager generates a random number R_m
- b. Manager uses DH equation to translate R_m to a public number z

$z = g^{R_m} \text{ MOD } p$ where g is the from the set of Diffie-Hellman parameters, p is the prime from those parameters

2. CM configuration file is created to include (security name, public number) pair and CM MUST support a minimum of 5 pairs. For example:

TLV type 34.1 (SnmpV3 Kickstart Security Name) = docsisManager

TLV type 34.2 (SnmpV3 Kickstart Public Number) = z

CM MUST support VACM entries defined in section 2.2.4 “VACM Profile”.

During the CM boot up process, the above values (security name, public number) will (MUST) be populated in the usmDHKickstartTable.

At this point:

usmDHKickstartMgrpublic.1 = “z” (octet string)
usmDHKickstartSecurityName.1 = “docsisManager”

When usmDHKickstartMgrpublic.n is set with a valid value during the registration, a corresponding row is created in the usmUserTable with the following values:

usmUserEngineID: localEngineID
usmUserName: usmDHKickstartSecurityName.n value
usmuserSecurityName: usmDHKickstartSecurityName.n value
usmUserCloneForm: ZeroDotZero
usmUserAuthProtocol: usmHMACMD5AuthProtocol
usmuserAuthKeyChange: derived from set value
usmUserOwnAuthKeyChange: derived from set value
usmUserPrivProtocol: usmDESPrivProtocol

usmUserPrivKeyChange: derived from set value
 usmUserOwnPrivKeyChange: derived from set value
 usmUserPublic: ""
 usmUserStorageType: permanent
 usmUserStatus: active

Note: For (CM) dhKickstart entries in usmUserTable, Permanent means it MUST be written to but not deleted and is not saved across reboots.

After the CM has registered with the CMTS.

CM generates a random number x_a for each row populated in the usmDhKickstartTable which has a non zero length usmDhKickstartSecurityName and usmDhKickstartMgrPublic.

CM uses DH equation to translate x_a to a public number c (for each row identified above)

$c = g^{x_a} \text{ MOD } p$ where g is the from the set of Diffie-Helman parameters, p is the prime from those parameters

At this point:

usmDhKickstartMyPublic.1 = "c" (octet string)
 usmDhKickstartMgrPublic.1 = "z" (octet string)
 usmDhKickstartSecurityName.1 = "docsisManager"

CM calculate shared secret sk where $sk = z^{x_a} \text{ mod } p$

CM uses sk to derive the privacy key and authentication key for each row in usmDhKickstartTable and sets the values into the usmUserTable.

As specified in RFC-2786, the privacy key and the authentication key for the associated username, "docsisManager" in this case, is derived from sk by applying the key derivation function PBKDF2 defined in PKCS#5v2.0.

```

privacy key <--- PBKDF2( salt = 0xd1310ba6,
                        iterationCount = 500,
                        keyLength = 16,
                        prf = id-hmacWithSHA1)
authentication key <---- PBKDF2( salt = 0x98dfb5ac,
                                iterationCount = 500,
                                keyLength = 16 (usmHMACMD5AuthProtocol),
                                prf = id-hmacWithSHA1)
  
```

At this point the CM has completed its SNMPv3 initialization process and MUST allow appropriate access level to a valid securityName with the correct authentication key and/or privacy key.

DOCSIS 1.1 compliant CM MUST properly populate keys to appropriate tables as specified by the SNMPv3 related RFCs and RFC-2786.

The following describes the process that the manager uses to derive CM's unique authentication key and privacy key.

The SNMP manager accesses the contents of the usmDhKickstartTable using the security name of 'dhKickstart' with no authentication.

DOCSIS 1.1 compliant CM MUST provide preinstalled entries in the USM table and VACM tables to correctly create user 'dhKickstart' of security level noAuthnoPriv that has read only access to system group and usmDhkickstartTable.

SNMP manager gets the value of CM's usmDhKickstartMypublic number associated with the security name that manager wants to derive authentication and privacy keys for. With the manager's knowledge of the private random number, the manager can calculate the DH shared secret. From that shared secret, the manager can derive operational authentication and confidentiality keys for the security name that the manager is going to use to communicate with the CM.

2.2.3 DH Key Changes

DOCSIS 1.1 compliant CM MUST support the key-change mechanism specified in the RFC-2786.

2.2.4 VACM Profile

This section will address the default VACM profile for DOCSIS CM when it is operating in SNMP Coexistence mode.

The following VACM entries **MUST** be included by default in a compliant CM:

- The system manager, with full read/write/config access
 - vacmSecurityModel: 3 (USM)
 - vacmSecurityName:
 - docsisManager
 - vacmGroupName: docsisManager
 - vacmSecurityToGroupStorageType: permanent
 - vacmSecurityToGroupStatus: active
- An operator/CSR with read/reset access to full modem
 - vacmSecurityModel: 3 (USM)
 - RF Monitoring with read access to RF plant statistics
 - vacmSecurityModel: 3 (USM)
 - vacmSecurityName: docsisMonitor
 - vacmGroupName: docsisMonitor
 - vacmSecurityToGroupStorageType: permanent
 - vacmSecurityToGroupStatus: active
- User debugging with read access to "useful" variables
 - vacmSecurityModel: 3 (USM)
 - vacmSecurityName: docsisUser
 - vacmGroupName: docsisUser
 - vacmSecurityToGroupStorageType: permanent
 - vacmSecurityToGroupStatus: active
- Group name to view translations
 - vacmGroupName: docsisManager
 - vacmAccessContextPrefix:
 - vacmAccessSecurityModel: 3 (USM)
 - vacmAccessSecurityLevel: AuthPriv
 - vacmAccessContextMatch: exact
 - vacmAccessReadViewName: docsisManagerView
 - vacmAccessWriteViewName: docsisManagerView
 - vacmAccessNotifyViewName: docsisManagerView
 - vacmAccessStorageType: permanent
 - vacmAccessStatus: active

 - vacmGroupName: docsisOperator
 - vacmAccessContextPrefix:
 - vacmAccessSecurityModel: 3 (USM)
 - vacmAccessSecurityLevel: AuthPriv & AuthNoPriv
 - vacmAccessContextMatch: exact
 - vacmAccessReadViewName: docsisManagerView
 - vacmAccessWriteViewName: docsisOperatorWriteView
 - vacmAccessNotifyViewName: docsisManagerView
 - vacmAccessStorageType: permanent
 - vacmAccessStatus: active

 - vacmGroupName: docsisMonitor
 - vacmAccessContextPrefix:
 - vacmAccessSecurityModel: 3 (USM)

vacmAccessSecurityLevel: AuthNoPriv
 vacmAccessContextMatch: exact
 vacmAccessReadViewName: docsisMonitorView
 vacmAccessWriteViewName:
 vacmAccessNotifyViewName: docsisMonitorView
 vacmAccessStorageType: permanent
 vacmAccessStatus: active

vacmGroupName: docsisUser
 vacmAccessContextPrefix:
 vacmAccessSecurityModel: 3 (USM)
 vacmAccessSecurityLevel: AuthNoPriv
 vacmAccessContextMatch: exact
 vacmAccessReadViewName: docsisUserView
 vacmAccessWriteViewName:
 vacmAccessNotifyViewName:
 vacmAccessStorageType: permanent
 vacmAccessStatus: active

- The views

docsisManagerView

subtree: 1.3.6.1 (Entire mib).

docsisOperatorWriteView

subtree: docsDevBase
 subtree: docsDevSoftware
 subtree: docsDevEvControl
 subtree: docsDevEvThrottleAdminStatus

docsisMonitorView

subtree: 1.3.6.1.2.1.1 (system)
 subtree: docsIfBaseObjects
 subtree: docsIfCmObjects

docsisUserView

subtree: 1.3.6.1.2.1.1 (system)
 subtree: docsDevBase
 subtree: docsDevSwOperStatus
 subtree: docsDevSwCurrentVersion
 subtree: docsDevServerConfigFile
 subtree: docsDevEventTable
 subtree: docsDevCpeTable
 subtree: docsIfUpstreamChannelTable
 subtree: docsIfDownstreamChannelTable
 subtree: docsIfSignalQualityTable
 subtree: docsIfCmStatusTable

DOCSIS 1.1 compliant CM MUST also support additional VACM users as they are configured via an SNMP-embedded configuration file.

3 Management Information Bases (MIBs)

This section defines the minimum set of managed objects required to support the management of CM and CMTS. Vendors MAY augment this MIB with objects from other standard or vendor-specific MIBs where appropriate.

DOCSIS OSSI 1.1 specification has priority over IETF MIB specification. Vendor MUST implement MIB requirements in accordance with the texts specified in OSSI 1.1 specification. Certain objects are deprecated or obsolete but may be required by the OSSI specification as mandatory and MUST be implemented.

Deprecated objects are optional. That is, a vendor can choose to implement or not implement the object. If a vendor chooses to implement the object, the object MUST be implemented correctly according to the MIB definition. If a vendor chooses not to implement the object, an agent MUST NOT instantiate such object and MUST respond with the appropriate error/exception condition. (e.g., no such object for SNMPv2c)

Optional object. A vendor can choose to implement or not implement the object. If a vendor chooses to implement the object, the object MUST be implemented correctly according to the MIB definition. If a vendor chooses not to implement the object, an agent MUST NOT instantiate such object and MUST respond with the appropriate error/exception condition. (e.g., no such object for SNMPv2c)

Obsolete object. It is optional. A vendor can choose to implement or not implement the object. If a vendor chooses to implement the object, the object MUST be implemented correctly according to the MIB definition. If a vendor chooses not to implement the object, an agent MUST NOT instantiate such object and MUST respond with the appropriate error/exception condition. (e.g., no such object for SNMPv2c)

Section 3.1 and 3.2 include an overview of the MIB modules required for the management of the facilities specified in SP-RFI-1.1 and BPI+ specifications.

3.1 IPCDN Drafts and Others

Table 1. IPCDN Drafts

REFERENCE	MIB	Applicable Device(s)
[IETF4]	IETF Proposed Standard RFC-version of Qos MIB, "draft-ietf-ipcdn-qos-mib-04.txt" DOCS-QOS-MIB	CM and CMTS
[IETF6]	IETF Proposed Standard RFC-version of BPI+ MIB, "draft-ietf-ipcdn-bpiplus-mib-05.txt" DOCS-BPI2-MIB	CM and CMTS
[IETF7]	IETF Proposed Standard RFC-version of USB MIB, "dolnik-usb-mib-00.txt" USB-MIB	CM only
[IETF9]	IETF Proposed Standard RFC-version of Subscriber Management MIB, "draft-ietf-ipcdn-subscriber-mib-02.txt" DOCS-SUBMGT-MIB	CMTS only
[IETF11]	IETF Proposed Standard RFC-version of RF MIB, "draft-ietf-ipcdn-docs-rfmibv2-05.txt" DOCS-IF-MIB	CM and CMTS

3.2 IETF RFCs

Table 2. IETF RFCs

REFERENCE	MIB	Applicable Device(s)
[RFC-2669]	DOCSIS Cable Device MIB: DOCS-CABLE-DEVICE-MIB	CM and CMTS
[RFC-3083]	Baseline Privacy Interface MIB: DOCS-BPI-MIB	CM
[RFC-2933]	Internet Group Management Protocol MIB: IGMP-STD-MIB	CM and CMTS
[RFC-2863]	The Interfaces Group MIB using SMIv2: IF-MIB	CM and CMTS
[RFC-2665]	Ethernet Interface MIB: EtherLike-MIB	CM and CMTS
[RFC-1493]	Bridge MIB: BRIDGE-MIB	CM and CMTS
[RFC-2011]	SNMPv2 Management Information Base for the Internet Protocol using SMIv2: IP-MIB	CM and CMTS
[RFC-2013]	Management Information Base for the User Datagram Protocol using SMIv2: UDP-MIB	CM and CMTS
[RFC-3418]	Management Information Base (MIB) for the Simple Network Management Protocol (SNMP): SNMPv2-MIB	CM and CMTS
[RFC-3410] [RFC-3411] [RFC-3412] [RFC-3413] [RFC-3414] [RFC-3415] [RFC-2576]	SNMP v3 MIBs: SNMP-FRAMEWORK-MIB, SNMP-MPD-MIB, SNMP-NOTIFICATION-MIB, SNMP-TARGET-MIB, SNMP-USER-BASED-SM-MIB, SNMP-VIEW-BASED-ACM-MIB, SNMP-COMMUNITY-MIB	CM and CMTS
[RFC-2786]	RFC-2786: Diffie-Helman USM Key: SNMP-USM-DH-OBJECTS-MIB	CM and CMTS

3.3 Managed Objects Requirements

The following sections detail any additional implementation requirements for the RFCs listed. Reference Appendix A for specific object implementation requirements.

The CM and CMTS MUST support a minimum of 10 available SNMP Table Rows unless otherwise specified by RFC or DOCSIS specification. The CM/CMTS minimum number of available SNMP Table Rows SHOULD mean rows (per table) that are available to support device configuration. CM/CMTS used (default) SNMP Table Row entries MUST NOT apply to the minimum number of available SNMP Table Rows.

3.3.1 CMTS MIB requirements

DOCSIS 1.1 compliant CMTS MUST implement Subscribe Management MIB.

3.3.2 Requirements for RFC-2669

RFC-2669 MUST be implemented by DOCSIS 1.1 compliant CMs. DOCSIS 1.1 compliant CMTS MUST implement mandatory required objects (as specified by Appendix A), and SHOULD implement the other non-mandatory required objects.

3.3.3 Requirements for DOCS-IF-MIB

The DOCS-IF-MIB MUST be implemented by DOCSIS 1.1 compliant CMTS and CMs.

The docsIfDownChannelPower object-type MUST be implemented in a CMTS that provides an integrated RF upconverter. If the CMTS relies on an external upconverter, then the CMTS SHOULD implement the docsIfDownChannelPower object-type. The CMTS transmit power reported in the MIB object MUST be within 2 dB of the actual transmit power in dBmV when implemented. If transmit power management is not implemented, the MIB object will be read-only and report the value of 0 (zero).

The docsIfDownChannelPower object-type MUST be implemented in DOCSIS 1.1 conforming CM's. This object is read-only. When operated at nominal line voltage, at normal room temperature, the reported power MUST be within 3 dB of the actual received channel power. Across the input power range from -15 dBmV to +15 dBmV, for any 1 dB change in input power, the CM MUST report a power change in the same direction that is not less than 0.5 dB. and not more than 1.5 dB.

The access of docsIfDownChannelFrequency object MUST be implemented as RW if a CMTS is in control of the downstream frequency. But if a CMTS provides IF output, docsIfDownChannelFrequency MUST be implemented as read-only and return 0.

All objects added as a result of the DOCS-IF-MIB upgrade from RFC2670 to draft-ietf-ipcdn-docs-rfmibv2-05.txt are optional for DOCSIS 1.1 devices, with the exception of objects 'transferred' from the docsIfExt MIB, and objects indicating the CM modulation type. These objects are mandatory for DOCSIS 1.1 devices, and include docsIfDocsisBaseCapability, docsIfCmStatusDocsisOperMode, docsIfCmStatusModulationType, docsIfCmtsCmStatusDocsisRegMode, and docsIfCmtsCmStatusModulationType. docsIfCmtsChannelUtilizationTable, docsIfCmtsDownChannelCounterTable, and only the first nine MIB objects of docsIfCmtsUpChannelCounterTable. Refer to Appendix A for details on optional/mandatory status of new DOCS-IF-MIB objects.

“The docsIfQosProfMaxTransmitBurst range MUST be the same as the one defined in the RFIv1.1 specification, section C.1.1.4.6 “Maximum Upstream Channel Transmit Burst Configuration Setting” which has range 0 to 65535.”

If the CMTS implements the docsIfUpChannelStatus object-type, the CMTS MUST NOT allow it to be set from active(1) directly or indirectly to destroy(6). The CMTS MUST return a wrongValue error. Entries with docsIfUpChannelStatus set to active(1) are logically linked to a physical interface, not temporarily created to clone parameters.

3.3.4 Requirements for RFC-2863

RFC-2863 MUST be implemented by DOCSIS 1.1 compliant CMTS and CMs.

The CMTS/CM ifAdminStatus object MUST provide administrative control over both MAC interfaces and individual channel and MUST be implemented as RW.

The ifType object has been assigned the following enumerated values for each instance of a Data Over Cable Service (DOCS) interface:

CATV MAC interface:: docsCableMacLayer (127)

CATV downstream channel: docsCableDownstream (128)

CATV upstream channel:: docsCableUpStream (129)

3.3.4.1 Interface Organization and Numbering

Assigned interface numbers for CATV-MAC and Ethernet (Ethernet-like interface) are used in both the NMAccessTable and IP/LLC filtering table to configure access and traffic policy at these interfaces. These configurations are generally encoded in the configuration file using TLV encoding. To avoid provisioning complexity the interface-numbering scheme MUST comply with the following requirements:

An instance of IfEntry MUST exist for each CATV-MAC interface, downstream channel, upstream channel, and each LAN interface enabled by the CM. The enablements of LAN interfaces MAY be fixed a priori during manufacturing process or MAY be determined dynamically during operation by the CM according to if an interface has a CPE device attached to it or not.

If the CM has multiple CPE interfaces but only one CPE interface can be enabled at any given time, then the ifTable MUST only contain the entry corresponding to the enabled or the default CPE interface. If a MAC interface consists of more than one upstream and downstream channel, then a separate instance of ifEntry MUST also exist for each channel.

The ifStack group ([RFC-2863]) must be implemented to identify relationship among sub-interfaces. Note that the CATV-MAC interface MUST exist, even though it is broken out into sub-interfaces.

The example below illustrates a MAC interface with one downstream and two upstream channels for a CMTS.

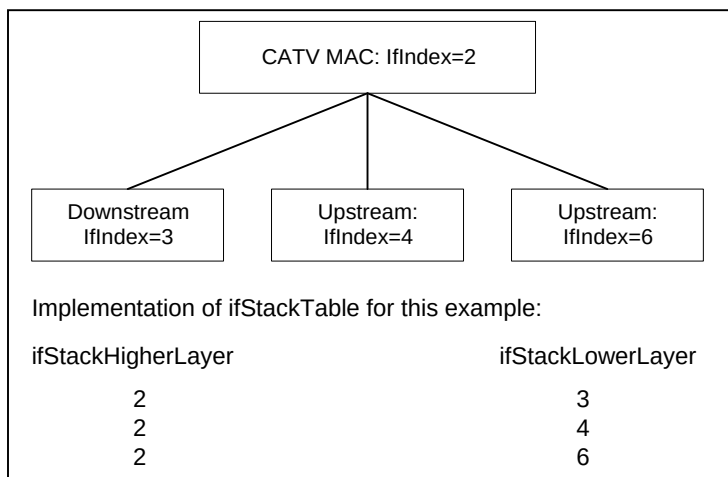


Figure 1. Ifindex Example for CMTS

At the CMTS, interface number is at the discretion of the vendor, and SHOULD correspond to the physical arrangement of connections. If table entries exist separately for upstream and downstream channels, then the ifStack group ([RFC-2863]) MUST be implemented to identify the relationship among sub-interfaces. Note that the CATV MAC interface(s) MUST exist, even if further broken out into sub-interfaces.

At the CM, interface MUST be numbered as:

Table 3. CM Interface numbering

Interfaces	Type
1	primary CPE interface
2	CATV-MAC
3	RF-down
4	RF-Up
5 – 15, 32+n	Other interfaces
16 - 31	Other interfaces (Reserved)

If CM has more than one CPE interface, then the vendor **MUST** define which of (n) CPE interfaces is the primary CPE interface. The definition of the primary CPE interface **MAY** be fixed a prior during manufacturing process or **MAY** be determined dynamically during operation by the CM according to which interface has a CPE device attached to it. Regardless how many CPE interfaces the CM has or how the primary CPE interface is defined, the primary interface **MUST** be interface number 1.

The definition of the secondary CPE interface **MAY** be fixed a prior during manufacturing process or **MAY** be determined dynamically during operation by the CM according to which interface has a CPE device attached to it. The secondary CPE, and other interfaces, will start at 5.

DOCSIS CM may have multiple interfaces. If filter(s) (Ip, LLC, or NmAccess) are applied to CM IfIndex 1, the same filter(s) **MUST** also be applied to the "Other interfaces" (IfIndexes 5 and above); however, filters are never used to limit traffic between the CPE and "Other" interfaces within the CM.

3.3.4.2 docsIfCmStatusValue and ifOperStatus Relationship

For CM RF downstream, RF upstream and RF MAC interfaces; the following are the expected relationship of ifOperStatus and docsIfCmStatusValue when ifAdminStatus = up (taken from DOCS-IF-MIB).

Table 4. docsIfCmStatusValue and ifOperStatus Relationship

ifOperStatus	docsIfCmStatusValue
down(2):	other(1), notReady(2)
dormant(5):	notSynchronized(3), phySynchronized(4), usParametersAcquired(5), rangingComplete(6), ipCompleet(7), todEstablished(8), paramTransferComplete(10), accessDenied(13)
up(1):	registrationComplete(11), securityEstablished(9), operational(12)

3.3.4.2.1 ifOperStatus and traffic

If the CM and CMTS interface's ifAdminStatus = down, the interface **MUST** not accept or forward any traffic (traffic includes data and MAC management traffic).

3.3.5 Interface MIB and Trap Enable

Interface MIB and Trap Enable specified in RFC-2863 **MUST** be implemented by DOCSIS 1.1 compliant CMTS and CMs.

If a multi-layer interface model is present in the device, each sub-layer for which there is an entry in the ifTable can generate linkUp/Down traps. Since interface state changes would tend to propagate through the interface stack (from top to bottom, or bottom to top), it is likely that several traps would be generated for each linkUp/Down occurrence. The CM and CMTS **MUST** implement the ifLinkUpDownTrapEnable object to allow managers to control trap generation, and configure only the interface sub-layers of interest.

The default setting of ifLinkUpDownTrapEnable **MUST** limit the number of traps generated to one, per interface, per linkUp/Down event. Interface state changes, of most interest to network managers, occur at the lowest level of an interface stack.

On CM linkUp/Down event a trap **SHOULD** be generated by the CM MAC interface and not by any sub-layers of the interface. Therefore, the default setting of ifLinkUpDownTrapEnable for CM MAC **MUST** be set to enable, and the default setting of ifLinkUpDownTrapEnable for CM RF-Up **MUST** be set to disable, and the default setting of ifLinkUpDownTrapEnable for CM RF-Down **MUST** be set to disable.

On CMTS interfaces (MAC, RF-Downstream(s), RF-Upstream(s)) the linkUp/Down event/trap **SHOULD** be generated by each CMTS interface. Therefore, the default setting of ifLinkUpDownTrapEnable for each CMTS interface (MAC, RF-Downstream(s), RF-Upstream(s)) **MUST** be set to enable.

3.3.6 Requirements for RFC-2665

RFC-2665 MUST be implemented by DOCSIS 1.1 compliant CMTS and CM if Ethernet or Fast Ethernet interfaces are present.

3.3.7 Requirements for RFC-1493

RFC-1493 MUST be implemented by DOCSIS 1.1 compliant CMTS and CMs.

In both the CM and the CMTS (if the CMTS implements transparent bridging), the Bridge MIB ([RFC-1493]) MUST be implemented to manage the bridging process.

In the CMTS that implements transparent bridging, the Bridge MIB MUST be used to represent information about the MAC Forwarder states.

3.3.8 Requirements for RFC-2011

RFC-2011 MUST be implemented by DOCSIS 1.1 compliant CMTS and CMs.

3.3.8.1 The IP Group

The IP group MUST be implemented. It does not apply to IP packets forwarded by the device as a link-layer bridge. For the CM, it applies only to the device as an IP host. At the CMTS, it applies to the device as an IP host, and as a routers if IP routing is implemented.

3.3.8.2 The ICMP Group

The ICMP group MUST be implemented. It does not apply to IP packets forwarded by the device as a link-layer bridge. For the CM, it applies only to the device as an IP host. At the CMTS, it applies to the device as an IP host, and as a routers if IP routing is implemented.

Since CMs do not generate ICMP requests and do not support ICMP Timestamps, Table 24. Detailed MIB Requirements, lists MIB objects that are optional.

3.3.9 Requirements for RFC-2013

RFC-2013 MUST be implemented by DOCSIS 1.1 compliant CMTS and CMs. The UDP group does not apply to IP packets forwarded by the device as a link-layer bridge. For the CM, it applied only to the device an IP host. At the CMTS, it applies to the device only as an IP host.

3.3.10 Requirements for RFC-3418

RFC-3418 MUST be implemented by DOCSIS 1.1 compliant CMTS and CMs.

3.3.10.1 The System Group

The System Group from RFC-3418 MUST be implemented.

3.3.10.2 The SNMP Group

The SNMP Group from RFC-3418 MUST be implemented.

3.3.11 Requirements for DOCS-QOS-MIB

“draft-ietf-ipcdn-qos-mib-04.txt” MUST be implemented by DOCSIS 1.1 compliant CMTS and CMs

The default values for the MIB objects in docsQosParamSetTable and docsQosServiceClassTable MUST follow the referenced ones in the RFIv1.1 specification. For example, docsQosParamSetMaxTrafficBurst default value is 3044 (which is 1522 * 2), docsQosServiceClassMaxTrafficBurst DEFVAL is 3044, docsQosParamSetMaxConcatBurst default value is 1522, and docsQosServiceClassMaxConcatBurst DEFVAL is 1522. If in the future, there are any related default values changed in the RFIv1.1 specification, the related default values in DOCS-QOS-MIB docsQosParamSetTable and docsQosServiceClassTable MUST be changed accordingly even though the MIB file is not changed in time.

3.3.12 Requirements for “draft-ietf-ipcdn-igmp-mib-01.txt”

“draft-ietf-ipcdn-igmp-mib-01.txt” requirements have been deleted for CMTS and CMs.

3.3.13 Requirements for RFC-2933

RFC-2933 MUST be implemented by DOCSIS 1.1 compliant CMTS and CMs.

Refer to “Appendix J, “Application of RFC-2933 to DOCSIS 1.1 active/passive IGMP devices” for DOCSIS 1.1 IGMP cable device implementation details.

3.3.14 Requirements for DOCS-BPI2-MIB

“draft-ietf-ipcdn-bpiplus-mib-0705.txt” MUST be implemented by DOCSIS 1.1 compliant CMTS and CM as specified in Appendix A.

3.3.15 Requirements for USB-MIB

See Appendix O.

3.3.16 Requirements for DOCS-SUBMGT-MIB

“draft-ietf-ipcdn-subscriber-mib-02-.txt” MUST be implemented by DOCSIS 1.1 compliant CMTS.

DOCSIS 1.1 compliant CMTS MUST support a minimum number of filter groups; (30) thirty groups of (20) twenty filters each.

3.3.17 Requirements for RFC-2786

RFC-2786 MUST be implemented by DOCSIS 1.1 compliant CMs. It (RFC-2786) MAY be implemented on the CMTS.

3.3.18 Requirements for RFC-3083

RFC-3083 MUST be implemented by DOCSIS 1.1 compliant CMs as specified in Appendix A.

Due to the editorial error in RFC-3083, the DOCSIS 1.1 compliant CM MUST use the following definition for docsBpiCmAuthState and not the definition in RFC-3083.

```
docsBpiCmAuthState      OBJECT-TYPE
SYNTAX      INTEGER {
                    start(1),
                    authWait(2),
                    authorized(3),
                    reauthWait(4),
                    authRejectWait(5)
                }
MAX-ACCESS   read-only

STATUS       current
```

DESCRIPTION

"The value of this object is the state of the CM authorization FSM. The start state indicates that FSM is in its initial state."

REFERENCE

"DOCSIS Baseline Privacy Interface Specification, Section 4.1.2.1."
 ::= {docsBpiCmBaseEntry 3 }

In addition, compliant CMs MAY create new entries in the docsBpiCmTEKTable for any multicast SID(s) it receives in Auth-Reply messages. If implemented, the multicast SID MUST be used as an index in the docsBpiCmTEKTable in the docsIfCmServiceId field. Note that if the multicast SID is used in the docsBpiCmTEKTable, there MUST NOT be a corresponding entry in the docsIfCmServiceTable for the multicast SID, due to the definition of the docsIfCmService ID in the DOCS-IF-MIB.

3.3.19 Requirement for DOCS-IF-EXT-MIB

A DOCSIS 1.1 compliant CM/CMTS MAY support the DOCS-IF-EXT MIB, which is defined in Appendix L. If a DOCSIS 1.1 CM/CMTS supports the deprecated docsIfExt MIB objects in the docsCableDevice MIB trap definitions, then it MUST also support the DOCS-IF-EXT MIB.

3.3.20 Requirements for DOCS-CABLE-DEVICE-TRAP-MIB

DOCSIS 1.1 compliant CM/CMTS must implement DOCS-CABLE-DEVICE-TRAP-MIB, as specified in Appendix I.

3.3.21 Requirements for SNMPv3 MIBs

DOCSIS 1.1 compliant CM/CMTS MUST implement the MIBs defined in RFC 3411-3415 and RFC 2576.

For CMs, the default value for any SNMPv3 object with a storageType textual convention MUST be 'volatile'. This overrides the default value specified in RFC 3413-3415 and RFC 2576.

The CM MUST only accept the value of 'volatile' on any SNMPv3 storageType object.

An attempted set to a value of other(1), nonVolatile(3), permanent(4), or readOnly(5) will result an 'inconsistentValue' error. Values other than the valid range (1-5) would result a 'wrongValue' error.

The CM and CMTS SHOULD support a minimum of 30 available rows in the vacmViewTreeFamilyTable object.

3.4 CM Configuration Files, TLV-11 and MIB OIDs/Values

The following sections define the use of CM configuration file TLV-11 elements and the CM rules for translating TLV-11 elements into SNMP PDU (SNMP MIB OID/instance and MIB OID/instance value combinations; also referred to as SNMP varbinds).

This section also defines the CM behaviors, or state transitions, after either pass or fail of the CM configuration process.

For TLV-11 definitions refer to [DOCSIS 5; Appendix C].

3.4.1 CM configuration file TLV-11 element translation (to SNMP PDU)

TLV-11 translation defines the process used by CM to convert CM configuration file information (TLV-11 elements) into SNMP PDU (varbinds). The CM MUST translate CM configuration file TLV-11 elements into a single SNMP PDU containing (n) MIB OID/instance and value components (SNMP varbinds). Once a single SNMP PDU is constructed, the CM will process the SNMP PDU and determine CM configuration pass/fail based on the rules for CM configuration file processing, described below. However, if a CM is not physically capable of processing a, potentially large, single CM configuration file generated SNMP PDU, then the CM must still behave as if all MIB OID/instance and value components (SNMP varbinds), from CM configuration file TLV-11 elements, are processed as a single SNMP PDU.

In accordance with [RFC-3416], the single CM configuration file generated SNMP PDU will be treated “as if simultaneous” and the CM must behave consistently, regardless of the order in which TLV-11 elements appear in the CM configuration file, or SNMP PDU. The singular CM configuration file generated SNMP PDU requirement is consistent with SNMP PDU packet behaviors, received from an SNMP manager; SNMP PDU varbind order does not matter, and there is no defined MAX SNMP PDU limit.

The CM configuration file MUST NOT contain duplicate TLV-11 elements (duplicate means SNMP MIB object has either identical OID or OID from the old and new MIB that actually point to the same SNMP MIB object). If duplicate TLV-11 elements are received by the CM, from the CM configuration file, then the CM MUST fail CM configuration.

3.4.1.1 Rules for CreateAndGo and CreateAndWait

The CM MUST support CreateAndGo for row creation.

The CM MAY support CreateAndWait; with the constraint that CM configuration file TLV-11 elements MUST NOT be duplicated (all SNMP MIB OID/instance must be unique). For instance, an SNMP PDU, constructed from CM configuration file TLV-11 elements, which contains an SNMP CreateAndWait value, for a given SNMP MIB OID/instance, MUST NOT also contain an SNMP Active value for the same SNMP MIB OID/instance (and vice versa). A CM configuration file MAY contain a TLV-11 CreateAndWait element if the intended result is to create an SNMP table row which will remain in the SNMP NotReady or SNMP NotInService state until a non-configuration file SNMP PDU is issued, from an SNMP manager, to update the SNMP table row status.

Both SNMP NotReady and SNMP NotInService states are valid table row states after an SNMP CreateAndWait instruction.

3.4.2 Ignore CM configuration TLV-11 elements which are not supported by CM

If any CM configuration file TLV-11 elements translate to SNMP MIB OIDs that are not MIB OID elements supported by the CM, then those SNMP varbinds MUST be ignored, and treated as if they had not been present, for the purpose of CM configuration. This means that the CM will ignore SNMP MIB OIDs for other vendor's private MIBs as well as standard MIB elements that the CM does not support.

CMs that do not support SNMP CreateAndWait for a given SNMP MIB table MUST ignore, and treated as if not present, the set of columns associated with the SNMP table row.

If any CM configuration file TLV-11 element(s) are ignored, then the CM MUST report via the CM configured notification mechanism(s), after the CM is registered. The CM notification method MUST be in accordance with the “Standard DOCSIS event” section, defined within this document.

3.4.3 CM state after CM configuration file processing success

After successful CM configuration, via CM configuration file, CM MUST proceed to register, with CMTS, and pass data.

3.4.4 CM state after CM configuration file processing failure

If any CM configuration file generated SNMP PDU varbind performs an illegal set operation (illegal, bad, or inconsistent value) to any MIB OID/instance supported by the CM, then processing of the CM configuration file MUST fail. Any CM configuration file generated SNMP PDU varbind set failure MUST cause a CM configuration failure, and the CM MUST NOT proceed with CM registration.

3.5 Treatment and Interpretation of MIB Counters on the CM

Octet and packet counters implemented as counter32 and counter64 MIB objects are defined to be monotonically increasing positive integers with no specific initial value and a maximum value based on the counter size that will roll-over to zero when it is exceeded. In particular, counters are defined such that the only meaningful value is the difference between counter values as seen over a sequence of counter polls. However there are two situations that can cause this consistent monotonically increasing behavior to change: 1) resetting the counter due to a system or interface reinitialization or 2) a rollover of the counter when it reaches its maximum value of $2^{32}-1$ or $2^{64}-1$. In these situations, it must be clear what the expected behavior of the counters should be.

Case 1: Whenever the state of an interface changes resulting in an “interface counter discontinuity” as defined in RFC-2863. In this case the value of the ifXTable.ifXEntry.ifCounterDiscontinuityTime for the affected interface MUST be set to the current value of sysUpTime and ALL counters for the affected interface MUST be set to ZERO. Setting the ifAdminStatus of specified interface to down(2) MUST NOT be considered as an interface reset.

Case 2: SNMP Agent Reset. In this case, the value of the sysUpTime MUST be set to ZERO, all interface ifCounterDiscontinuityTime values MUST be set to ZERO, and all interface counters MUST be set to ZERO. Also, all other counters being maintained by the SNMP Agent MUST be set to ZERO.

Case 3: Counter Rollover. When a counter32 object reaches its maximum value of 4,294,967,295 the next value MUST be ZERO. When a counter64 object reaches its maximum value of 18,446,744,073,709,551,615 the next value MUST be ZERO. Note that unless a CM or CMTS vendor provides a means outside of SNMP to preset a counter64 or counter32 object to an arbitrary value, it will not be possible to test any rollover scenarios for counter64 objects (and many counter32 objects as well). This is because it is not possible for these counters to rollover during the service life of the device (see discussion in RFC-2863 section 3.1.6).

3.6 Config File Element – SNMP V3Notification Receiver

The following sections detail the CM Configuration File TLV-38 “DOCSIS V3 Notification Receiver” mapping into SNMP V3 functional tables. A CM MUST support a minimum of 10 TLV-38 elements in a configuration file. For TLV-38 definitions refer to [DOCSIS 5; Appendix C].

Upon receiving one TLV 38, the CM MUST make entries to the following tables in order to cause the desired trap transmission: snmpNotifyTable, snmpTargetAddrTable, snmpTargetAddrExtTable, snmpTargetParamsTable, snmpNotifyFilterProfileTable, snmpNotifyFilterTable, nmpCommunityTable, usmUserTable, vacmSecurityToGroupTable, vacmAccessTable, and vacmViewTreeFamilyTable.

A config file MAY also contain TLV MIB elements that make entries to any of the 10 tables listed above. These TLV MIB elements MUST NOT use index columns that start with the characters “@config”.

3.6.1 Mapping of TLV fields into created SNMP V3 Table rows

The tables in this section show how the fields from the Config file TLV element (the tags in angle brackets <>) are placed into the SNMP V3 tables.

The correspondence between TLV fields and table tags <TAG> is shown below:

<IP Address>	TLV 38.1
<Port> -	TLV 38.2
<Trap type>	TLV 38.3
<Timeout>	TLV 38.4
<Retries>	TLV 38.5
<Filter OID>	TLV 38.6
<Security Name>	TLV 38.7

These tables are shown in the order that the agent will search down through them when a notification is generated in order to determine who to send the notification to and how to fill out the contents of the notification packet.

3.6.1.1 snmpNotifyTable

Create 2 rows with fixed values, if 1 or more TLV elements are present

Table 5. snmpNotifyTable

snmpNotifyTable (RFC-2573 - SNMP-NOTIFICATION-MIB)	1st Row	2nd Row
Column Name (* = Part of Index)	Column Value	Column Value
* snmpNotifyName	"@config_inform"	"@config_trap"
snmpNotifyTag	"@config_inform"	"@config_trap "
snmpNotifyType	inform (2)	trap (1)
snmpNotifyStorageType	volatile	volatile
snmpNotifyRowStatus	Active (1)	Active (1)

3.6.1.2 snmpTargetAddrTable

Create 1 row for each TLV element in the config file

Table 6. snmpTargetAddrTable

snmpTargetAddrTable (RFC-2573 - SNMP-TARGET-MIB)	New Row
Column Name (* = Part of Index)	Column Value
* snmpTargetAddrName	"@config_n" Where n ranges from 0 to m-1 where m is the number of notification receiver TLV elements in the config file
snmpTargetAddrTDomain	snmpUDPDDomain = snmpDomains.1
snmpTargetAddrTAddress (IP Address and UDP Port of the Notification Receiver)	OCTET STRING (6) Octets 1-4: <IP Address> Octets 5-6: <Port>
snmpTargetAddrTimeout	<Timeout> from the TLV
snmpTargetAddrRetryCount	<Retries> from the TLV
snmpTargetAddrTagList	If <Trap type> == 1, 2 or 4 "@config_trap" Else If <Trap type> = 3 or 5 "@config_inform"
snmpTargetAddrParams	"@config_n" (Same as snmpTargetAddrName value)
snmpTargetAddrStorageType	volatile
snmpTargetAddrRowStatus	active (1)

3.6.1.3 snmpTargetAddrExtTable

Create 1 row for each TLV element in the config file

Table 7. snmpTargetAddrExtTable

snmpTargetAddrExtTable (RFC-2576 - SNMP-COMMUNITY-MIB)	New Row
Column Name (* = Part of Index)	Column Value
* snmpTargetAddrName	"@config_n" Where n ranges from 0 to m-1 where m is the number of notification receiver TLV elements in the config file
snmpTargetAddrTMask	<Zero length octet string>
snmpTargetAddrMMS	0

3.6.1.4 snmpTargetParamsTable

Create 1 row for each TLV element in the config file. If <Trap type> is 1, 2, or 3, or if the <Security Name> Field is zero-length, create the table as follows:

Table 8. snmpTargetParamsTable for <Trap type> 1, 2, or 3

snmpTargetParamsTable (RFC-2573 - SNMP-TARGET-MIB)	New Row
Column Name (* = Part of Index)	Column Value
* snmpTargetParamsName	"@config_n" Where n ranges from 0 to m-1 where m is the number of notification receiver TLV elements in the config file
snmpTargetParamsMPModel SYNTAX: SnmmpMessageProcessingModel	If <Trap type> = 1 SNMPv1 (0) Else If <Trap type> = 2 or 3 SNMPv2c (1) Else if <Trap type> = 4 or 5 SNMPv3 (3)
snmpTargetParamsSecurityModel SYNTAX: SnmpSecurityModel	If <Trap type> = 1 SNMPv1 (1) Else If <Trap type> = 2 or 3 SNMPv2c (2) Else if <Trap type> = 4 or 5 USM (3) NOTE: The mapping of SNMP protocol types to value here are different from snmpTargetParamsMPModel
snmpTargetParamsSecurityName	"@config"
snmpTargetParamsSecurityLevel	noAuthNoPriv
snmpTargetParamsStorageType	volatile
snmpTargetParamsRowStatus	active (1)

If <Trap type> is 4 or 5, and the <Security Name> Field is non-zero length, create the table as follows:

Table 9. snmpTargetParamsTable for <Trap type> 4 or 5

snmpTargetParamsTable (RFC-2573 - SNMP-TARGET-MIB)	New Row
Column Name (* = Part of Index)	Column Value
* snmpTargetParamsName	"@config_n" Where n ranges from 0 to m-1 where m is the number of notification receiver TLV elements in the config file
snmpTargetParamsMPModel SYNTAX: SnmMessageProcessingModel	If <Trap type> = 1 SNMPv1 (0) Else If <Trap type> = 2 or 3 SNMPv2c (1) Else if <Trap type> = 4 or 5 SNMPv3 (3)
snmpTargetParamsSecurityModel SYNTAX: SnmpSecurityModel	If <Trap type> = 1 SNMPv1 (1) Else If <Trap type> = 2 or 3 SNMPv2c (2) Else if <Trap type> = 4 or 5 USM (3) NOTE: The mapping of SNMP protocol types to value here are different from snmpTargetParamsMPModel
snmpTargetParamsSecurityName	<Security Name>
snmpTargetParamsSecurityLevel	The security level of <Security Name>
snmpTargetParamsStorageType	volatile
snmpTargetParamsRowStatus	active (1)

3.6.1.5 snmpNotifyFilterProfileTable –

Create 1 row for each TLV that has a non-zero <Filter Length>

Table 10. snmpNotifyFilterProfileTable

snmpNotifyFilterProfileTable (RFC-2573 - SNMP-NOTIFICATION-MIB)	New Row
Column Name (* = Part of Index)	Column Value
* snmpTargetParamsName	"@config_n" Where n ranges from 0 to m-1 where m is the number of notification receiver TLV elements in the config file
snmpNotifyFilterProfileName	"@config_n" Where n ranges from 0 to m-1 where m is the number of notification receiver TLV elements in the config file
snmpNotifyFilterProfileStorType	volatile
snmpNotifyFilterProfileRowStatus	active (1)

3.6.1.6 snmpNotifyFilterTable

Create 1 row for each TLV that has a non-zero <Filter Length>

Table 11. snmpNotifyFilterTable

snmpNotifyFilterTable (RFC-2573 - SNMP-NOTIFICATION-MIB)	New Row
Column Name (* = Part of Index)	Column Value
* snmpNotifyFilterProfileName	"@config_n" Where n ranges from 0 to m-1 where m is the number of notification receiver TLV elements in the config file
* snmpNotifyFilterSubtree	<Filter OID> from the TLV
snmpNotifyFilterMask	<Zero Length Octet String>
snmpNotifyFilterType	included (1)
snmpNotifyFilterStorageType	volatile
snmpNotifyFilterRowStatus	active (1)

3.6.1.7 snmpCommunityTable

Create 1 row with fixed values if 1 or more TLVs is present

This causes SNMPV1 and V2c Notifications to contain the community string in snmpCommunityName

Table 12. snmpCommunityTable

snmpCommunityTable (RFC-2576 - SNMP-COMMUNITY-MIB)	1st Row
Column Name (* = Part of Index)	Column Value
* snmpCommunityIndex	"@config"
snmpCommunityName	"public"
snmpCommunitySecurityName	"@config"
snmpCommunityContextEngineID	<The engineID of the cable modem>
snmpCommunityContextName	<Zero length octet string>
snmpCommunityTransportTag	<Zero length octet string>
snmpCommunityStorageType	volatile
snmpCommunityStatus	active (1)

3.6.1.8 usmUserTable

Create 1 row with fixed values, if 1 or more TLVs is present. Other rows are created, one each time the engine ID of a trap receiver is discovered

This specifies the user name on the remote notification receivers to send notifications to.

One row in the usmUserTable is created. Then when the engine ID of each notification receiver is discovered, the agent copies this row into a new row and replaces the 0x00 in the usmUserEngineID column with the newly discovered value.

Table 13. usmUserTable

usmUserTable (RFC-2574 - SNMP-USER-BASED-SM-MIB)	1st Row
Column Name (* = Part of Index)	Column Value
* usmUserEngineID	0x00
* usmUserName	"@config" - When other rows are created, this is replaced with the <Security Name> field from the TLV element.
usmUserSecurityName	"@config" - When other rows are created, this is replaced with the <Security Name> field from the TLV element.
usmUserCloneFrom	<don't care> - can't clone this row
usmUserAuthProtocol	None - When other rows are created, this is replaced with None or MD5, depending on the security level of the V3 User
usmUserAuthKeyChange	<don't care> - write only
usmUserOwnAuthKeyChange	<don't care> - write only
usmUserPrivProtocol	None - When other rows are created, this is replaced with None or DES, depending on the security level of the V3 User
usmUserPrivKeyChange	<don't care> - write only
usmUserOwnPrivKeyChange	<don't care> - write only
usmUserPublic	<zero length string>
usmUserStorageType	volatile
usmUserStatus	Active (1)

3.6.1.9 vacmSecurityToGroupTable

Create 3 rows with fixed values, if 1 or more TLVs is present

These are the 3 rows with fixed values - These are used for the TLV entries with <Trap Type> set to 1, 2, or 3 or with a zero length <Security Name>

Table 14. vacmSecurityToGroupTable

vacmSecurityToGroupTable (RFC-2575 - SNMP-VIEW-BASED-ACM-MIB)	1st Row	2nd Row	3rd Row
Column Name (* = Part of Index)	Column Value	Column Value	Column Value
* vacmSecurityModel	SNMPV1 (1)	SNMPV2c (2)	USM (3)
* vacmSecurityName	"@config"	"@config"	"@config"
vacmGroupName	"@configV1"	"@configV2"	"@configUSM"
vacmSecurityToGroupStorageType	volatile	volatile	volatile
vacmSecurityToGroupStatus	active (1)	active (1)	active (1)

The TLV entries with <Trap Type> set to 4 or 5 and a non-zero length <Security Name> will use the rows created in the vacmSecurityToGroupTable by the DH Kickstart process.

3.6.1.10 vacmAccessTable

Create 3 rows with fixed values, if 1 or more TLVs is present

These are the 3 rows with fixed values - These are used for the TLV entries with <Trap Type> set to 1, 2, or 3 or with a zero length <Security Name>

Table 15. vacmAccessTable

vacmAccessTable (RFC-2575 - SNMP-VIEW-BASED-ACM-MIB)	1st Row	2nd Row	3rd Row
Column Name (* = Part of Index)	Column Value	Column Value	Column Value
* vacmGroupName	"@configV1"	"@configV2"	"@configUSM"
* vacmAccessContextPrefix	<Zero length string>	<Zero length string>	<Zero length string>
* vacmAccessSecurityModel	SNMPV1 (1)	SNMPV2c (2)	USM (3)
* vacmAccessSecurityLevel	noAuthNoPriv (1)	noAuthNoPriv (1)	noAuthNoPriv (1)
vacmAccessContextMatch	exact (1)	exact (1)	exact (1)
vacmAccessReadViewName	<Zero length octet string>	<Zero length octet string>	<Zero length octet string>
vacmAccessWriteViewName	<Zero length octet string>	<Zero length octet string>	<Zero length octet string>
vacmAccessNotifyViewName	"@config"	"@config"	"@config"
vacmAccessStorageType	volatile	volatile	volatile
vacmAccessStatus	active (1)	active (1)	active (1)

The TLV entries with <Trap Type> set to 4 or 5 and a non-zero length <Security Name> will use the rows created in the vacmAccessTable by the DH Kickstart process.

3.6.1.11 vacmViewTreeFamilyTable

Create 1 row with fixed values if 1 or more TLVs is present

This row is used for the TLV entries with <Trap Type> set to 1, 2, or 3 or with a zero length <Security Name>

Table 16. vacmViewTreeFamilyTable

vacmViewTreeFamilyTable (RFC-2575 - SNMP-VIEW-BASED-ACM-MIB)	1st Row
Column Name (* = Part of Index)	Column Value
* vacmViewTreeFamilyViewName	"@config"
* vacmViewTreeFamilySubtree	1.3
vacmViewTreeFamilyMask	<Default from MIB>
vacmViewTreeFamilyType	included (1)
vacmViewTreeFamilyStorageType	volatile
vacmViewTreeFamilyStatus	active (1)

The TLV entries with <Trap Type> set to 4 or 5 and a non-zero length <Security Name> will use the rows created in the vacmViewTreeFamilyTable by the DH Kickstart process.

4 OSSI for Radio Frequency Interface

4.1 Subscriber Account Management Interface Specification

Note: The Subscriber Account Management Interface Specification is OPTIONAL for CMTS vendors at this time. However, if a billing interface is provided by a CMTS vendor, it MUST conform to the specification in this section.

The Subscriber Account Management Interface Specification is defined to enable prospective vendors of cable modems and cable modem termination systems to address the operational requirements of subscriber account management in a uniform and consistent manner. It is the intention that this would enable operators and other interested parties to define, design and develop Operations and Business Support System (OBSS) necessary for the commercial deployment of different class of services over cable networks with accompanying usage-based billing of services for each individual subscriber.

Subscriber Account Management described here refers to the following business processes and terms:

- Class of Service Provisioning Processes, which are involved in the automatic and dynamic provisioning and enforcement of subscribed class of policy-based service level agreements (SLAs);
- Usage-Based Billing Processes, which are involved in the processing of bills based on services rendered to and consumed by paying subscribers. This Specification focuses primarily on bandwidth-centric usage-based billing scenarios. It complements the current Telephony Billing Specification that is being developed within the PacketCable architecture.

In order to develop the DOCSIS-OSS Subscriber Account Management Specification, it is necessary to consider high-level business processes common to cable operators and the associated operational scenarios. These issues are discussed in Appendix B.

4.1.1 Service Flows, Service Classes, and Subscriber Usage Billing

The DOCSIS 1.1 RFI specification provides a mechanism for a Cable Modem (CM) to register with its Cable Modem Termination System (CMTS) and to configure itself based on external Quality of Service (QoS) parameters when it is powered up or reset. To quote (in part) from Section 8.1 Theory of Operation:

The principal mechanism for providing enhanced QoS is to classify packets traversing the RF MAC interface into a Service Flow. A Service Flow is a unidirectional flow of packets that is provided a particular Quality of Service. The CM and the CMTS provide this QoS by shaping, policing, and prioritizing traffic according to the QoS Parameter Set defined for the Service Flow.

The requirements for Quality of Service include:

- A configuration and registration function for pre-configuring CM-based QoS Service Flows and traffic parameters.
- Utilization of QoS traffic parameters for downstream Service Flows.
- Classification of packets arriving from the upper layer service interface to a specific active Service Flow
- Grouping of Service Flow properties into named Service Classes, so upper layer entities and external applications (at both the CM and the CMTS) can request Service Flows with desired QoS parameters in a globally consistent way.

A Service Class Name (SCN) is defined in the CMTS via provisioning (see *DOCS-QOS-MIB*). An SCN provides a handle to an associated QoS Parameter Set (QPS) template. Service Flows that are created using an SCN are considered to be “named” Service Flows. The SCN identifies the service characteristics of a Service Flow to external systems such as a billing system or customer service system. For consistency in billing, operators should ensure that SCNs are unique within an area serviced by the same BSS that utilizes this interface. A descriptive SCN might be something like *PrimaryUp*, *GoldUp*, *VoiceDn*, or *BronzeDn* to indicate the nature and direction of the Service Flow to the external system.

A Service Package implements a Service Level Agreement (SLA) between the MSO and its Subscribers on the RFI interface. A Service Package might be known by a name such as *Gold*, *Silver*, or *Bronze*. A Service Package is itself implemented by the set of named Service Flows (using SCNs) that are placed into a CM Configuration File¹ that is stored on a TFTP server. The set of Service Flows defined in the CM Config File are used to create active Service Flows when the CM registers with the CMTS. Note that many Subscribers are assigned to the same Service Package, therefore, many CMs use the same CM Config File to establish their active Service Flows. Also, note that a Service Package has to define at least two Service Flows known as Primary Service Flows that are used by default when a packet matches none of the classifiers for the other Service Flows. A CM Config File that implements a Service Package, therefore, must define the two primary Service Flows using SCNs (e.g., *PrimaryUp* and *PrimaryDn*) that are known to the CMTS if these Service Flows are to be visible to external systems via this billing interface. Note that it is often the practice in a usage sensitive billing environment to segregate the operator's own maintenance traffic to and from the CM into the primary service flows so that this traffic is not reflected in the traffic counters associated the subscriber's SLA service flows.

The DOCSIS 1.1 RFI specification also provides for dynamically created Service Flows. An example could be a set of dynamic Service Flows created by an embedded PacketCable Multimedia Terminal Adapter (MTA) to manage VoIP signaling and media flows. All dynamic Service Flows must be created using an SCN known to the CMTS if they are to be visible to the billing system. These dynamic SCNs do not need to appear in the CM Config File but the MTA may refer to them directly during its own initialization and operation.

During initialization, a CM communicates with a DHCP Server that provides the CM with its assigned IP address and, in addition, receives a pointer to the TFTP Server that stores the assigned CM Config File for that CM. The CM reads the CM Config File and forwards the set of Service Flow definitions (using SCNs) up to the CMTS. The CMTS then performs a macro-expansion on the SCNs (using its provisioned SCN templates) into QoS Parameter Sets sent in the Registration Response for the CM. Internally, each active Service Flow is identified by a 32-bit SFID assigned by the CMTS to a specific CM (relative to the RFI interface). For billing purposes, however, the SFID is not sufficient as the only identifier of a Service Flow because the billing system cannot distinguish the class of service being delivered by one SFID from another. Therefore, the SCN is necessary, in addition to the SFID, to identify the Service Flow's class of service characteristics to the billing system. The billing system can then rate the charges differently for each of the Service Flow traffic counts based on its Service Class (e.g., Gold octet counts are likely to be charged more than Bronze octet counts). Thus, the billing system obtains from the CMTS the traffic counts for each named Service Flow (identified by SFID and SCN) that a subscriber's CM uses during the billing data collection interval. This is true even if multiple active Service Flows (i.e. SFIDs) are created using the same SCN for a given CM over time. This will result in multiple billing records for the CM for Service Flows that have the same SCN (but different SFIDs). Note that the SFID is the primary key to the Service Flow. When an active Service Flow exists across multiple sequential billing files the SFID allows the sequence of recorded counter values to be correlated to the same Service Flow instance.

4.1.2 IP Detail Record (IPDR) Standard

The IPDR Organization (see www.ipdr.org) has defined a generic model for using XML Schema in IP Detail Recording applications. Industry specific IP billing applications such as the Cable Data Systems Subscriber Usage Billing Record can be added to the IPDR standard by mapping the application semantics onto the NDM-U XML Schema syntax. See Appendix E for the DOCSIS OSSI Service Specification submission to IPDR.org for the *DOCSIS Cable Data Systems Subscriber Usage Billing Record*. Appendix E also contains an example IPDR XML format Subscriber Usage Billing file and the IPDR standard XML Schema (.xsd) files that describe the DOCSIS IPDR syntax.

4.1.2.1 IPDR Network Model

The IPDR Network Model is given in the *NDM-U 3.1* specification and is portrayed in Figure 2 below. Note that in Figure 2 the highlighted blocks and interfaces are the only ones defined in this specification. In this network model, the Service Consumer (SC) is the Cable Data Service Subscriber identified by their Cable Modem MAC address, current

¹ The CM Configuration File contains several kinds of information needed to properly configure the CM and its relationship with the CMTS, but for the sake of this discussion, only the Service Flow and Quality of Service components are of interest.

CM IP address, and current CPE IP addresses. The Service Element (SE) is the CMTS identified by its host name, IP address, and current value of its sysUpTime object. The IPDR Recorder (IR) is the billing record formatter function that creates the NDM-U 3.1 schema format XML IPDRs from the internal counters maintained by the CMTS for each Subscriber's running and terminated Service Flows. The IPDR Store (IS) is the function that maintains the billing file in the FTP file system and detects that the billing file has been deleted by the billing collector. The IPDR Recorder and the IPDR Store are functions that may be implemented within the CMTS or hosted on another platform such as an Element Management System (EMS) or Record Keeping Server (RKS). The IPDR Transmitter (IT) represents the billing record collectors that retrieve the billing records from the IPDR Store as specified in section 4.1.5. In this specification the IT retrieves the compressed and possibly encrypted billing file from the IS on a collection cycle determined by the IT.

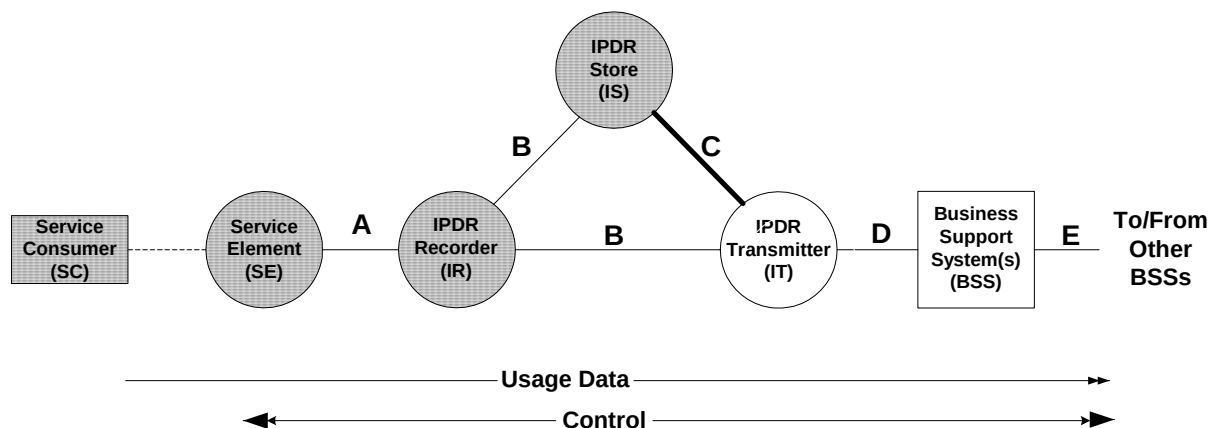


Figure 2. Basic Network Model (ref. NDM-U 3.1 from www.ipdr.org)

Note that the A-interface is not specified by the NDM-U specification because it is an internal interface between the SE and the IR components. The B-interface between the IR and the IS component is also internal to the implementation and is not specified here. In addition, the other B-interface between the IR and the IT components is not used by this specification and is outside the scope of this specification. The C-interface is specified by the NDM-U specification as a file of IPDR records formatted according to the IPDRdoc XML Schema (.xsd) files (see Appendix E). In addition, the billing file in the C-interface is compressed as required by section 4.1.5. The C-interface billing file **MUST** be implemented using the *DOCSIS Cable Data Systems Subscriber Usage Billing Record* submission to the IPDR standard as defined in Appendix E. The D- and E-interfaces are beyond the scope of this specification.

4.1.3 High-Level Requirements for Subscriber Usage Billing Records

This section provides the high-level, functional requirements of this interface. Use of spec words is intentionally avoided as subsequent sections will specify the actual requirements necessary for interoperability utilizing this interface.

The CMTS, or its supporting Element Management System (EMS), must provide formatted Subscriber Usage Billing Records for all subscribers attached to the CMTS on demand to a mediation system or a billing system. The minimum billing record collection interval that must be supported by a CMTS is 15 minutes. The following are the requirements for processing and transmitting Subscriber Usage Billing Records:

The Subscriber Usage Billing File must identify the CMTS by host name and IP address and the time that the billing file was created. The sysUpTime value for the CMTS must also be recorded.

Subscriber usage billing records must be identified by CM MAC address (but not necessarily sorted). The Subscriber's current CM IP address must also be present in the billing record for the Subscriber. If the CMTS is tracking CPE IP addresses behind the Subscriber's CM, then these CPE IP addresses must also be present in the billing record.

Subscriber usage billing records must have entries for each active Service Flow (identified by SFID and Service Class Name) used by all CMs operating in DOCSIS 1.1 (or higher) registration mode during the collection interval². This includes all currently running Service Flows as well as all terminated Service Flows that were deleted and logged during the collection interval. Note well that a provisioned or admitted state SF that was deleted before it became active is not recorded in the billing file, even though it was logged by the CMTS. In addition, billing records for CMs operating in DOCSIS 1.0 registration mode may be created by reporting the DOCSIS 1.0 service as a pair of upstream and downstream Service Flows that contain the aggregate packet and octet counters for each direction. In this case, the billing record must identify the CM as operating in 1.0 mode. Note that there will be null Service Class Names associated with these DOCSIS 1.0 Service Flows.

It must be possible to distinguish running Service Flows from terminated Service Flows in the billing records. Internal CMTS Service Flow log records must not be deleted from the CMTS until after they have been recorded in a billing file stored in non-volatile storage. The CMTS must maintain a separate view of the internal Service Flow log for SNMP access via the DOCS-QOS-MIB. It must not be possible to delete internal Service Flow log entries via SNMP until they have been released by the billing formatter. A terminated Service Flow must be reported into a Billing File exactly once.

It must be possible to identify the Service Flow direction as upstream or downstream without reference to the Service Class Name. The number of packets and octets passed must be collected for each upstream and downstream Service Flow. The number of packets dropped and the number of packets delayed due to enforcement of QoS maximum throughput parameters (SLA) must also be collected for each Service Flow. In the case of an upstream Service Flow, the reported SLA drop and delay counters must represent only the policing performed by the CMTS. Note that since it is possible for a Subscriber to change from one service package to another and back again or to have dynamic service flows occur multiple times, it is possible that there will be multiple entries for a given SCN within a Subscriber's billing record for the collection period. This could also occur if a CM re-registers for any reason (such as CM power failure).

All traffic counters must be based on absolute 64-bit counters as maintained by the CMTS. These counters must be reset to zero by the CMTS if it reinitializes its management interface. The CMTS sysUpTime value is used to determine if the management interface has been reset between adjacent collection intervals. It is expected that the 64-bit counters will not roll over within the service lifetime of the CMTS.

To facilitate processing of the Subscriber Usage Billing Records by a large number of diverse billing and mediation systems an Extensible Markup Language (XML) format is required. Specifically, the IP Detail Record (IPDR) standard as described in IPDR.org's *Network Data Management – Usage, Version 3.1* (NDM-U 3.1) as extended for XML schema format DOCSIS Cable Data Systems Subscriber Usage Billing Records must be used. See Appendix E for the *DOCSIS Cable Data Systems Subscriber Usage Billing Records* Service Specification submission to IPDR.org, the DOCSIS IPDR schema, and an example DOCSIS IPDR XML Schema billing file. See also <http://www.ipdr.org> for more information on the NDM-U specification and Service Specification Guidelines.

To improve the performance of storage and transmission of the NDM-U XML format billing records a compressed file format is required. Lossless compression in GZIP 4.3 format as described in RFC-1952 must be used to store and transmit the billing file. It is expected that an IPDRv3 XML format billing file will compress on the order of 30:1 or better. See also <http://www.gnu.org/software/gzip> for more information.

To improve the network performance of the billing collection activity, a reliable high-throughput TCP stream must be used to transfer billing records between the record formatter and the collection system. Standard FTP GET of the compressed (and optionally encrypted) billing file from the record formatter by the collection system must be supported.

To allow for decoupled scheduling, the billing collection cycle must be driven by the collection system through the standard FTP GET and FTP DELETE operations. Since the collection interval may vary over time, the record formatter is only required to maintain one current billing file in its FTP file system. The collection system (operating on its own schedule) may retrieve the current billing file using FTP GET at any time after it has been constructed and placed in the FTP file system by the record formatter. The collection system must explicitly FTP DELETE the billing file when it no longer needs it. The retrieval model is detailed in Section 4.1.5.

² Subscriber billing records are a method of byte usage accounting only. Some types of Service Flows can consume system resources without bytes actually being passed (e.g., an active RTPS flow or an admitted UGS flow). Billing for these types of resources is beyond the scope of this specification.

To ensure the end-to-end privacy and integrity of the billing records, while either stored or in transit, an authentication and encryption mechanism must be provided between the record formatter and the collection system. The security model is detailed in Section 4.1.6.

4.1.4 Billing Collection Interval

Subscriber Usage Billing Records report the absolute traffic counter values for each Service Flow used by a Cable Modem (Subscriber) that has become active during the billing collection interval as seen at the end of the interval. The collection interval is defined as the time between the creation of the previous billing file (T_{prev}) and the creation of the current billing file (T_{now}). See Figure 3 below. There are two kinds of Service Flows that are reported in the current billing file: 1) SFs that are still running at the time the billing file is created and 2) terminated SFs that have been deleted and logged during the collection interval. A provisioned or admitted state SF that was deleted before it became active **MUST NOT** be recorded in the billing file, even though it was logged by the CMTS.

The CMTS (or supporting EMS) **MUST** record any currently running SFs using T_{now} as the timestamp for its counters and **MUST** identify them in the IPDR SFtype element as “Interim”. Terminated SFs that have a deletion time (T_{del}) later than T_{prev} are the only ones recorded in the current billing file (i.e. a terminated SF **MUST BE** reported exactly once). A CMTS **MUST** record a terminated SF using its T_{del} from the log as the timestamp for its counters and **MUST** identify it in the IPDR SFtype element as “Stop”. Note that the timestamps are based on the formatter’s recording times, not the collection system’s retrieval times. Since the collection cycle may vary over time, the recording times in the billing file can be used to construct an accurate time base over sequences of billing files.

In the example shown in Figure 3 below there are four Service Flows recorded for a Subscriber in the current billing file being created at T_{now} . SFa is a long running SF that was running during the previous collection interval (it has the same SFID in both the current and the previous billing files). SFa was recorded as type Interim at T_{prev} in the previous billing file and is recorded again as type Interim at T_{now} in the current file. SFb is a running SF that was created during the current collection interval. SFb is recorded as type Interim for the first time at T_{now} in the current file. SFc is a terminated SF that was running during the previous collection interval but was deleted and logged during the current collection interval. SFc was recorded as type Interim at T_{prev} in the previous billing file and is recorded as type Stop at the logged $T_{del}(c)$ in the current file. SFd is a terminated SF that was both created and deleted during the current collection interval. SFd is recorded only once as type Stop at the logged $T_{del}(d)$ in the current billing file only.

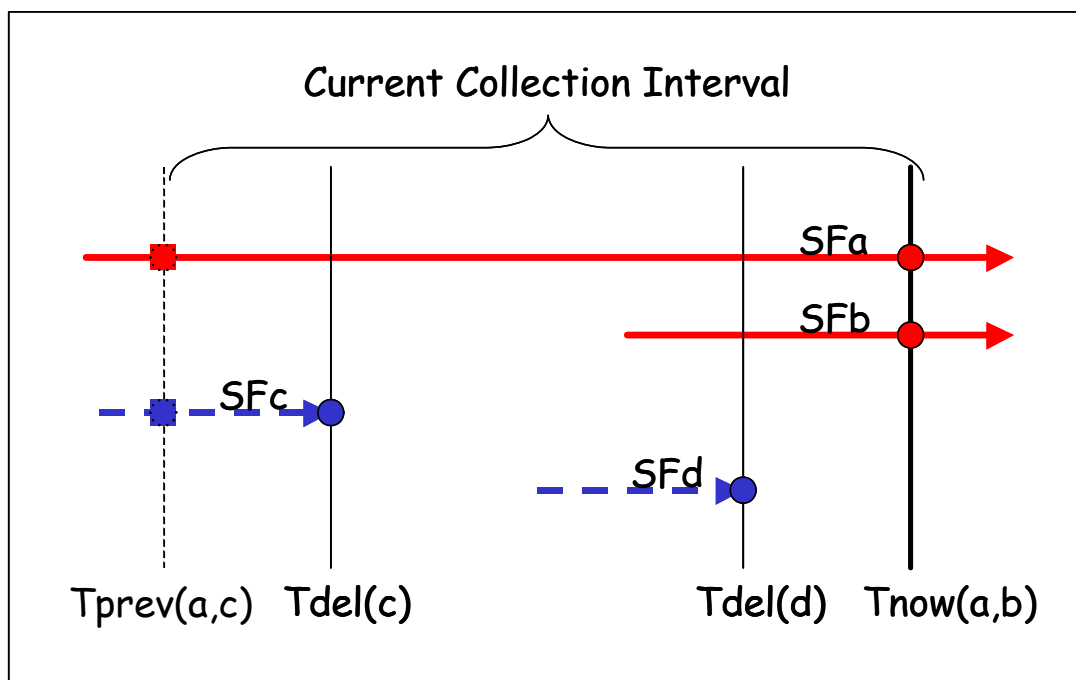


Figure 3. Billing Collection Interval Example

4.1.5 Billing File Retrieval Model

Billing files are built by the record formatter on the CMTS (or supporting EMS) and are then retrieved by the collection system in a decoupled manner using FTP semantics. There is no explicit signaling protocol between them and no prior arrangement regarding the frequency of billing collection. The CMTS (or supporting EMS) is responsible for creating the current billing file and **MUST** place it into its FTP file system only when the file is completely built. The formatter only creates one billing file which it **MUST** protect until the collection system is done with it. The collection system **MAY** retrieve the current billing file via FTP GET at any time after the file becomes available in the formatter's FTP file system. When the collection system has successfully retrieved the billing file, it **MUST** remove the file via FTP DELETE from the formatter's FTP file system. The formatter **MUST** monitor the existence of the billing file in its FTP file system and when it no longer exists, the formatter **MUST** begin to create the next billing file. The formatter **MUST** finish constructing the next billing file and have it ready for retrieval in its FTP file system within 15 minutes of the previous file's deletion. If the billing file does not yet exist in the formatter's FTP file system when the collection system comes to retrieve it, the collection system **MUST** back off and return later to try again. The specific timeout for collection system retries is implementation dependent, however, the collection system **MUST NOT** make more than 3 retrieval attempts within any 5-minute period.

Note that if the collection system fails for any reason, the formatter will retain and protect the last billing file created until the collection system returns to retrieve the file. In this case, even though the recording timestamps in the current billing file may be quite old, the collection system will still retrieve the current file and delete it in the standard manner. The formatter will then immediately begin construction of a new billing file based on the current values of the CMTS's internal absolute 64-bit counters and the current timestamp. The collection system may then return at any time after the minimum cycle time (i.e. 15 minutes) and retrieve the new billing file with the current timestamps. The absolute values of the counters will always be preserved by the CMTS while it is operating, only the collection interval will be extended due to the outage on the collection system. The billing system can use the recording timestamps in the two files to accurately reconstruct the time base of the counters. Furthermore, the collection system **MAY** deliberately vary its collection cycles based on time of day or day of week. This decoupled billing file retrieval model works well for this case also.

The decoupled billing file retrieval model also supports multiple retrievals by multiple collection systems so long as the last collection system deletes the billing file when it is done with it. However, there is no requirement to support multiple simultaneous file transfers from the formatter. How the multiple collection systems coordinate this between themselves is beyond the scope of this specification

4.1.6 Billing File Security Model

The billing file security model has two components: 1) secure user authentication to control access to the billing file in the formatter's FTP file system and 2) secure file transfer to ensure the privacy and the integrity of the billing file while it is in transit. Both of these components are provided by the Secure Shell protocol version 2 (SSH2) and its Secure FTP (SFTP) subsystem as described by Internet drafts maintained by the IETF's SECSH working group at www.ietf.org/html.charters/secsh-charter.html. Additional information may be obtained from www.openssh.org, which provides an open source implementation of SSH2 and SFTP. A CMTS (or supporting EMS) hosting the billing formatter **MUST** provide secure access to its FTP file system via SSH2 and SFTP. It is also strongly recommended that the operator disable legacy insecure Telnet and FTP access to the formatter's platform when SSH2/SFTP are active. How legacy Telnet and FTP are disabled is beyond the scope of this specification.

To ensure restricted access to billing information, the billing collector **MUST** have its own userid and password for access to the formatter's billing file directory via SSH2/SFTP. Furthermore, the billing collector's userid **MUST NOT** be shared with any other applications or users hosted on the formatter's platform. SSH2 user public key authentication is **OPTIONAL** for the billing collector's userid. How userids, keys, and passwords are administered on the formatter's platform is beyond the scope of this specification. Note also that the collection system requires both read and delete access permissions to the billing file directory in the formatter's FTP file system.

While the formatter's platform **MUST** provide secure authentication and file transfer capabilities, the operator may elect to not utilize them. In this case, the formatter's platform **MUST** provide access to the billing file directory via legacy insecure FTP and the billing collector **MUST** have its own userid and password for legacy FTP access as well. Again, it is strongly recommended that the operator not allow insecure legacy FTP access to the formatter's billing file

4.1.7 IPDR Record Structure

The NDM-U 3.1 specification defines the IPDRDoc record structure. The IPDRDoc 3.1 XML schema (see IPDRDoc3.1.xsd in Appendix E) defines the hierarchy of elements within the IPDR document that MUST be supported by the CMTS (as shown in Figure 4 below).

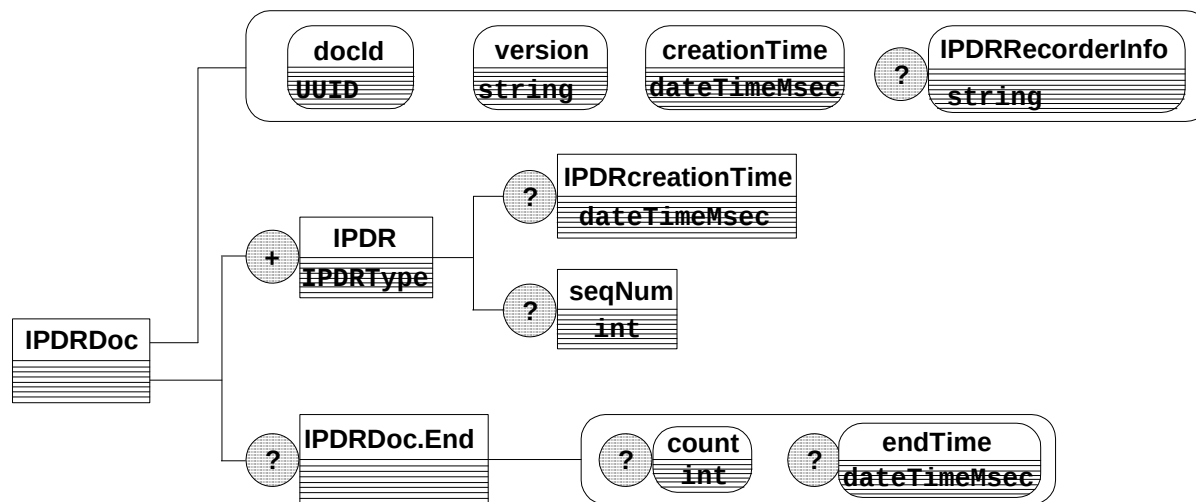


Figure 4. IPDRDoc 3.1 Generic Schema

The IPDRDoc3.1.xsd schema defines the generic structure of any IPDR document regardless of application. To complete the definition of an application specific IPDR record structure, an application schema must be provided that imports the basic IPDRDoc3.1.xsd schema. The DOCSIS IPDR Version 3.1 schema (see DOCSIS-3.1-B.0.xsd in Appendix E.2.1) defines the elements that record the DOCSIS specific information that MUST be supported by the CMTS (as shown in Figure 5 below). Note that the DOCSIS-Type in is the application specific implementation of the IPDR element shown in Figure 4 above. Thus, the DOCSIS specific elements are sub elements of the IPDR element.

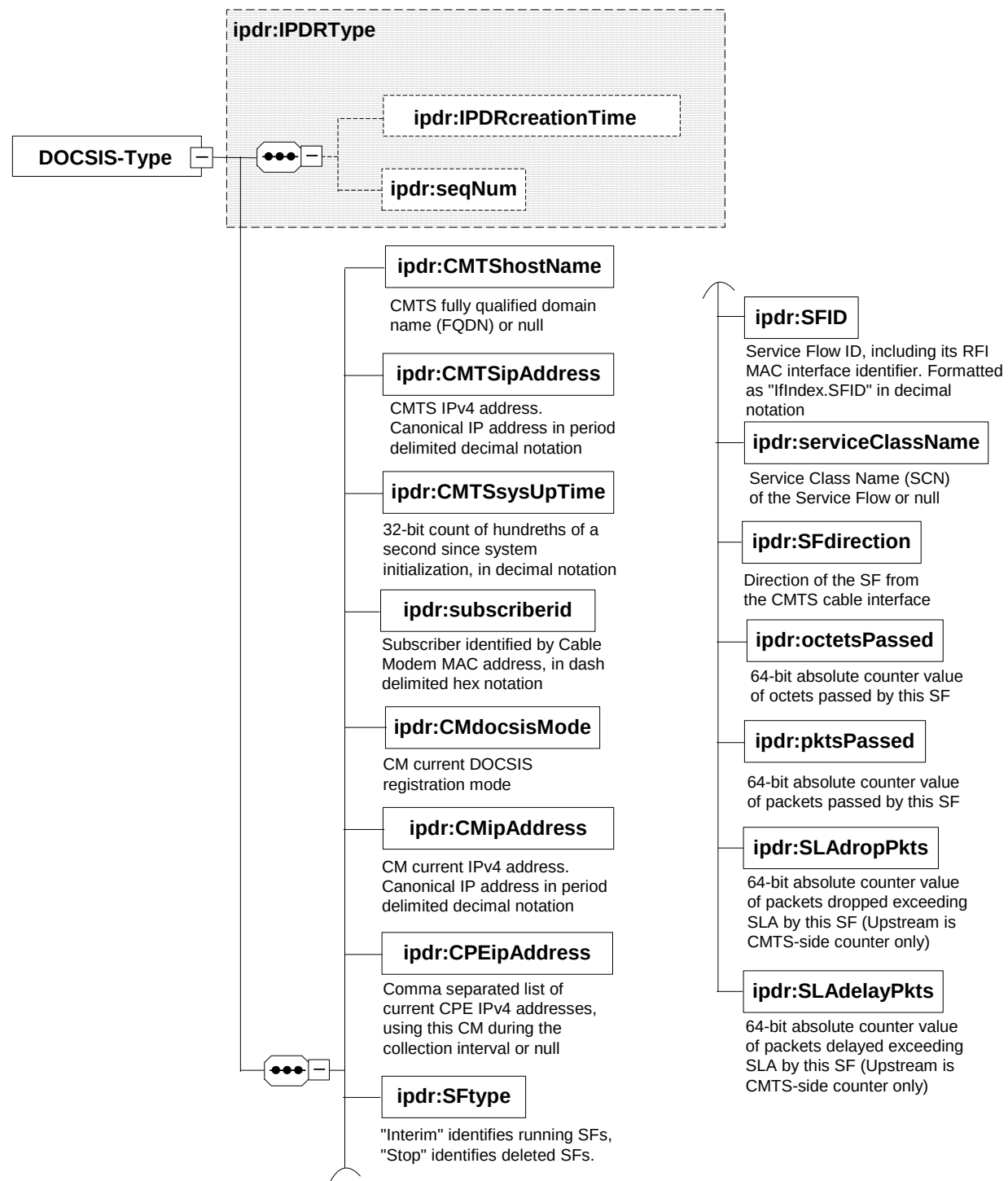


Figure 5. DOCSIS IPDR 3.1 Schema

The following elements and attributes are the only ones used by the DOCSIS Cable Data Systems Subscriber Usage Billing Record IPDR instance document (see Appendix E). These elements and attributes are described below:

The *IPDRDoc* element is the outermost element that describes the IPDR billing file itself. It defines the XML namespace, the identity of the XML schema document, the version of the specification, the timestamp for the file, a unique document identifier, and the identity of the IPDR recorder. An *IPDRDoc* is composed of multiple IPDR records. The attributes for the *IPDRDoc* element MUST be as follows:

- a) *xmlns*="http://www.ipdr.org/namespaces/ipdr"
Constant: the XML namespace identifier. Defined by ipdr.org.
- b) *xmlns:xsi*="http://www.w3.org/2001/XMLSchema-instance"
Constant: the XML base schema identifier. Defined by ipdr.org.
- c) *xsi:schemaLocation*="DOCSIS-3.1-B.0.xsd"
Constant: the name of the DOCSIS application specific schema file.
- d) *version*="3.1"
Constant: the version of the IPDR document. Defined by ipdr.org.
- e) *creationTime* ="yyyy-mm-ddThh:mm:ssZ"
UTC time stamp at the time the billing file is created (in ISO format). For example:
creationTime="2002-06-12T21:11:21Z". Note that IPDR timestamps MUST always be in UTC/GMT (Z).
- f) *docId*="<32-bit UTC timestamp>-0000-0000-0000-<48-bit MAC address>"
The unique document identifier. The DOCSIS *docId* is in a simplified format that is compatible with the Universal Unique Identifier (UUID) format required by the IPDR NDM-U 3.1 specification. The 32-bit UTC timestamp component MUST be the IPDRDoc *creationTime* in seconds since the epoch 1 Jan 1970 UTC formatted as eight hex digits. The 48-bit MAC address component MUST be the ethernet address of the CMTS management interface formatted as 12 hex digits. All other components MUST be set to zero. In the context of the minimum 15-minute IPDR billing file collection cycle specified in this document, this simplified UUID is guaranteed to be unique across all CMTSs and for the foreseeable future. For example:
docId="3d07b8f9-0000-0000-0000-00015c11bfbe".
- g) *IPDRRecorderInfo*="hostname.mso.com"
Identifies the IPDR Recorder (IR) from the network model in Figure 2 above. This attribute MUST identify the billing record formatter by the fully qualified hostname of the CMTS or the EMS where the formatter resides. If a hostname is not available, then this MUST be the IPv4 address of the CMTS or EMS formatted in dotted decimal notation.

An *IPDR* element MUST describe a single Subscriber Usage Billing Record for a single DOCSIS service flow. The *IPDR* is further structured into DOCSIS specific sub elements that describe the details of the CMTS, the subscriber (CM and CPE), and the service flow itself. While the generic IPDR record structure is designed to describe most time-based and event-oriented IP services, this feature is not particularly relevant to the Cable Data Service Subscriber Usage Billing Records and is largely ignored. This is because a service session at the CMTS is just the aggregate usage of an active Service Flow during the billing collection interval. Another way to look at it is as if there is really only one event being recorded: the billing collection event itself. The attributes for the *IPDR* element are

xsi:type="DOCSIS-Type"

Constant: identifies the DOCSIS application specific type of the IPDR record.

The *IPDRcreationTime* element identifies the time associated with the counters for this service flow. The format MUST be the same as the *IPDRDoc creationTime* attribute (see 1e. above). *IPDRcreationTime* MUST be the same as the *IPDRDoc creationTime* when the service flow is still running (i.e. *SFtype* = Interim). *IPDRcreationTime* MUST be the time the service flow was deleted when the service flow has been terminated (i.e. *SFtype* = Stop). Note that a Stop IPDR is always earlier than the *IPDRDoc creationTime*. Also, note that this sub element is optional in the basic IPDR 3.1 schema, but is REQUIRED for all DOCSIS IPDRs.

The *seqNum* element is an optional sub element of the basic IPDR 3.1 schema. It MUST NOT be used in DOCSIS IPDRs. Note that there is no ordering implied in DOCSIS IPDRs within an *IPDRDoc*.

The *CMTShostName* element is a REQUIRED element that contains the fully qualified domain name (FQDN) of the CMTS if it exists. For example: *cmts01.mso.com*. This element MUST be null if no FQDN exists (i.e. *<CMTShostName></CMTShostName>* or *<CMTShostName/>*).

The *CMTSipAddress* element contains the IP address of the management interface of the CMTS. This element is REQUIRED and MUST be represented in standard IPv4 decimal dotted notation (for example: 10.10.10.1).

The *CMTSsysUpTime* element contains the value of the sysUpTime SNMP object in the CMTS taken at the IPDRDoc creationTime. This element is REQUIRED and MUST be the count of 100ths of seconds since the CMTS management interface was initialized. If the CMTSsysUpTime regresses between adjacent IPDRDocs, then the CMTS management interface has been reset and all service flow counters have been reset to zero. Note well: this value MUST be the same for each IPDR within a given IPDRDoc file, regardless of the IPDRcreationTime of a given IPDR.

The *subscriberId* element contains the unique identifier of the subscriber. This element is REQUIRED and MUST be the subscriber's cable modem 48-bit MAC address formatted as dash delimited hex digits. For example: 11-11-11-11-11-11.

The *CMdocsisMode* element identifies the registration mode of the Cable Modem as "1.0", "1.1", or "2.0". If the registration mode is "1.0" then the reported Service Flow contains the aggregate packet and octet counters for the DOCSIS 1.0 service in this direction. This element is REQUIRED.

The *CMipAddress* element contains the current IP address of the subscriber's cable modem. This element is REQUIRED and MUST be represented in standard IPv4 decimal dotted notation (for example, 10.100.100.123). Note that this address can change over a set of IPDRDoc files if the operator's DHCP server reassigns IP addresses to cable modems.

The *CPEipAddress* element MUST contain a comma delimited list of the current IP addresses of all of the subscriber's CPE using this cable modem or null if there are none being tracked by the CMTS (i.e. <CPEipAddress></CPEipAddress> or <CPEipAddress/>). If there are multiple CPE using the CM, then there MUST be multiple CPE IP addresses in the list. Each CPE IP address MUST be represented in standard IPv4 decimal dotted notation (for example: 12.12.12.123 or 12.12.12.123, 12.12.12.124, 12.12.12.125). Note that the configuration state of the DOCS-SUBMGT-MIB influences whether CPE IP addresses are being tracked by the CMTS and are thus being reported in the IPDRs (the DOCS-SUBMGT-MIB controls the CM and CPE filters on the CMTS).

The *SFtype* element identifies the kind of service flow being described by this IPDR. This element is REQUIRED and MUST have either of two values: "Interim" identifies this SF as currently running in the CMTS and "Stop" identifies this SF as having been terminated in the CMTS. A running service flow has active counters in the CMTS and this IPDR MUST contain the current sample of these counters. A terminated service flow has logged counters in the CMTS and this IPDR MUST contain the final counter values for this service flow. Note well: the internal logged SF counters on the CMTS MUST NOT be deleted until after the terminated service flow has been recorded into an IPDR record that has been stored in non-volatile memory, regardless of any other capability to manage them via SNMP through the DOCS-QOS-MIB.

The *SFID* element contains the internal service flow identifier known to the CMTS. This element is REQUIRED and is needed to correlate the IPDRs for an individual service flow between adjacent IPDRDoc files when computing delta counters between samples. Note that SFIDs are relative to their RFI MAC interface. Therefore, the SFID element MUST be formatted as ifIndex.SFID where the ifIndex component is the interface index in the CMTS ifTable for the RFI MAC interface and the SFID component is the 32-bit identifier assigned by the CMTS to this service flow. Both components MUST be represented as decimal values (for example, 15.34567). To avoid potential confusion in the billing system, the CMTS MUST NOT reuse the SFID component for a minimum of two billing collection cycles.

The *serviceClassName* element contains the name associated with the QoS parameter set for this service flow in the CMTS. The SCN is an ASCII string identifier, such as "GoldUp" or "SilverDn", that can be used by external operations systems to assign, monitor, and bill for different levels of bandwidth service without having to interpret the details of the QoS parameter set itself. A service flow is associated with an SCN whenever a cable modem configuration file uses the SCN to define an active service flow. A dynamic service flow application such as PacketCable may also assign an SCN to a service flow as a parameter during the dynamic creation of the service flow. Note that use of SCNs is optional within the context of the DOCSIS RFI specification, however, for operational purposes, especially when billing for tiered data services per this specification, their use often becomes mandatory. Since this policy is within the control of the operator, the use of SCNs is not mandatory in this specification, but rather highly recommended. Note well: this element is REQUIRED in the IPDR record, but if no SCN is used to identify the service flow in the CMTS, then this element MUST have a null value (that is <serviceClassName></serviceClassName> or <serviceClassName/>). Note also that a CM operating in DOCSIS 1.0 mode will not have any SCNs assigned and this element will be null.

The *SFdirection* element identifies the service flow direction relative to the CMTS RFI interface. This element is REQUIRED and MUST have one of two values: "Upstream" identifies service flows passing packets from the cable modem to the CMTS, and "Downstream" identifies service flows passing packets from the CMTS to the cable modem.

The *octetsPassed* element MUST contain the current 64-bit count of the number of octets passed by this service flow formatted in decimal notation. This element is REQUIRED. If the SFtype is Interim, then this is the current value of the running counter. If the SFtype is Stop, then this is the final value of the terminated counter. The 64-bit counter value will not wrap around within the service lifetime of the CMTS. If the CMdocsisMode for this service flow is "1.0" then this element contains the aggregate octet count for the DOCSIS 1.0 service in this direction.

The *pktsPassed* element MUST contain the current 64-bit count of the number of packets passed by this service flow formatted in decimal notation. This element is REQUIRED. If the SFtype is Interim, then this is the current value of the running counter. If the SFtype is Stop, then this is the final value of the terminated counter. The 64-bit counter value will not wrap around within the service lifetime of the CMTS. If the CMdocsisMode for this service flow is "1.0" then this element contains the aggregate packet count for the DOCSIS 1.0 service in this direction.

The *SLAdropPkts* and *SLAdelayedPkts* elements contain the current 64-bit count of the number of packets dropped or delayed by this service flow due to enforcement of the maximum throughput limit specified by the Service Level Agreement (SLA) as implemented by the QoS parameter set. These elements are REQUIRED for all service flows. For upstream service flows, these counters record only the SLA enforcement performed by the CMTS. Upstream packets dropped or delayed at the CM are not recorded here. These counters are formatted in decimal notation. If the SFtype is Interim, then this is the current value of the running counter. If the SFtype is Stop, then this is the final value of the terminated counter. The 64-bit counter value will not wrap around within the service lifetime of the CMTS. If the CMdocsisMode for this service flow is "1.0" then these elements contain the aggregate SLA policing packet count for the DOCSIS 1.0 service in this direction. Note that these values are provided to aid the operator in identifying subscribers who are attempting to use more bandwidth than their SLA provides. This may be an opportunity to offer the subscriber a higher capacity SLA consistent with his/her demonstrated needs.

IPDRDoc.End MUST be the last element inside IPDRDoc that describes the IPDR billing file itself. It defines the count of IPDRs that are contained in the file and the ending timestamp for the file creation.

count="nnnn"

Where nnnn MUST be the decimal count of the number of IPDR records in this IPDRDoc.

endTime="yyyy-mm-ddThh:mm:ssZ"

MUST be the UTC time stamp at the time the billing file is completed (formatted as above). For example: *endTime*="2002-06-12T21:11:23Z".

4.2 Configuration Management

Configuration management is concerned with initializing, maintaining, adding and updating network components. In a DOCSIS environment, this includes a cable modem and/or CMTS. Unlike performance, fault, and account management, which emphasize network monitoring, configuration management is primarily concerned with network control. Network control, as defined by this interface specification, is concerned with modifying parameters in and causing actions to be taken by the cable modem and/or CMTS. Configuration parameters could include both identifiable physical resources (for example, Ethernet Interface) and logical objects (for example, IP Filter Table).

Modifying the configuration information of a CM and/or CMTS can be categorized as follows:

- Non-operational
- Operational

Non-operational changes occur when a manager issues a modify command to a CM/CMTS, and the change doesn't effect the operating environment. For example, a manager may change contact information, such as the name and address of the person responsible for a CMTS.

Operational changes occur when a manager issues a modify command to a CM/CMTS, and the change affects the underlying resource or environment. For example, a manager may change the docsDevResetNow object from false to true, which in turn will cause the CM to reboot.

To adjust the necessary attribute values, the CM and CMTS MUST support MIB objects as specified in section 3 of this document.

While the network is in operation, configuration management will be responsible for monitoring the configuration and making changes in response to commands via SNMP or in response to other network management functions.

For example, a *performance management function* may detect that response time is degrading due to a high number of uncorrected frames, and may issue a configuration management change to modify the modulation type from 16Qam to QPSK. A *fault management function* may detect and isolate a fault and may issue a configuration management change to bypass the fault.

4.2.1 Version Control

The CM MUST support software revision and operational parameter configuration interrogation.

The CM MUST include at least the hardware version, Boot ROM image version, vendor name, software version, and model number in the sysDescr object (from [RFC-3418]). The CM MUST support docsDevSwCurrentVers MIB object and the object MUST contain the same software revision information as shown in the software information included in the sysDescr object.

The format of the specific information contained in the sysDescr MUST be as follows:

To report	Format of each field
Hardware Version	HW_REV: <Hardware version>
Vendor Name	VENDOR: <Vendor name>
Boot ROM	BOOTR: <Boot ROM Version>
Software Version	SW_REV: <Software version>
Model Number	MODEL: <Model number>

Each type value pair MUST be separated with a colon and blank space. Each pair is separated by a “;” followed by a blank. For instance, a sysDescr of a CM of vendor X, hardware version 5.2, Boot ROM version 1.4, SW version 2.2, and model number X

MUST appear as following:

any text<<HW_REV: 5.2; VENDOR: X; BOOTR: 1.4; SW_REV 2.2; MODEL: X>>any text

The CM MUST report at least all of the information necessary in determining what SW the CM is capable of being upgraded to. If any fields are not applicable, the CM MUST report “NONE” as the value. For example; CM with no BOOTR, CM will report BOOTR: NONE.

The CM MUST implement the docsDevSwCurrentVers object ([RFC-2669]) to report the current software version.

The intent of specifying the format of sysDescr is to define how to report information in a consistent manner so that sysDescr field information can be programmatically parsed. This format specification does not intend to restrict the vendor’s hardware version numbering policy.

The CMTS MUST implement the sysDescr object (from [RFC-3418]). For CMTS, format of information and the content of the information in sysDescr is vendor dependent.

4.2.2 System Initialization and Configuration

There are several methods available to configure CM and CMTS including console port, SNMP set, configuration file, and configuration-file-based SNMP encoded object. The CM MUST support system initialization and configuration via configuration file, configuration-file-based SNMP encoded object and SNMP set. The CMTS MUST support system initialization and configuration via telnet connection, console port, and SNMP set. The CM and CMTS (only CMTS that support configuration by configuration file) MUST support any valid configuration file regardless of configuration file size.

4.2.3 Secure Software Upgrades

The CM secure software upgrade detail process is documented in the Appendix D of BPI+ specification.

DOCSIS 1.1 CM MUST use secure software upgrade mechanism to perform software upgrade regardless of what DOCSIS CMTS version (1.0 or 1.1) it is connected to. When a 1.1 CM is connected to a 1.1 CMTS, the 1.1 CM MUST operate in either DOCSIS 1.1 mode or DOCSIS 1.0 mode. When a 1.1 CM is connected to a 1.0 CMTS, the 1.1 CM MUST operate in DOCSIS 1.0 mode. This means that a DOCSIS 1.1 CM MUST use secure software upgrade mechanism to perform software upgrade regardless of what mode it operates in (1.0 mode or 1.1 mode).

There are two available secure software download schemes including manufacture control scheme and operator control scheme.

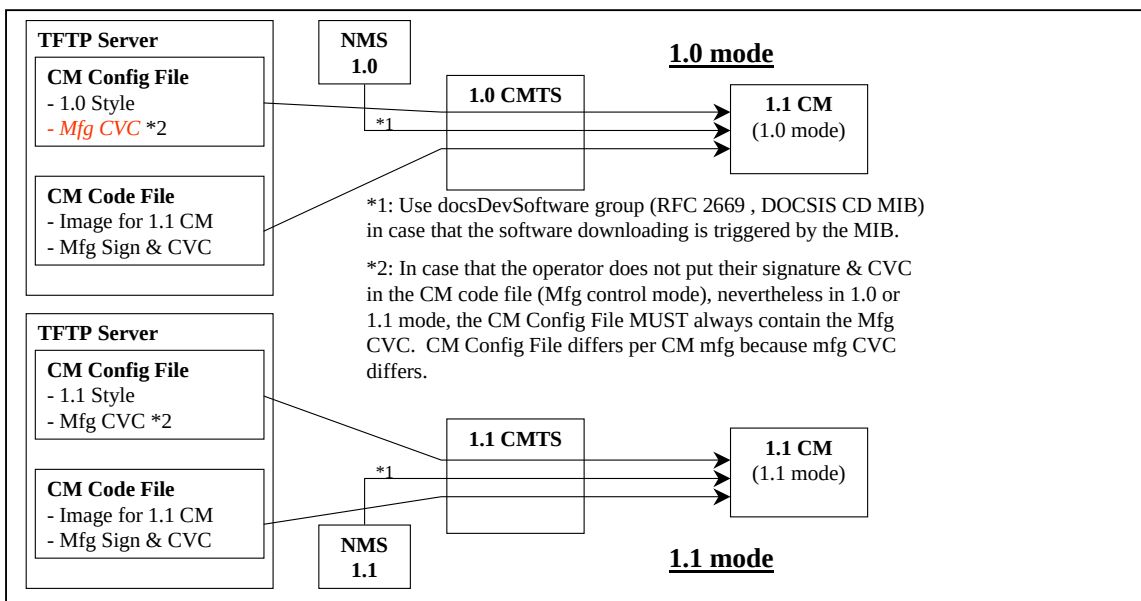


Figure 6. Manufacture control scheme

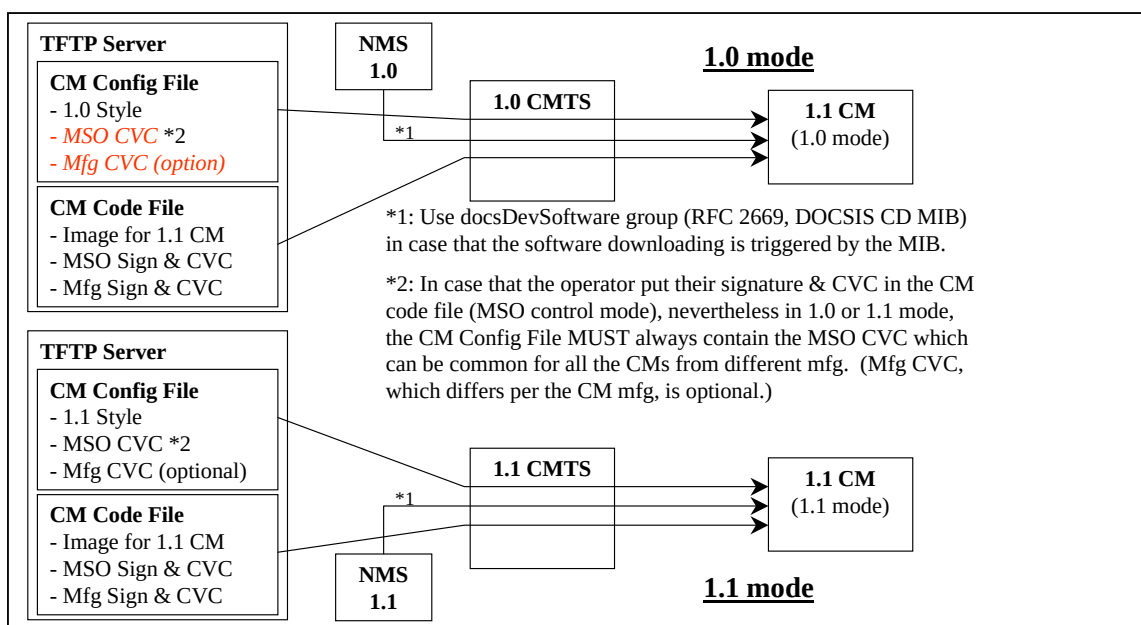


Figure 7. Operator control scheme

Prior to secure software upgrade initialization, CVC information is needed to be initialized at the CM for software upgrade. Depending on the scheme (described above) that the operator chooses to implement, appropriate CVC information MUST be include in the configuration file. It is recommended that CVC information always be present in the configuration file so that a device will always have the CVC information initialized and read if the operator decides

to use SNMP-initiate upgrade as a method to trigger a secure software upgrade operation. If the operator decides to use configuration-file-initiate upgrade as a method to trigger secure software download, CVC information is needed to be present in the configuration file at the time the modem is rebooted to get the configuration file that will trigger the upgrade only.

There are two methods to trigger secure software download including SNMP-initiated and configuration-file-initiated. Both methods **MUST** be supported by CM and **MAY** be supported by CMTS.

The following describes the SNMP-initiated mechanism. Prior to SNMP-initiate upgrade, a CM **MUST** have valid X.509 compliant code verification certificate information. From a network management station:

- Set docsDevSwServer to the address of the TFTP server for software upgrades
- Set docsDevSwFilename to the file pathname of the software upgrade image
- Set docsDevSwAdminStatus to Upgrade-from-mgt.

If docsDevSwAdminStatus is set to ignoreProvisioningUpgrade(3), the CM **MUST** ignore any software download configuration file setting and not attempt a configuration file initiated upgrade.

docsDevSwAdminStatus **MUST** persist across reset/reboots until over-written from an SNMP manager or via a TLV-11 setting in the CM configuration file.

The default state of docsDevSwAdminStatus **MUST** be allowProvisioningUpgrade{2} until it is over-written by ignoreProvisioningUpgrade{3} following a successful SNMP initiated software upgrade or otherwise altered by the management station.

docsDevSwOperStatus **MUST** persist across resets to report the outcome of the last software upgrade attempt.

After the CM has completed a configuration-file-initiated secure software upgrade, the CM **MUST** reboot and become operational with the correct software image as specified in [DOCSIS 5]. After the CM is registered, it **MUST** adhere to the following requirements:

- docsDevSwAdminStatus **MUST** be allowProvisioningUpgrade{2}
- docsDevSwFilename **MAY** be the filename of the software currently operating on the CM
- docsDevSwServer **MAY** be the address of the TFTP server containing the software that is currently operating on the CM
- docsDevSwOperStatus **MUST** be completeFromProvisioning{2}
- docsDevSwCurrentVer **MUST** be the current version of the software that is operating on the CM

After the CM has completed an SNMP-initiated secure software upgrade, the CM **MUST** reboot and become operational with the correct software image as specified in [DOCSIS 5]. After the CM is registered, it **MUST** adhere to the following requirements:

- docsDevSwAdminStatus **MUST** be ignoreProvisioningUpgrade{3}
- docsDevSwFilename **MAY** be the filename of the software currently operating on the CM
- docsDevSwServer **MAY** be the address of the TFTP server containing the software that is currently operating on the CM
- docsDevSwOperStatus **MUST** be completeFromMgt{3}
- docsDevSwCurrentVer **MUST** be the current version of the software that is operating on the CM

The CM **MUST** properly use ignoreProvisioningUpgrade status to ignore software upgrade value that may be included in the CM configuration file and become operation with the correct software image and after the CM is registered, it **MUST** adhere to the following requirements:

- docsDevSwAdminStatus **MUST** be ignoreProvisioningUpgrade{3}
- docsDevSwFilename **MAY** be the filename of the software currently operating on the CM
- docsDevSwServer **MAY** be the address of the TFTP server containing the software that is currently operating on the CM

- docsDevSwOperStatus MUST be completeFromMgt{3}
- docsDevSwCurrentVer MUST be the current version of the software that is operating on the CM

Retries due to a power loss or reset are only required for an SNMP-initiated upgrade. If a power loss or reset occurs during a config file-initiated upgrade, the CM will follow the upgrade TLV directives in the configuration file upon reboot. It will not retry the previous upgrade. The config file upgrade TLVs essentially provides a retry mechanism that is not available for an SNMP-initiated upgrade.

If a CM suffers a loss of power or resets during an SNMP-initiated upgrade, the CM MUST resume the upgrade without requiring manual intervention and when the CM resumes the upgrade process:

- docsDevSwAdminStatus MUST be Upgrade-from-mgt{1}
- docsDevSwFilename MUST be the filename of the software image to be upgraded
- docsDevSwServer MUST be the address of the TFTP server containing the software upgrade image to be upgraded
- docsDevSwOperStatus MUST be inProgress{1}
- docsDevSwCurrentVers MUST be the current version of software that is operating on the CM

In case where the CM reaches the maximum number of TFTP download retries (max retries = 3) resulting from multiple losses of power or resets during an SNMP-initiated upgrade, the CM MUST behave as specified in [DOCSIS 5]; in addition, the CM's status MUST adhere to the following requirements after it is registered:

- docsDevSwAdminStatus MUST be allowProvisioningUpgrade{2}
- docsDevSwFilename MUST be the filename of the software that failed the upgrade process.
- docsDevSwServer MUST be the address of the TFTP server containing the software that failed the upgrade process
- docsDevSwOperStatus MUST be other{5}
- docsDevSwCurrentVer MUST be the current version of software that is operating on the CM

If a CM suffers a loss of power or resets during a configuration file-initiated upgrade, when the CM reboots the CM MUST ignore the fact that a previous upgrade was in progress and either not perform an upgrade if no upgrade TLVs are present in the config file, or if upgrade TLVs are present take the action described in the requirements in section 10.1 of [DOCSIS 5], at the time of the reboot.

In the case where the CM had a configuration file initiated upgrade in progress during a reset and if there are no upgrade TLVs in the config file upon reboot:

- docsDevSwAdminStatus MUST be allowProvisioningUpgrade{2}
- docsDevSwFilename MAY be the filename of the current software image.
- docsDevSwServer MAY be the address of the TFTP server containing the software that is currently operating in the CM.
- docsDevSwOperStatus MUST be other{5}
- docsDevSwCurrentVers MUST be the current version of software that is operating on the CM

In the case where the CM had a configuration file initiated upgrade in progress during a reset, if there are upgrade TLVs in the config file upon reboot:

- docsDevSwAdminStatus MUST be allowProvisioningUpgrade{2}
- docsDevSwFilename MUST be the filename contained in TLV-9 of the config file.
- docsDevSwServer MUST be the address of the TFTP server containing the software to be loaded into the CM, (either the value of TLV-21 in the config file if present, or the address of the configuration file TFTP server if TLV-21 is not present per the requirements stated in section 10.1 of [DOCSIS 5].)
- docsDevSwOperStatus MUST be inProgress{1}
- docsDevSwCurrentVers MUST be the current version of software that is operating on the CM

If a CM exhausts the required number of TFTP retries by issuing a total of 16 consecutive TFTP requests, the CM MUST behave as specified in [DOCSIS 5] and then the CM MUST fall back to last known working image and proceed to an operational state and adhere to the following requirements:

- docDevSwAdminStatus MUST be allowProvisioningUpgrade{2}
- docDevSwFilename MUST be the filename of the software that failed the upgrade process
- docsDevSwServer MUST be the address of the TFTP server containing the software that failed the upgrade process
- docsDevSwOperStatus MUST be failed{4}
- docsDevSwCurrentVer MUST be the current version of software that is operating on the CM

In the case where CM successfully downloads (or detects during download) an image that is not intended for the CM device, the CM MUST behave as specified in [DOCSIS 5], section 10.1 “Downloading Cable Modem Operating Software” and adhere to the following requirements:

- docsDevSwAdminStatus MUST be allowProvisioningUpgrade{2}
- docsDevSwFilename MUST be the filename of the software that failed the upgrade
- docsDevSwServer MUST be the address of the TFTP server containing the software that failed the upgrade process
- docsDevSwOperStatus MUST be other{5}
- docsDevSwCurrentVer MUST be the current version of software that is operating on the CM

In the case where CM determines that the download image is damaged or corrupted, the CM MUST reject the newly downloaded image. The CM MAY re-attempt to download if the maximum number of TFTP download retries (max retries = 3) has not been reached. If the CM chooses not to retry, the CM MUST fall back to the last known working image and proceed to an operational state, generate appropriate event notification as specified in Appendix F, and adhere to the following requirements:

- docsDevSwAdminStatus MUST be allowProvisioningUpgrade{2}
- docsDevSwFilename MUST be the filename of the software that failed the upgrade
- docsDevSwServer MUST be the address of the TFTP server containing the software that failed the upgrade process
- docsDevSwOperStatus MUST be other{5}
- docsDevSwCurrentVer MUST be the current version of software that is operating on the CM

In the case where CM determines that the image is damaged or corrupted, the CM MUST reject the newly downloaded image. The CM MAY re-attempt to download the new image if the maximum number of TFTP download retries (max retries = 3) has not been reached. On the third consecutive failed retry of the CM software download attempt, the CM MUST fall back to the last known working image and proceed to an operational state. In this case, the CM MUST send two notifications, one to notify that the max retry limit has been reached, and another to notify that the image is damaged. Immediately after the CM reaches the operational state the CM MUST adhere to the following requirements:

- docsDevSwAdminStatus MUST be allowProvisioningUpgrade{2}
- docsDevSwFilename MUST be the filename of the software that failed the upgrade
- docsDevSwServer MUST be the address of the TFTP server containing the software that failed the upgrade process
- docsDevSwOperStatus MUST be other{5}
- docsDevSwCurrentVer MUST be the current version of software that is operating on the CM

4.3 Protocol Filters

The CM MUST implement LLC, SNMP Access, and IP protocol filters. The LLC protocol filter entries can be used to limit CM forwarding to a restricted set of network-layer protocols (such as IP, IPX, NetBIOS, and AppleTalk). The IP protocol filter entries can be used to restrict upstream or downstream traffic based on source and destination IP addresses, transport-layer protocols (such as TCP, UDP, and ICMP), and source and destination TCP/UDP port numbers.

CM MUST apply filters (or more properly, classifiers) in an order appropriate to the following layering model; specifically, the inbound MAC (or LLC) layer filters are applied first, then the "special" filters, then the IP layer inbound filters, then the IP layer outbound filters, then any final LLC outbound filters. Note that LLC outbound filters are expected future requirements of the DOCS-CABLE-DEVICE-MIB.

4.3.1 LLC Filter

Inbound LLC filters, from docsDevFilterLLCTable, MUST be applied to layer-2 frames entering the CM from either the CATV MAC interface{2} and/or any CM CPE interface.

The object docsDevFilterLLCUnmatchedAction MUST apply to all (CM) interfaces. The default value of the (CM) docsDevFilterLLCUnmatchedAction MUST be set to accept.

docsDevFilterLLCUnmatchedAction:

If (CM docsDevFilterLLCUnmatchedAction is) set to discard(1), any L2 packet that does not match any LLC filters will be discarded, otherwise accepted. If (CM docsDevFilterLLCUnmatchedAction is) set to accept, any L2 packet that does not match any LLC filters will be accepted, otherwise discarded.

Another way to interpret this is the following:

```

action = UnMatchedAction
Iterate through the table
    if there is a match (packet.protocol = row.protocol)
    {
        reverse the action (accept becomes discard, discard becomes accept)
        apply action to the packet
        terminate the iteration
    }

```

LLC (CM) filters MUST apply to in-bound traffic direction only. Traffic generated from CM MUST not be applied to LLC filters (i.e. ARP requests, SNMP responses).

The CM MUST support a minimum of ten LLC protocol filter entries.

4.3.2 Special Filter

Special filters are IP spoofing filters and SNMP access filters. IP spoofing filters MUST only be applied to packets entering the CM from CMCI interface(s). SNMP access filters are in effect when the CM is not running in SNMPv3 agent mode and can be applied to both CMCI and CATV interfaces.

According to the interface number section of document, CMCI interface is a generic reference to any current or future form of CM CPE interface port technology.

4.3.3 IP Spoofing Filter

DOCSIS 1.1 CM MAY implement IP spoofing filter specified in RFC-2669.

If a CM supports the IP Spoofing filter functionality specified in RFC-2669, the CM MUST adhere to the following requirements:

Implement all MIB objects in the docsDevCpeGroup