Data-Over-Cable Service Interface Specifications

Operations Support System Interface Specification

SP-OSSIv1.1-I03-001220

INTERIM SPECIFICATION

Document Status Sheet

Document Control Number:	SP-OSSIv1.1-I03-001220
Revision History:	I01 – First Interim Release, April 7, 2000 I02– Incorporated ECN OSS-N-00039, OSS-N-00040, OSS-N-00041, OSS-N-00054, July 14, 2000 I03– Incorporated accepted OSSI ECNs see Appendix R. Revisions, December 15, 2000 I03– Reposted 001215 to fix errors: 1) incorporated oss-r-00134 removed, 2) Figure 3 and Figure 4 from oss-n-00096 incorporated, 3) oss-n-00109 incorporated, December 20, 2000
Date:	December 20, 2000
Status Code:	Work in Process ~~Draft~~ Interim ~~Released~~
Distribution Restrictions:	CableLabs® & Members Only CableLabs®, Members, and Vendors Only Public

Key to Document Status Codes

Work in Process	An incomplete document, designed to guide discussion and generate feedback, that may include several alternative requirements for consideration.
Draft	A document in specification format considered largely complete, but lacking review by Members and vendors. Drafts are susceptible to substantial change during the review process.
Interim	A document which has undergone rigorous review by Members and vendors, suitable for use by vendors to design in conformance with, and suitable for field testing.
Released	A stable document, reviewed, tested and validated, suitable to enable cross-vendor interoperability.

Table of Contents

1 Scope and Purpose

1.1 Scope

This [illegible] specification defines the Network Management requirements for sup[illegible]t a DOCSIS 1.1 en[illegible]onment. More [illegible]fical[illegible] th[illegible]spe[illegible]n d[illegible]e S[illegible] v[illegible]co[illegible]w i[illegible]sts [illegible]MI[illegible]2. The RFCs and [illegible]ana[illegible]me[illegible]n[illegible]mat[illegible]e ([illegible]equi[illegible]ments [illegible]eta[illegible] as well [illegible]i[illegible]fac[illegible]ameri[illegible]f[illegible]ring, event notif[illegible]s, etc[illegible]ne[illegible]manag[illegible]t p[illegible]ipals[illegible] as ac[illegible], co[illegible]tion, [illegible], and p[illegible]mance management are incorpora[illegible]d 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 work or the adjective "REQUIRED" means that the item is an absolute requirement for this specification.
"MUST NOT"	This phrase means that the item is an absolute prohibition of this specification.
"SHOULD"	This word of 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 weighted 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 weighted before implementing any behavior described with this label.
"MAY"	This word or the adjective "OPTONAL" 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.

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-2570	Introduction to Version 3 of the internet-standard Network Management
RFC-2571	An Architecture for Describing SNMP Management Frameworks
RFC-2572	Message Processing and Dispatching for the Simple Network Management Protocol (SNMP)
RFC-2573	SNMP Applications
RFC-2574	User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)
RFC-2575	View-based Access Control Model (VACM) for the simple Network Management Protocol (SNMP)
RFC-1905	Protocol Operations for Version 2 of the Simple Network Management Protocol (SNMPv2)
RFC-1906	Transport Mappings for Version 2 of the Simple Network Management Protocol (SNMPv2)
RFC-1907	Management Information Base for Version 2 of the Simple Network Management Protocol (SNMPv2)
RFC-1901	Introduction to Community-based SNMPv2
RFC-1157	A Simple Network Management Protocol

For support of SMIv2 the following IETF SNMP related RFC's MUST be implemented:

RFC-2578	Structure of Management Information Version 2 (SMIv2)
RFC-2579	Textual Conventions for SMIv2
RFC-2580	Conformance Statements for SMIv2
RFC-2786	Diffie-Helman USM Key

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 RFC2571-RFC2576 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 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 MIB's (Community MIB, TARGET-MIB, VACM-MIB, USM-MIB, NOTIFICATION-MIB) 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 MIB's (Community MIB, TARGET-MIB, VACM-MIB, USM-MIB, NOTIFICATION-MIB) are accessible.

d.) CMTS SNMP Coexistence Mode MUST operate with the following requirements/limitations:

- SNMP v1/v2c/v3 Packets are processed as described by RFC2571-RFC2576.
- docsDevNmAccessTable is not accessible. (If the CMTS also support Cable Device MIB)
- Access control and trap destinations are determined by the snmpCommunityTable, Notification MIB, Target MIB, VACM-MIB, and USM-MIB
- Community MIB controls the translation of SNMPv1/v2c packet community string into securityName which select entries in the USM MIB. Access control is provided by the VACM MIB.
- USM MIB and VACM MIB controls SNMPv3 packets
- Trap destinations are specified in the Target MIB and 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-2574 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, CM MUST operate as follows:

- SNMP V1/V2c read-only Access to all MIB variables which are required to be in view during SNMP V1/V2c operation is allowed from the CMCI port. No access is allowed from the RF port.
- SNMP V1/V2c packets are accepted which contain any community string.
- All SNMP V3 packets are dropped.
- Access SHOULD be prohibited to any mib variable that would allow determination of the modem's IP address, like the MIB-2 IpAddrTable

- None of the SNMP V3 MIB's (Community MIB, TARGET-MIB, VACM-MIB, USM-MIB, NOTIFICATION-MIB) are accessible, except that they may be set from the config file.
- None of the elements in the SNMP-USM-DH-OBJECTS-MIB are accessible except that they may be set from the configuration file
- 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 (docsDevNmAccessTable, snmpCommunityTable settings, TLV 34.1 and 34.2) then the CM is in NmAccess mode.
- CM is in SNMP coexistence mode if the CM configuration file contains
  - snmpCommunityTable setting or
  - TLV type 34.1 and 34.2. or both.
  - 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.

c.1) 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 RFC2669
- None of the SNMP V3 MIB's (Community MIB, TARGET-MIB, VACM-MIB, USM-MIB, NOTIFICATION-MIB) are accessible.

c.2) SNMP Coexistence Mode

c.2.1) 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 USM MIB, community MIB and VACM-MIB.

c.2.2) 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 RFC2571-RFC2576.
- docsDevNmAccessTable is not accessible.

- Access control and trap destinations are determined by the snmpCommunityTable, Notification MIB, Target MIB, VACM-MIB, and USM-MIB
- Community MIB control the translation of SNMPv1/v2c packet community string into security name which select entries in the USM MIB. Access control is provided by the VACM mib.
- USM MIB and VACM MIB controls SNMPv3 packets
- Trap destinations are specified in the Target MIB and 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 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.

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 Rm

b. Manager uses DH equation to translate Rm to a public number z

z = g ^ Rm 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 2.3 "VACM Profile". Only VCAM entries specified by the corresponding security name in the CM configuration file will (MUST) be active.

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.

1) CM generates a random number xa for each row populated in the usmDHKickstartTable which has a non zero length usmDHKickstartSecurityName and usmDHKickstartMgrPublic.

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

c = g ^ xa 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”

3) CM calculate shared secret sk where sk = z ^xa mod p

4) 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.

5) 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 securityname 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 securityname 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.

In addition to RFC-2575, 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)
vacmSecurityName	'docsisOperator'
vacmGroupName	'docsisOperator'
vacmSecurityToGroupStorageType	permanent
vacmSecurityToGroupStatus	active

-- 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 object. It is 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 instantiated such object and MUST respond with the appropriate error/exception condition. (e.g., no such object for SNMPv2c)

1. 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 instantiated 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 instantiated 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

MIB	Applicable Device(s)
IETF Proposed Standard RFC version of Qos MIB, **"draft-ietf-ipcdn-qos-mib-04.txt"**	CM and CMTS
IETF Proposed Standard RFC version of BPI+ MIB, "**draft-ietf-ipcdn-bpiplus-mib-03.txt"**	CM and CMTS
IETF Proposed Standard RFC version of USB MIB, "**draft-ietf-xxxx-xxxx-xxxx-00.txt"**	CM only
IETF Proposed Standard RFC version of BPI MIB, "**draft-ietf-ipcdn-mcns-bpi-mib-01.txt"**	CM and CMTS
IETF Proposed Standard RFC version of Subscriber Management MIB, "**draft-ietf-ipcdn-subscriber-mib-02.txt"**	CMTS only

3.2 IETF RFCs

MIB	Applicable Device(s)

RFC-2933: Internet Group Management Protocol MIB	CM and CMTS
RFC-2669: DOCSIS Cable Device MIB	CM and CMTS
RFC-2670: Radio Frequency (RF) Interface MIB	CM and CMTS
RFC-2665: Ethernet Interface MIB.	CM and CMTS
RFC-2233: The Interfaces Group MIB using SMIv2	CM and CMTS
RFC-1493: Bridge MIB	CM and CMTS
RFC-2011: SNMPv2 Management Information Base for the Internet Protocol using SMIv2	CM and CMTS
RFC-2013: SNMPv2 Management Information Base for the User Datagram Protocol using SMIv2	CM and CMTS
RFC-1907: Management Information Base for Version 2 of the Simple Network Management Protocol (SNMPv2)	CM and CMTS
RFC-2786: Diffie-Helman USM Key	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.

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 specify by Appendix A), and SHOULD implement the other non-mandatory required objects.

3.3.3 Requirements for RFC-2670

RFC-2670 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.

3.3.4 Requirements for RFC-2233

RFC-2233 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 LAN 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-2233]) 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.

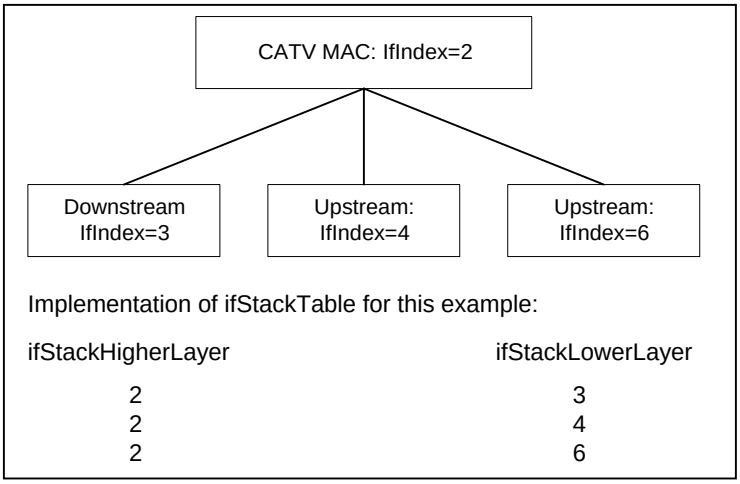


Figure 1: Ifindex Example for CMTS

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-2233]) 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:

Interfaces	Type
1	primary CPE interface
2	CATV-MAC
3	RF-down
4	RF-Up
4+n	Other interfaces

If CM has more than one CPE interface, then the vendor MUST define which of (n) CPE interfaces is the primary CPE interface; regardless, the Primary CPE interface MUST be Interface number 1.

The secondary CPE, and other interfaces, will start at 5.

DOCSIS CM may have multiple interfaces. If filter(s) are applied to CM IfIndex 1, the same filter(s) MUST also applied to each CPE interface; however, filters are never used to limit traffic between multiple CPE interfaces within the CM.

3.3.5 Interface MIB and Trap Enable

Interface MIB and Trap Enable specified in RFC-2233 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.

3.3.9 Requirements for RFC-2013

RFC-2013 MUST be implemented by DOCSIS 1.1 compliant CMTS and CMs.

The UDP group in the RFC-2013 MUST be implemented.

3.3.10 Requirements for RFC-1907

RFC-1907 MUST be implemented by DOCSIS 1.1 compliant CMTS and CMs.

3.3.10.1 The System Group

The System Group from RFC-1907 MUST be implemented. See Section 4.2.1 for sysObjectID requirements.

3.3.10.2 The SNMP Group

The SNMP Group from RFC-1907 MUST be implemented.

3.3.11 Requirements for "draft-ietf-ipcdn-qos-mib-04.txt"

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

3.3.12 Requirements for "draft-ietf-ipcdn-igmp-mib-01.txt" have been deleted

"draft-ietf-ipcdn-igmp-mib-01.txt" requirements 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 I. 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 "draft-ietf-ipcdn-bpiplus-mib-03.txt" BPI+ MIB

"draft-ietf-ipcdn-bpiplus-mib-03.txt" MUST be implemented by DOCSIS 1.1 compliant CMTS and CMs as specified in Appendix A.

3.3.15 Requirements for "draft-ietf-xxxx-xxxx-xxxx-00.txt" USB MIB

"draft-ietf-xxxx-xxxx-xxxx-00.txt" MUST be implemented by DOCSIS 1.1 compliant CMs that support USB.

3.3.16 Requirements for "draft-ietf-ipcdn-subscriber-mib-02.txt" Subscriber Management 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 Diffie-Helman USM Key

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 "draft-ietf-ipcdn-mcns-bpi-mib-01.txt" BPI+ MIB

"draft-ietf-ipcdn-mcns-bpi-mib-01.txt" MUST be implemented by DOCSIS 1.1 compliant CMs as specified in Appendix A.

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 [MCNS 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 translating 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-1905], 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 OID's 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 MIB's 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

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 I: Interface Reset. 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.

Case ii: 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 iii: 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-2233 section 3.1.6).

3.6 Config File Element - docsisV3NotificationReceiver

This config file element specifies a Network Management Station that will receive notifications from the modem when it is in Coexistence mode. Up to 10 of these elements may be included in the configuration file.

Here is the format of this element:

Definition of fields of docsisV3NotificationReceiver Element

All multi-byte fields have the most significant

bytes first in the field.

This TLV (Assigned type XX) consists of several Sub-TLV's inside of the TLV config file element:

Sub-TLV XX.1 – IP Address of trap receiver, in binary

IP Address	4 bytes	IP Address of the trap receiver, in binary.

Sub-TLV XX.2 – UDP Port number of the trap receiver, in binary

Port	2 bytes	UDP Port number of the trap receiver, in binary. (If not present, the default value 162 is used)

Sub-TLV XX.3 – Type of trap sent by the modem (Note 2)

Trap type 2 bytes
1 = SNMP v1 trap in an SNMP v1 packet
2 = SNMP v2c trap in an SNMP v2c packet
3 = SNMP inform in an SNMP v2c packet
4 = SNMP v2c trap in an SNMP v3 packet
5 = SNMP inform in an SNMP v3 packet

Sub-TLV XX.4 – Timeout, in milliseconds, used for sending inform

Timeout 2 bytes 0-65535.

Sub-TLV XX.5 – Number of retries when sending an inform, after sending the inform the first time.

Retries 2 bytes 0-65535.

Sub-TLV XX.6 - Notification Filtering Parameters

If this Sub-TLV is not present, the notification receiver will receive all notifications generated by the SNMP agent.

Filter OID ASN.1 formatted Object Identifier of the snmpTrapOID value that identifies the notifications to be sent to the notification receiver. This notification and all below it will be sent. <z> is the length, in bytes of the ASN.1 encoding. This field starts with the ASN.1 Universal type 6 (Object Identifier) byte, then the ASN.1 length field, then the ASN.1 encoded object identifier components.

Sub-TLV XX.7 - Security Name to use when sending SNMP V3 Notification

This Sub-TLV is not required for Trap type = 1, 2, or 3 above. If it is not supplied for a Trap type of 4 or 5, then the V3 Notification will be sent in the noAuthNoPriv security level using the security name "@config". (Note 2)

SecurityName The V3 Security Name to use when sending a V3 Notification. Only used if Trap Type is set to 4 or 5. This name must be a name specified in

a Config File TLV Type 34 as part of the DH Kickstart procedure. The notifications will be sent using the Authentication and Privacy Keys calculated by the modem during the DH Kickstart procedure.

Notes:

(1) Upon receiving one of these TLV elements, the modem SHALL make entries to the following tables in order to cause the desired trap transmission: snmpNotifyTable, snmpTargetAddrTable, snmpTargetParamsTable, snmpNotifyFilterProfileTable, snmpNotifyFilterTable, snmpCommunityTable, usmUserTable, vacmSecurityToGroupTable, vacmAccessTable, and vacmViewTreeFamilyTable

(2) Trap Type: The community String for traps in SNMP V1 and V2 packets SHALL be "public". The Security Name in traps and informs in SNMP V3 packets where no security name has been specified SHALL be "@config and in that case the security level SHALL be noAuthNoPriv.

(3) Filter OID: SNMP V3 allows the specification of which Trap OID's are to be sent to a trap receiver. The filter OID in the config element specifies the OID of the root of a trap filter sub-tree. All Traps with a Trap OID contained in this trap filter sub-tree SHALL be sent to the trap receiver.

(4) Config file TLV number: The type field of this TLV SHALL be (TBD).

(5) The config file MAY also contain TLV MIB elements that make entries to any of the 10 tables listed in note 1. These TLV MIB elements SHALL NOT use index columns that start with the characters "@config".

(6) This TLV element SHALL be processed only if the modem has entered SNMP V3 Coexistence Mode during processing of the config file.

3.6.1 Mapping of TLV fields into created SNMP V3 Table rows

These tables show how the fields from the Config file TLV element are placed into the SNMP V3 tables.

The TLV fields are shown below as:

<IP Address> - A 32 bit IP address of a notification receiver

<Port> - A 16 bit UDP Port number on the notification receiver to receive the notifications.

<Trap type> - Defines the notification type as explained above.

<Timeout> - 16 bit timeout, in milliseconds to wait before sending a retry of an Inform Notification.

<Retries> - 16 bit number of times to retry an Inform after the first Inform transmission

<Filter OID> - The OID of the snmpTrapOID value that is the root of the MIB subtree that defines all of the notifications to be sent to the Notification Receiver.

<Security Name> - The security name specified on the TLV element, or "@config" if not specified.

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.

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

snmpNotifyTable (RFC2573 - 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	<Default value from MIB>	<Default value from MIB>
snmpNotifyRowStatus	Active (1)	Active (1)

snmpTargetAddrTable - Create 1 row for each TLV element in the config file

snmpTargetAddrTable (RFC2573 - 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	snmpUDPDomain = 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	<don't care>
snmpTargetAddrRowStatus	active (1)

snmpTargetAddrExtTable - Create 1 row for each TLV element in the config file

snmpTargetAddrExtTable (RFC2576 - 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

snmpTargetParamsTable - Create 1 row for each TLV element in the config file

a. Create it this way if <Trap type> = 1, 2, or 3, or if the <Security Name> Field is zero length

snmpTargetParamsTable (RFC2573 - 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: SnmpMessageProcessingModel	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	<Don't care>
snmpTargetParamsRowStatus	active (1)

b. Otherwise create it this way.

snmpTargetParamsTable (RFC2573 - 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: SnmpMessageProcessingModel	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	<Don't care>
snmpTargetParamsRowStatus	active (1)

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

snmpNotifyFilterProfileTable (RFC2573 - 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	<Don't care>
snmpNotifyFilterProfileRowStatus	active (1)

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

snmpNotifyFilterTable (RFC2573 - 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	<don't care>
snmpNotifyFilterRowStatus	active (1)

snmpCommunityTable - Create 1 row with fixed values if 1 or more TLV's are present

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

snmpCommunityTable (RFC2576 - 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	<Don't Care>
snmpCommunityStatus	active (1)

usmUserTable - Create 1 row with fixed values, if 1 or more TLV's are 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.

usmUserTable (RFC2574 - 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	<don't care>
usmUserStatus	Active (1)

vacmSecurityToGroupTable - Create 3 rows with fixed values, if 1 or more TLV's are 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>

vacmSecurityToGroupTable (RFC2575 - 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	<don't care>	<don't care>	<don't care>
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.

vacmAccessTable - Create 3 rows with fixed values, if 1 or more TLV's are 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>

vacmAccessTable (RFC2575 - 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	<Zero length octet	<Zero length octet

	string>	string>	string>
vacmAccessWriteViewName	<Zero length octet string>	<Zero length octet string>	<Zero length octet string>
vacmAccessNotifyViewName	"@config"	"@config"	"@config"
vacmAccessStorageType	<don't care>	<don't care>	<don't care>
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.

vacmViewTreeFamilyTable - Create 1 row with fixed values if 1 or more TLV's are 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>

vacmViewTreeFamilyTable (RFC2575 - 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	<don't care>
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. SCNs MUST be unique within an MSO's operation, and a descriptive SCN might be something like *PrimaryUp*, *GoldTCPUp, VoiceDown,* or *BronzeUDPDown* 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 File[1] 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 MUST 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 *PrimaryDown*) that are known to the CMTS if these Service Flows are to be visible via the billing interface to external systems.

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, provides 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 the provisioned SCN templates) into QoS Parameter Sets needed to establish active Service Flows for the CM. Internally, each active Service Flow is identified by a 32-bit SFID assigned by the CMTS to a specific CM. For billing purposes, however, the SFID is not sufficient as the only identifier of a Service Flow because the billing system cannot distinguish the 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

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

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.

4.1.2 Requirements for Subscriber Usage Billing Records

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. It is expected that the billing record collection interval could be as short as 10 minutes. The following are the requirements for processing and transmitting Subscriber Usage Billing Records:

1. 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.
2. Subscriber billing records MUST be organized by CM MAC address (but not necessarily sorted). The Subscriber's current CM IP address and all current CPE IP addresses MUST be present in the billing record for the Subscriber.
3. Subscriber billing records MUST have entries for each active Service Flow (identified by SFID and Service Class Name) used by the CM during the collection interval. This includes all currently active Service Flows as well as all active 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, eventhough it was logged by the CMTS. It MUST be possible to distinguish continuing Service Flows from deleted Service Flows in the billing records.
4. It MUST be possible to identify the Service Flow direction as upstream or downstream without reference to the SCN. 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 MUST also be collected for each downstream Service Flow. 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 ranges and re-registers for any reason (such as CPE power failure).
5. 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.
6. 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 *Network Data Management – Usage, Version 1.1* (NDM-U 1.1) as extended for XML format Cable Data Systems Subscriber Usage Billing Records MUST be used. See Appendix C for the *Cable Data Systems Subscriber Usage Billing Records* standard submission to ipdr.org and an example IPDR XML billing file. See also www.ipdr.org for more information on the IPDR standards.
7. To improve the performance of storage and transmission of the IPDR 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 IPDR XML format billing file will compress on the order of 15:1 or better. See also www.gnu.org/software/gzip for more information.
8. 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.

9. 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 record formatter MUST begin construction of a new billing file when it detects that the current billing file has been deleted. The record formatter MUST protect the billing file until it is deleted by the collection system. The record formatter MUST support a minimum 15 minute collection interval. If the collection system attempts to retrieve the billing file before the record formatter has completed building it, the collection system will determine that the billing file does not exist yet in the formatter's FTP file system. In this case, the collection system MUST back off and try again at a later time. If multiple retrievals of the billing file by multiple collectors is desired, then the last collector must delete the billing file. How this is coordinated between the multiple collectors is beyond the scope of this specification.
10. 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. A locally administered FTP userid and password MUST be provided to control access to the billing file. A locally administered 40-bit single DES encryption key MUST be provided to encrypt the compressed billing file. The DES encryption key MUST be provided solely for the purpose of encrypting the billing file and MUST NOT be shared with any other application. Authorization and encryption are optional for the CMTS operator, therefore authorization MUST be turned off when the FTP userid and password are null (aka anonymous FTP) and encryption MUST be turned off when the DES encryption key is null.

4.1.3 Billing Collection Interval

Subscriber Usage Billing Records report the absolute traffic counter values for each active Service Flow used by a Cable Modem (Subscriber) during 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 (Tprev) and the creation of the current billing file (Tnow). See Figure 1 below. There are two kinds of Service Flows that are reported in the current billing file: 1) SFs that are still active at the time the billing file is created and 2) active SFs that have been deleted and logged during the collection interval. A provisioned or admitted state SF that was deleted before it became active is not recorded in the billing file, eventhough it was logged by the CMTS.

A currently active SF is recorded using Tnow as the timestamp for its counters and are identified in the IPDR UE element as type=Interim. Deleted SFs that have a deletion time (Tdel) later than Tprev are the only ones recorded in the current billing file (a deleted SF is reported only once). A deleted SF is recorded using its Tdel from the log as the timestamp for its counters and is identified in the IPDR UE element as type=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 may be used to construct an accurate timebase over sequences of billing files.

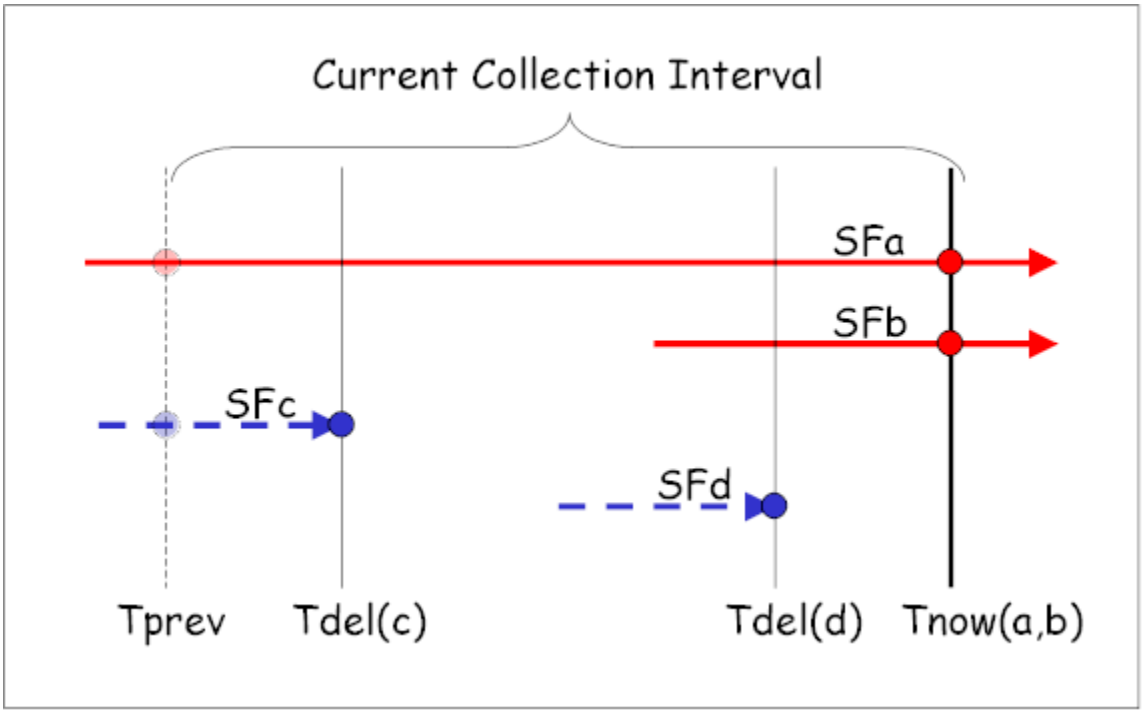


Figure 1 - Collection Interval Time Base

In the example in Figure 1 there are four Service Flows recorded for a Subscriber in the current billing file being created at Tnow. SFa is a long running SF that was active during the previous collection interval (it has the same SFID in both the current and the previous billing files). SFa was recorded using Tprev in the previous billing file and is recorded using Tnow in the current file. SFb is an active SF that was created during the current collection interval. SFb is recorded for the first time using Tnow in the current file. SFc is a deleted SF that was active during the previous collection interval but was deleted and logged during the current collection interval. SFc was recorded using Tprev in the previous billing file and is recorded using the logged Tdel(c) in the current file. SFd is a deleted SF that was both created and deleted during the current collection interval. SFd is recorded using the logged Tdel(d) in the current billing file only.

4.1.4 Billing File Retrieval Model

Billing files are built by the record formatter and retrieved by the collection system in a decoupled manner. There is no explicit signaling protocol between them and no prior arrangement regarding the frequency of billing collection. The formatter is responsible for creating the current billing file and placing 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 sucessfully retrieved the billing file, it removes the file via FTP DELETE in the formatter's FTP file system. The formatter monitors the existence of the billing file in it's FTP file system and when it no longer exists, the formatter begins to create the next billing file. The formatter must finish constructing the next billing file and have it ready for retrieval in it's FTP file system within 10 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 will back off and return at a later time to try again.

Note that if the collection system fails for any reason, the formatter will retain the "current" billing file until such time as the collection system returns to retrieve the file. In this case, eventhough 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. 10 minutes) and retrieve the new billing file with the current timestamps. The absolute values of the counters will always be preserved by the CMTS, 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 timebase 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.

4.1.5 Billing File Security Model

The billing file security model has two components: 1) authorization to control access to the billing file in the formatter's FTP file system and 2) encryption to ensure the privacy of the billing data and to guarantee its integrity while it is both stored and in transit.

Authorization is provided by the standard FTP userid and password mechanism. The collection system needs read and delete access permissions to the billing file in the formatter's FTP file system. The collection system's userid and password are locally administered by the formatter's FTP implementation.

Encryption is provided by a 40-bit single DES encryption algorithm using a locally administered "shared secret"encryption key in the formatter. The billing file is first compressed and then encrypted before it is finally stored in the formatter's FTP file system. This provides end-to-end security during any intervening storage or transmission steps between the formatter and the billing system. Note that intermediate collection systems that store or transmit the billing file may or may not have the encryption key if the operator so chooses. It is only required that the billing system or mediation system responsible for parsing the billing records must have the shared encryption key. Also, the billing encryption key must be provided soley for the use of the billing interface and not be shared with other applications in the formatter host.

Note that the formatter must provide authorization and encryption capabilities, but the operator may not utilize them. In this case the formatter must turn off authorization when the userid and password are null, and must turn off encryption when the billing file encryption key is null.

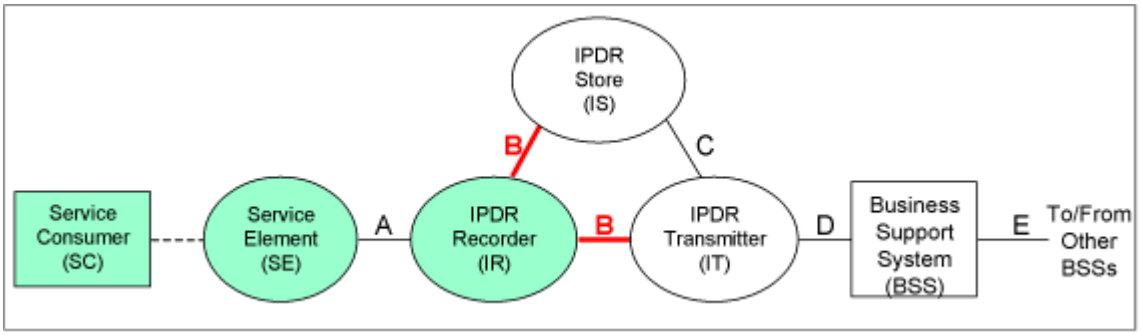
4.1.6 IP Detail Record (IPDR) Standard

The IPDR standards group (see www.ipdr.org) has defined a generic model for using XML 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 IPDR XML syntax. See Appendix C for the DOCSIS OSSI standards submission to ipdr.org for the *Cable Data Systems Subscriber Usage Billing Record*. Appendix C also contains an example IPDR XML format Subscriber Usage Billing file and the IPDR standard XML Data Type Definition (DTD) file that describes the IPDR syntax.

4.1.6.1 IPDR Network Model

The IPDR Network Model is given in the ipdr.org standard *NDM-U 1.1* and is portrayed in Figure 2 below.

Figure 2 - IPDR Basic Network Model (ref. NDM-U 1.1 from www.ipdr.org)



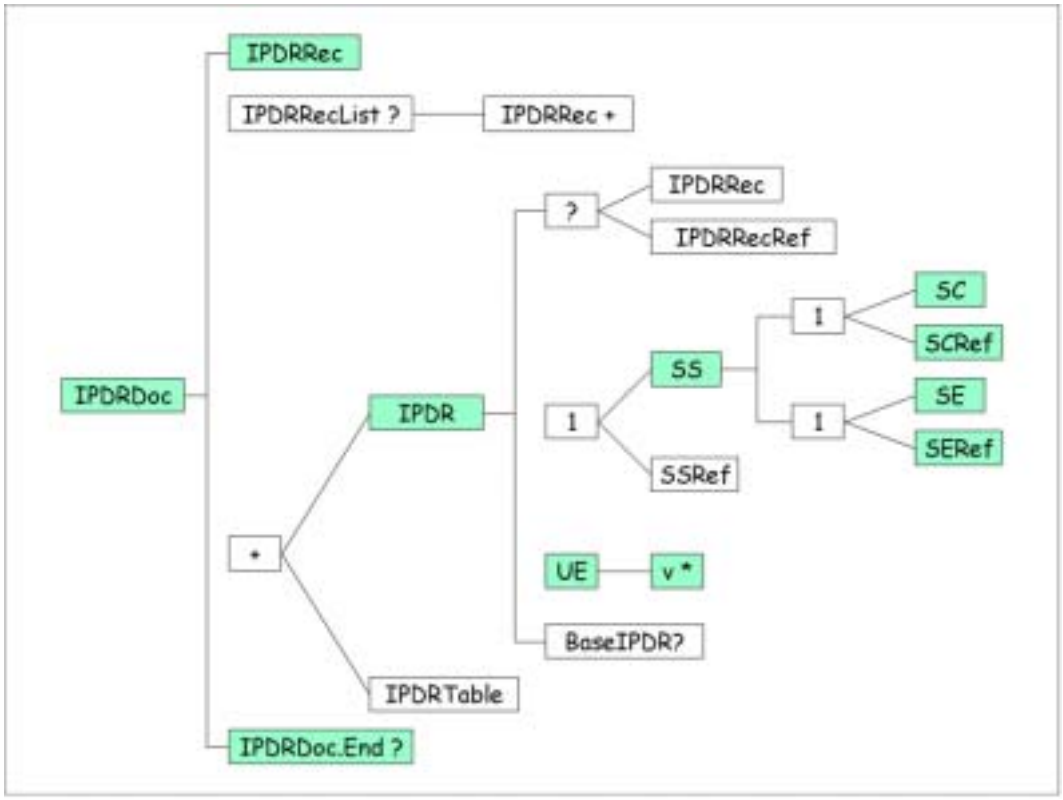
In this model, the Service Consumer (SC**)** is the Cable Data Service Subscriber identified by their Cable Modem MAC address, current CM IP address, and current CPE IP addresses. The Service Element (SE) is the CMTS identified by its host name and IP address. The IPDR Recorder (IR) is the billing record formatter function that creates the IPDR XML format billing records from the internal counters maintained by the CMTS for each Subscriber and active Service Flow. The IPDR Recorder is a function that may be implemented within the CMTS or hosted on another platform such as an Element Management System (EMS). The IPDR Store (IS) and the IPDR Transmitter (IT) represent the billing record collectors that retrieve the billing records from the IPDR Recorder. In this specification the IS or IT would retrieve the billing file from the IR on a collection cycle determined by the IT or IS. The A-interface is not specified by the IPDR standard because it is an internal interface between the SE and the IR. However, the B-interface is specified by the IPDR standard as a file of IPDR records formatted according to the IPDRdoc XML Data Type Definition (DTD) file (see Appendix C). The B-interface MUST be implemented using the

Cable Data Systems Subscriber Usage Billing Record submission to the IPDR standard as defined in Appendix C. The C-, D-, and E-interfaces are beyond the scope of this specification.

4.1.6.2 IPDR Record Structure

The IPDR standard specifies the IPDRDoc record structure. The IPDRDoc XML DTD (see Appendix C) defines the hierarchy of elements as shown in Figure 3 below.

Figure 2 - IPDR XML Element Hierarchy (ref. NDM-U 1.1 from www.ipdr.org)



The highlighted elements are the only ones used by the *Cable Data Systems Subscriber Usage Billing Record* (see Appendix C). These elements are described below.

1. *IPDRDoc* is the outermost element that describes the IPDR billing file itself. It defines the version of the specification and the timestamp for the file. An IPDRDoc is composed of multiple IPDR records.
2. *IPDRRec* describes the IPDR Recorder (IR) from the network model in Figure 1. This element identifies the billing record formatter by the fully qualified host name of the CMTS or the EMS where the formatter resides.
3. *IPDR* describes a single Subscriber Usage Billing Record. This element identifies the time of the billing collection event. The IPDR is further structured into a Service Session (SS) element and a Usage Event (UE) element. While the 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.

4. *SS* describes the Service Session which in a CMTS is an active Service Flow. This element identifies the Service Flow by its Service Class Name (SCN). The Service Session is further structured into the Service Consumer (SC), which is the Subscriber, and the Service Element (SE), which is the CMTS, that are related by this Service Session.
5. *SC* describes the Service Consumer from the network model in Figure 2. This is the Subscriber identified by its Cable Modem MAC address and its current IP address plus the current IP addresses of all CPEs using the Service Flows during the collection interval. Since several IPDRs will describe the Service Flows used by the Subscriber, the SC occurs just once. The common SC is then is referenced in each subsequent IPDR by an *SCRef* element.
6. *SE* describes the Service Element from the network model in Figure 2. This is the CMTS identified by its host name and its IP address and includes the sysUpTime for the CMTS. Since all the IPDRs in a given IPDRDoc are for a single CMTS, the SE occurs just once at the top of the document. The common SE is then referenced in each subsequent IPDR by an *SERef* element.
7. *UE* describes the Usage Event details. This element identifies the counters that are associated with the Service Flow and whether the Service Flow has terminated (type=Stop) or is continuing (type=Interim). A set of <v…/> elements identify the usage attribute values including the *SFID* and the actual 64-bit CMTS counters for the Service Flow in decimal format ASCII notation. The SFID facilitates the correlation of sequential counter sets for the same Service Flow from one IPDRDoc file to the next. Note well that the direction of a Service Session in the IPDR model is from the SC's frame of reference -- this means that a Service Flow is seen from the CM's point of view. Thus, downstream Service Flows in the CMTS are received by the CM while upstream Service Flows in the CMTS are sent by the CM. Also, note that since a Service Flow is unidirectional a UE may either have send-counters or receive-counters, but not both.The usage attribute value names for downstream Service Flows are *recvPkts*, *recvOctets*, *recvSLADropPkts,* and *recvSLADelayPkts*. The usage attribute value names for upstream Service Flows are *sendPkts* and *sendOctets*. Note that there are no drop or delay counters in the upstream direction because these are not known to the CMTS.

IPDRDoc.End is 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.

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 and CMTS SHOULD support software revision and operational parameter configuration interrogation.

The CM MUST (and the CMTS SHOULD) include at least the hardware version, Boot ROM image version and vendor name in the sysDescr object (from [RFC-1907]). To avoid duplication of management information, the CM (and CMTS) SysDescr MUST NOT contain the software version information; however, if the CMTS does not implement the optional docsDevSwCurrentVers MIB object, then the CMTS SysDescr MUST contain the software version information.

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>

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, and Boot ROM version 1.4

MUST appear as following:

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

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

The CM MUST (and the CMTS is optional) implement the docsDevSwCurrentVers object ([RFC 2669]) to report the current software version.

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.

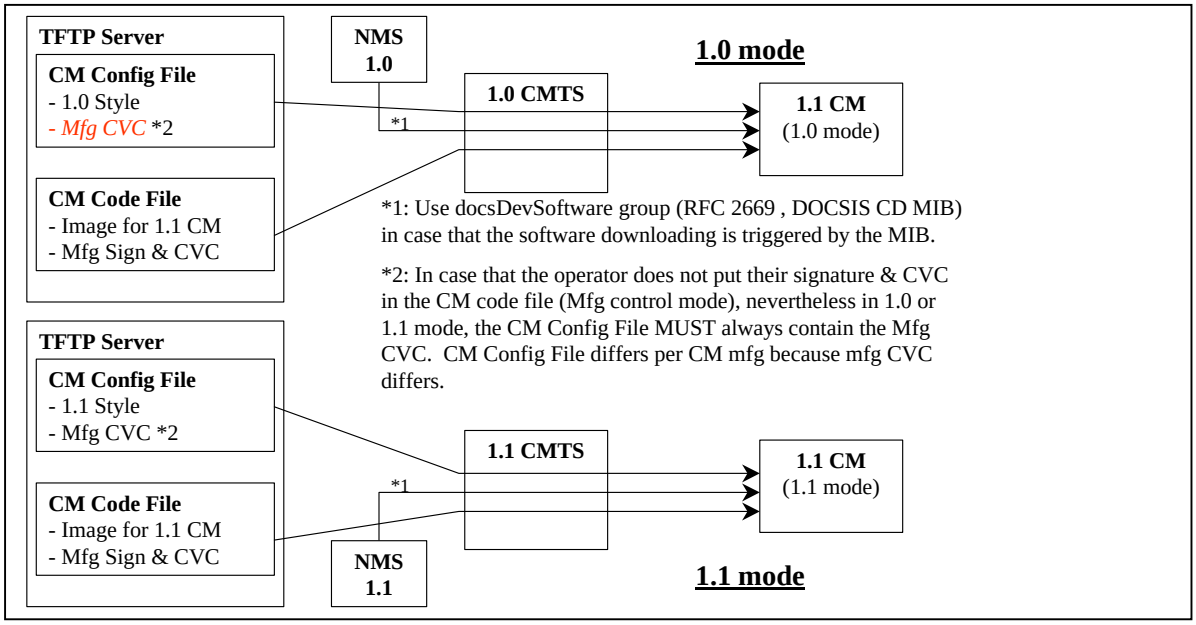
4.2.3 Secure Software Upgrades

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

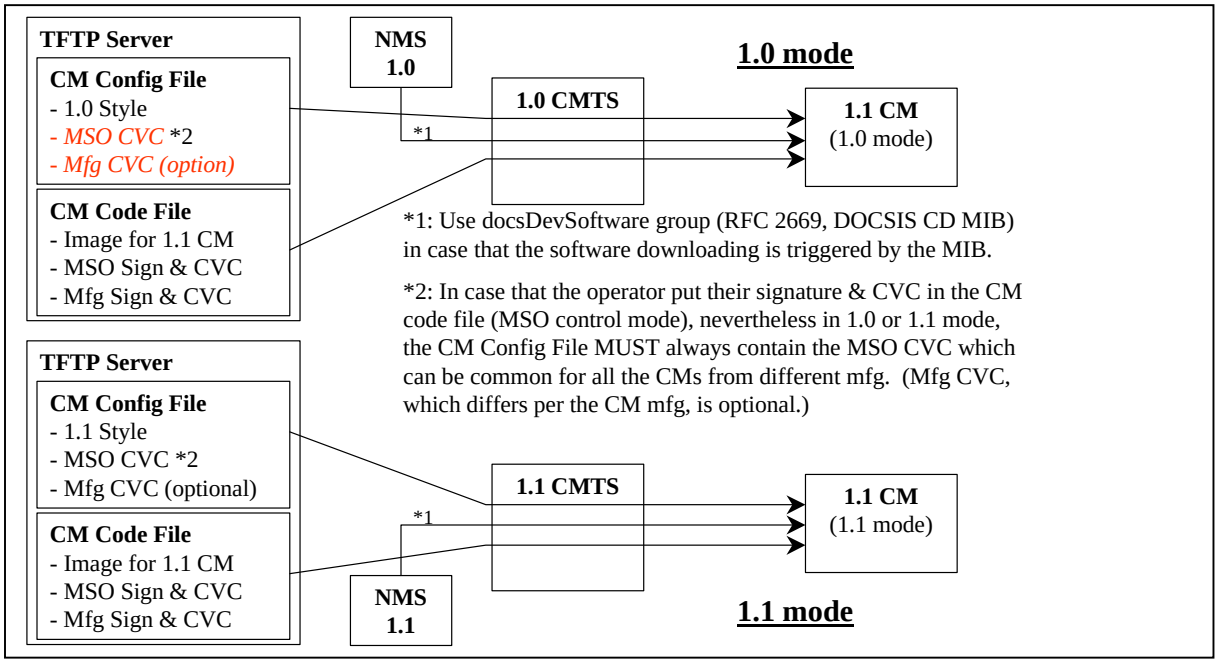
DOCSIS 1.1 CM MUST use secure software upgrade mechanism to perform software upgrade regardless of what DOCSIS CMTS version (1.0 or 1.1) it is connected to. When a 1.1 CM is connected to a 1.1 CMTS, the 1.1 CM MUST operate in either DOCSIS 1.1 mode or DOCSIS 1.0 mode. When a 1.1 CM is connected to a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

1. Manufacture control scheme:



2. 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.

docsDevSwAdminStatus MUST persist across reset/reboots until over-written from an SNMP manager or via the CM configuration file.

The default state of docsDevSwAdminStatus MUST be allowProvisioning Upgrade{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.

If a CM suffers a loss of power or resets during 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 retries (max retries = 3) resulting from multiple loss of powers or resets during either SNMP-initiated upgrade or configuration-file-initiated upgrade, the CM MUST behave as specified in [MCNS5]; 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 exhausts the required number of TFTP retries by issuing a total of 16 consecutive retries, the CM MUST behave as specified in [MCNS5] and then the CM MUST fall back to last known working image and proceed to an

operational state and adhere to the following requirements:

- docDevSwAdminStautus 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

After the CM has completed the SNMP-initiated secure software upgrade, the CM MUST behave as specified in [MCNS5] and MUST reboot and become operational with the correct software image and after the CM is registered, it MUST adhere to the following requirements:

- set its docsDevSwAdminStatus to ignoreProvisioningUpgrade{3}
- set its docsDevOperStatus to completeFromMgt{3}
- reboot

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

After the CM has completed the configuration-file-initiated secure software upgrade, the CM MUST behave as specified in [MCNS5] and MUST reboot and become operational with the correct software image. 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

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 (refer to [MCNS 5], section 10.1 “Downloading Cable Modem Operating Software”):

- DocsDevSwAdminStatus MUST be allowProvisioingUpgrade{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 MAX number of TFTP sequence retry has not been reached. If the CM chooses not to retry and the MAX number of TFTP sequence retry has not been reached, 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:

- DocsDevSwAdminStauts 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 MAX number of TFTP sequence retry has not been reached. On the 16th consecutive failed 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 is required to send two notifications, one to notify that the MAX TFTP 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:

- DocsDevSwAdminStauts 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, IpSpoofing, 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 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 docsDevFilterLLCUnmachedAction is) set to discard(1), any L2 packet that does not match any LLC filters will be discarded, otherwise accepted. If (CM docsDevFilterLLCUnmachedAction 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
       terminiate 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
- Default value of docsDevCpeIpMax = -1

4.3.3.1 DocsDevCpeIpMax, TLV type-18 (Maximum Number of CPEs) and there relationship with FilterCpeTable

The docsDevCpeIpMax value specifies the MAX number of docsDevCpeTable rows, and the TLV type-18 (Maximum Number of CPEs) value specifies the MAX number of CPE MAC address CM is allowed to bridge/forward. When TLV type-18 value is less than docsDevCpeIpMax value, the TLV type-18 value establishes the MAX number of docsDevCpeTable rows; otherwise, the docsDevCpeIpMax value establishes the MAX number of docsDevCpeTable rows.

Handling of configuration file containing both TLV type-18 value (>1) and docsDevCpeIPMax value (>1):

If docsDevCpeIpMax value is greater than TLV type-18 value, CM MUST limit the number of rows in the docsDevCpeTable to the TLV type-18 value.

Handling of configuration file with docsDevCpeIpMax value but no TLV type-18 value:

> **The [MCNS 5] (TLV type-18) requirement states that if TLV type-18 is not specified in the configuration file, the CM MUST default Maximum Number of CPEs to 1 (refer to section C.1.1.7 of MCNS5).**

If TLV type-18 is not supplied in the CM configuration file and docsDevCpeIpMax value is >= 0, CM MUST limit the number of row(s) in the docsDevCpeTable to 1.

4.3.4 SNMP Access Filter

The SNMP access filters MUST be applied to SNMP packets entering from any interfaces and destined for the CM. SNMP access filter MUST be applied after IP spoofing filters for the packets entering the CM from the CMCI interface. Since SNMP access filter function is controlled by docsDevNmAccessTable, SNMP access filter is available and applies only when the CM is in SNMP v1/v2c NmAccess mode.

When CM is running in SNMP Coexistence mode SNMP access MUST be controlled and specified by MIB Objects in [RFC2571-RFC2576].

docsDevNMAccessIP and docsDevNMAccessIpMask :

The device that implement docsDevNMAccessTable MUST apply the following rule in order to determine whether to permit SNMP access from a SrcIpAddr:

The (CMTS/CM) NmAccessIpMask MUST be set to 0.0.0.0 in order to allow any NMS access. The (CMTS/CM) default value of the docsDevNMAccessIpMask MUST be set to '0.0.0.0'.

```
if ((NmAccessIp AND NmAccessIpMask)) == (SrcIpAddr AND NmAccessIpMask))
        Permit the access from SrcIpAddr;
    else
        Do NOT permit the access from SrcIpAddr
```

Allow any NMS:

```
NmAccessIP      = any IP address
NmAccessIpMask  = 0.0.0.0
```

Allow single NMS:

```
NmAccessIP      = an IP address
NmAccessIpMask  = 255.255.255.255
```

Allow group of IP:

```
NmAccessIP      = IP address of the IP subnet
NmAccessIPMask  = Netmask of the subnet
```

Not allow any IP:

```
NmAccessIP      = 0.0.0.0
NmAccessIPMask  = 255.255.255.255
```

4.3.5 IP Filter

The object docsDevFilterIPDefault MUST apply to all (CM) interfaces. DOCSIS 1.1 compliant CM MUST support a minimum 16 IP filters.

4.4 Fault Management

The goals of fault management are remote monitoring/detection, diagnosis, and correction of problems. Network Management operators rely on the ability to monitor and detect problems(s) (such as ability to trace and identify faults, accept and act on error-detection events), as well as the ability to diagnose and correct problem(s) (such as perform a sequences of diagnostic tests, correct faults, and display/maintain event logs.)

This section defines what MUST be available to support remote monitoring/detection, diagnosis and correction of problems.

4.4.1 SNMP Usage

In the DOCSIS environment, the goals of fault management are the remote detection, diagnosis, and correction of network problems. Therefore, the CM MUST support SNMP management traffic across both the CPE and CATV MAC interfaces regardless of the CM's connectivity state. CM SNMP access may be restricted to support policy goals. CM installation personnel can use SNMP queries from a station on the CMCI side to perform on-site CM and diagnostics and fault classification (note that this may require temporary provisioning of the CM from a local DHCP server). Further, future CMCI side customer applications, using SNMP queries, can diagnose simple post-installation problems, avoiding visits from service personnel and minimizing help desk telephone queries.

Standard MIB-II support MUST be implemented to instrument interface status, packet corruption, protocol errors, etc. The transmission MIB for Ethernet-like objects [RFC-2665] MUST be implemented on each cable device (CMTS/CM) Ethernet and Fast Ethernet port. Each cable device (CMTS/CM) MUST implement the ifXTable [RFC-2233] to provide discrimination between broadcast and multicast traffic.

The cable device (CMTS/CM) MUST support managed objects for fault management of the PHY and MAC layers. The RFC-2670 MIB includes variables to track PHY state such as codeword collisions and corruption, signal-to-noise ratios, transmit and receive power levels, propagation delays, micro-reflections, in channel response, and Sync loss. The RFC-2670 MIB also includes variables to track MAC state, such as collisions and excessive retries for requests, immediate data transmits, and initial ranging requests.

For fault management at all layers, the cable device (CMTS/CM) MUST generate replies to SNMP queries (subject to policy filters) for counters and status. The cable device (CMTS/CM) MUST send SNMP traps to one or more trap NMSs (subject to policy), and MUST send SYSLOG events to a SYSLOG server (if a SYSLOG server is defined).

When the cable device (CM) is operating in SNMP v1/v2c NmAccess mode it MUST support the capability of sending traps as specify by the following MIB object (proposed MIB extension to the docsDevNmAccess table):

```
DocsDevNmAccessTrapVersion OBJECT-TYPE
    SYNTAX      INTEGER {
        DisableSNMPv2trap(1),
        EnableSNMPv2trap(2),
}
    MAX-ACCESS  read-create
    STATUS      current
```

DESCRIPTION

"Specifies the TRAP version that is sent to this NMS. Setting this object to DisableSNMPv2trap (1) causes the trap in SNMPv1 format to be sent to particular NMS. Setting this object to EnableSNMPv2trap (2) causes the trap in SNMPv2 format be sent to particular NMS"

DEFVAL { Disable SNMPv2trap }

::= { docsDevNmAccessEntry 8 }

Any cable device (CMTS/CM) SHOULD implement the ifTestTable [RFC-2233] for any diagnostic test procedures that can be remotely initiated.

4.4.2 Event Notification

A cable device (CMTS/CM) MUST generate asynchronous events that indicate malfunction situations and notify about important non-fault events. Events could be stored in CMTS/CM device internal event LOG file, in non-volatile memory, get reported to other SNMP entities (as TRAP or INFORM SNMP messages), or be sent as a SYSLOG event message to a pre-defined SYSLOG server. Events MAY also be sent to the cable device (CMTS/CM) console; as a duplicate (identical) message to the optional console destination.

Event notification implemented by a cable device (CMTS/CM) MUST be fully configurable, by priority class; including the ability to disable SNMP Trap, SYSLOG transmission, and local logging. CMTS/CM MUST implement docsDevEvControlTable to control reporting of event classes. The object docsDevEvReporting MUST be implemented as RW for CMTS/CM.

A cable device (CMTS/CM) MUST support the following event notification mechanisms (regardless of what SNMP mode the cable device is in):

- local event logging
- SNMP TRAP/INFORM (trap-versions/targets/limiting/throttling)
- SYSLOG (targets/limiting/throttling)
- (Refer to the following sections for event notification implementation details)

When a CM is in SNMP v1/v2c NmAccess mode, the CM MUST support event notification functions including local event logging, SYSLOG (targets/limiting/throttling) and SNMP TRAP (trap-versions/targets/limiting/throttling) as specified in RFC-2669 and OSSI 1.1. When CM is in SNMP coexistence mode, CM MUST support event notification functions including local event logging, SYSLOG (targets/limiting/throttling) and SNMP TRAP (limiting/throttling) as specified in RFC-2669 and OSSI 1.1, and SNMP notification functions as specified in RFC-2573.

If the CMTS supports, and is in SNMP v1/v2c NmAccess mode, the CMTS MUST support event notification functions including local event logging, SYSLOG (targets/limiting/throttling) and SNMP TRAP (limiting/throttling) as specified in RFC-2669 and OSSI 1.1; however, SNMP TRAP (trap-versions/targets) MAY be implemented as specified in RFC-2669 and OSSI 1.1, or vendor proprietary MIB. When CMTS is in SNMP Coexistence mode, CMTS MUST support event notification functions including local event logging, SYSLOG (targets/limiting/throttling) and SNMP TRAP (limiting/throttling) as specified in RFC-2669 and OSSI 1.1, and SNMP notification functions as specified in RFC-2573.

4.4.2.1 Local Event Logging

Event logging provides a mechanism to store critical and error events in non-volatile memory. The event log storage, and access mechanism, MUST be implemented in cable device (CMTS/CM) as described below. A DOCSIS 1.1 compliant cable device (CMTS/CM) MUST implement docsDevEventTable with additional requirement as specify by the OSSI 1.1

The cable device (CMTS/CM) event log MUST be organized as a cyclic buffer with a minimum of ten entries, and MUST persist across re-boot. In the event of a full CMTS/CM event log, new log events MUST overwrite the oldest event log entry (oldest event is the event that has oldest dosDevEvLastTime). The event log table MUST be accessible through docsDevEventTable[RFC-2669] by cable device (CM/CMTS). Only events of the priority which has Local Log enable MUST be stored in the local log and appear in the docsDevEventTable. Each entry in the dosDevEventTable contains an event-ID (identical to the Eventid requirement specify in section 4.4.2.2.2), event time stamp when the event occurred first time and last time, number of appearances and event description in human-readable English format. Total length of the each event description entry MUST not be longer then 255 characters (max. defined for SNMPadminString). A network management application could periodically poll the event log table and retrieve event log entries

Identical events mean that the event-IDs are identical. For those events that may have different display text strings (i.e. display different IP address), only the string from the last event occurred MUST be stored in docsDevEvText.

4.4.2.2 Format of Events

4.4.2.2.1 SNMP TRAP/Inform

All SNMP TRAPs associated with the events described in this document are defined in the corresponding parts of standard DOCSIS MIB-s (CABLE-DEVICE-MIB, BPI-PLUS-MIB, and DOCS-IF-MIB).

A cable device (CMTS/CM) MUST send the following generic SNMP TRAPs, as defined in standard MIB [RFC1907] and [RFC2233]:

- coldStart (warmStart is optional) [RFC-2233]
- linkUp [RFC-2233]
- linkDown [RFC-2233]
- SNMP authentication-Failure [RFC-1907]

Vendor-specific events reportable via SNMP TRAP MUST be described in the vendor private MIBs. The last digit of the MIB OID MUST be the EventId of the event associated with the trap. The event-ID digit is a 32-bit unsigned integer. EventId's from 0 to 2^31-1 are reserved by DOCSIS. The event-ID (number) is converted from the error codes defined in Appendix J of [MCNS5] (as described in the "SYSLOG Message Format" section).

The EventId from 2^31 to 2^32-1 will be used as vendor-specific event-ID with the following requirements:

Bit 31 top bit set on = Vendor Specific Event

Put the bottom 15 bits of the vendor's SNMP enterprise number in bits 30 to 16

Bits 15 to 0 can be used by vendor for events (65,535 possible events per vendor)

CMTS/CM in SNMP v1/v2c NmAccess mode MUST support SNMPv1 and SNMPv2c Traps. CMTS/CM in SNMP Coexistence mode MUST support SNMPv1, SNMPv2c, and SNMPv3 Traps. Cable device (CMTS/CM) MUST support INFORM.

The standard DOCSIS TRAPS and vendor-specific TRAPS in SNMPv1 format MUST have the 'enterprise' field set to the MIB OID of the ROOT part, of the notification section, of the corresponding MIB as defined in (CABLE-DEVICE-MIB, BPI-MIB or the private vendor MIB). The 'trap-specific' field, of SNMPv1 TRAP PDU, MUST have the EventId code of the corresponding event. This way the 'enterprise' field together with an OID of 0 and the 'specific-trap' will form the SNMPv2 trap OID for this trap. For example, for the Cable Device MIB, the enterprise field is {docsDev 2}. For a specific trap with eventID code of 1140850944, the SNMPv2 trap OID would be {docsDev 2.0.1140850944}. This is in accordance with RFC-2576, section 3.1, paragraph 2.

The TRAPS in SNMPv2c format must contain the sysUpTime.0 object and the snmpTrapOID.0 object set to the corresponding TRAP MIB OID.

Traps in both SNMPv1 and SNMPv2c format must include the object docsDevEvText, set to the textual description of the event, which MUST be identical to the text that is included in SYSLOG message and stored in the local event log file. Also, Traps MUST include all objects that are defined in the particular SNMP trap definition.

4.4.2.2.2 SYSLOG Message Format

CM's Syslog message MUST be sent in the following format:
<*level*>CABLEMODEM[*vendor*]: <*eventId*> *text*
Where:
Level - ASCII presentation of the event priority, enclosed in angle brackets, which is constructed as OR of the default Facility (128) and event priority (0-7). The resulted level has the range between 128 and 135.

CMTS's Syslog message MUST be sent in the following format:
<*level*>CMTS[*vendor*]: <*eventId*> *text*
Where:
Level - ASCII presentation of the event priority, enclosed in angle brackets, which is constructed as OR of the default Facility (128) and event priority (0-7). The resulted level has the range between 128 and 135

vendor - Vendor name for the vendor-specific SYSLOG messages or DOCSIS for the standard DOCSIS messages.

***EventId* - ASCII presentation of the INTEGER number in HEX format, enclosed in angle brackets, which uniquely identifies the type of event. This number MUST be the same number that is stored in docsDevEvId object in docsDevEventTable and also is associated with SNMP TRAP in the "SNMP TRAP/Inform" section. For the standard DOCSIS events specified in [SP-RFIv1.1] this number is converted from the error codes [SP-RFIv1.1 Appendix J] using the following rules:**

- The first byte (MSB) of the EventID is ASCII code for the letter in the Error code from Appendix J.
- Next 2 bytes of the EventID represent 2 or 3 digits before the dot in the Error code.
- The last byte is the number after the dot in the Error code.

  For example, event D04.2 has the number (in HEX) 0x44000402

  Event I114.1 has the number 0x49007201

text - for the standard DOCSIS messages this string MUST have the textual description as defined in [SP-RFIv1.1 Appendix J]. For the vendor-specific messages it could be any ASCII single-line string that describes event.

The example of the syslog event for the event D04.2
"Time of the day received in invalid format":

<132>CABLEMODEM[DOCSIS]: <44000402> Time of Day Response but invalid data/format.
The number 44000402 in the given example is the number assigned by DOCSIS to this particular event.

4.4.2.3 Standard DOCSIS Events for CM

The DOCSIS cable device MIB [RFC2669] document defines 8 different priority levels and the corresponding reporting mechanism for each level. The standard DOCSIS events specified in this document utilizes the subset of these priority levels.

Emergency event (priority 1)

Reserved for vendor-specific 'fatal' hardware or software errors that prevents normal system operation and causes reporting system to reboot.

> **Every vendor may define there own set of emergency events. The examples of such events could be 'no memory buffers available', 'memory test failure' etc. (Such basic cross-vendor type events should be included in the DOCSIS 1.1 "Events for Notification" Appendix F so that vendors do not define many overlapping EventId's in vendor-private scope)**

Alert event (priority 2)

A serious failure, which causes reporting system to reboot but it is not caused by h/w or s/w malfunctioning. After recovering from the critical event system MUST send the cold/warm start notification. Alert event could not be reported as a Trap or SYSLOG message and MUST be stored in the internal log file. The code of this event MUST be saved in non-volatile memory and reported later through docsIfCmStatusCode SNMP variable [RFC2670].

Critical event (priority 3)

A serious failure that requires attention and prevents the device from transmitting data but could be recovered without rebooting the system. After recovering from the error event Cable Modem Device MUST send the Link Up notification. Critical events could not be reported as a Trap or SYSLOG message and MUST be stored in the internal log file. The code of this event MUST be reported later through docsIfCmStatusCode SNMP variable [RFC2670]. The examples of such events could be configuration file problems detected by the modem or inability to get IP address from DHCP.

Error event (priority 4)

A failure occurred that could interrupt the normal data flow but does not cause modem to re-register. Error events could be reported in real time by using TRAP or SYSLOG mechanism.

Warning event (priority 5)

A failure occurred that could interrupt the normal data flow but does not cause modem to re-register. 'Warning' level is assigned to events both modem and CMTS have information about. So to prevent sending same event both from the CM and CMTS, trap and Syslog reporting mechanism is disabled by default for this level.

Notice event (priority 6)

The event of importance which is not a failure and could be reported in real time by using TRAP or SYSLOG mechanism. The examples of the NOTICE events are 'Cold Start', 'Warm Start', 'Link Up' and 'SW upgrade successful'.

Informational event (priority 7)
The not-important event, which is not failure, but could be helpful for tracing the normal modem operation. By default these events are not saved into the local event log and no Syslog/trap is sent.

Debug event (priority 8)
Reserved for vendor-specific non-critical events

The priority associated with the event is hard-coded and can't be changed. The reporting mechanism for each priority could be changed from the default reporting mechanism (Table 1) by using docsDevEvReporting object in cable device MIB [RFC2669].

Event Priority	Local Log	Trap	Syslog	Note
1 Emergency	Yes	No	No	Vendor-spec.
2 Alert	Yes	No	No	DOCSIS
3 Critical	Yes	No	No	DOCSIS
4 Error	Yes	Yes	Yes	DOCSIS
5 Warning	Yes	No	No	DOCSIS
6 notice	Yes	Yes	Yes	DOCSIS
7 informational	No	No	No	DOCSIS/vend.
8 debug	No	No	No	Vendor-spec.

Table 1 Default event priorities for the Cable Modem Device

DOCSIS 1.1 compliant CM MUST generate event notification based on events specified in Appendix F.

4.4.2.4 Standard DOCSIS Events for CMTS

CMTS uses the same levels of the event priorities as a CM; however, the severity definition of the events is different. Events with the severity level of Warning and less specify problems that could affect individual user (for example, individual CM registration problem).

Severity level of 'Error' indicates problems with a group of CMs (for example CMs that share same upstream channel).

Severity level of 'Critical' indicates problem that affects whole cable system operation, but is not a faulty condition of CMTS device. In all these cases CMTS MUST be able to send SYSLOG event and (or) SNMP TRAP to the NMS.

Severity level of 'Emergency' is vendor-specific and indicates problems with the CMTS hardware or software, which prevents CMTS operation.

Event Priority	Local Log	Trap	Syslog	Note

1 Emergency	Yes	No	No	Vendor-spec.
2 Alert	Yes	No	No	Vendor-spec.
3 Critical	Yes	Yes	Yes	DOCSIS
4 Error	Yes	Yes	Yes	DOCSIS
5 Warning	No	Yes	Yes	DOCSIS
6 notice	No	Yes	Yes	DOCSIS
7 informational	No	No	No	DOCSIS/vend.
8 debug	No	No	No	Vendor-spec.

Table 2 Default Event priorities for the CMTS Device

DOCSIS 1.1 compliant CMTS MUST generate event notification based on events specified in Appendix F.

4.4.3 Trap and Syslog Throttling, Trap and Syslog Limiting

DOCSIS 1.1 compliant cable device (CMTS/CM) MUST support SNMP TRAP/INFORM and SYSLOG throttling and limiting as described in RFC-2669, regardless of SNMP mode.

4.4.4 Non-SNMP Fault Management Protocols

The OSS can use a variety of tools and techniques to examine faults at multiple layers. For the IP layer, useful non-SNMP based tools include ping (ICMP Echo and Echo Reply), traceroute (UDP and various ICMP Destination Unreachable flavors). Pings to a CM from its CMCI side MUST be supported to enable local connectivity testing from a customer's PC to the modem. The CM and CMTS MUST support IP end-station generation of ICMP error messages and processing of all ICMP messages.

4.5 Performance Management

At the CATV MAC and PHY layers, performance management focuses on the monitoring of the effectiveness of cable plant segmentation and rates of upstream traffic and collisions. Instrumentation is provided in the form of the standard interface statistics [RFC-2233], as well as the docsifCmtsServiceTable and docsifCmServiceTable entries. It is not anticipated that the CMTS upstream bandwidth allocation function will require active network management intervention and tuning.

At the LLC layer, the performance management focus is on bridge traffic management. The CM and CMTS (if the CMTS implements transparent bridging) MUST implement the Bridge MIB RFC-1493, including the dot1dBase and dot1dTp groups. The CM and CMTS MUST implement a managed object that controls whether the 802.1d spanning tree protocol (STP) is run and topology update messages are generated; STP is unnecessary in hierarchical, loop-free topologies. If the STP is enabled for the CM/CMTS, then the CM/CMTS MUST implement the dot1dStp group. These MIB groups' objects allow the NMS to detect when bridge forwarding tables are full, and enable the NMS to modify aging timers.

A final performance concern is the ability to diagnose unidirectional loss. Both the CM and CMTS MUST implement the MIB-2 [RFC-2233] Interfaces group. When there exists more than one upstream or downstream channel, the CM/CMTS MUST implement an instance of IfEntry for each channel. The ifStack group [RFC-2233] MUST be used to define the relationships among the CATV MAC interfaces and their channels.

4.5.1 Additional MIB Implementation Requirements

To support performance monitoring and data collection for capacity, fault, and performance management, CM and CMTS MUST support MIB objects with:

- Accurate in measurement
- Counter properly working (i.e. counter roll over at maximum)
- Correct counter capacity
- Counter reset properly
- Update rate of 1 second

4.6 Coexistence

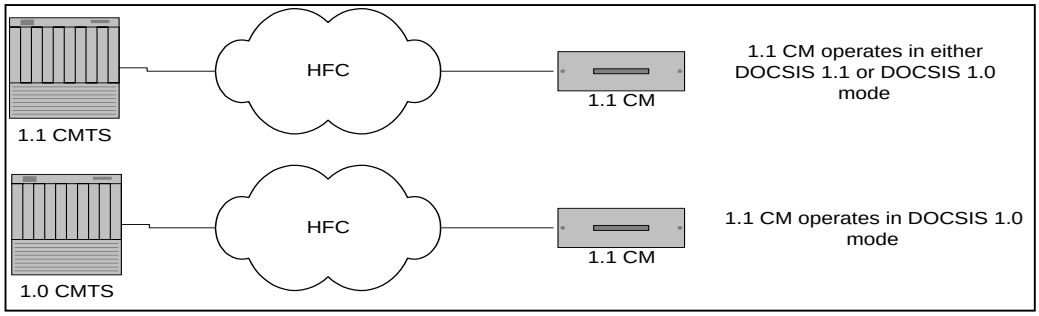


Figure 3. Coexistent (DOCSIS 1.0 mode VS DOCSIS 1.1 mode)

When DOCSIS 1.1 compliant CM is connected to 1.1 CMTS, it can operate in either DOCSIS 1.1 mode or DOCSIS 1.0 mode. When DOCSIS 1.1 compliant CM is connected to 1.0 CMTS, it operates in DOCSIS 1.0 mode. Refer to [MCNS5] and BPI+ specifications for more detail descriptions of what features are available when DOCSIS 1.1 compliant CM is operating in different modes.

4.6.1 Coexistence and MIBs

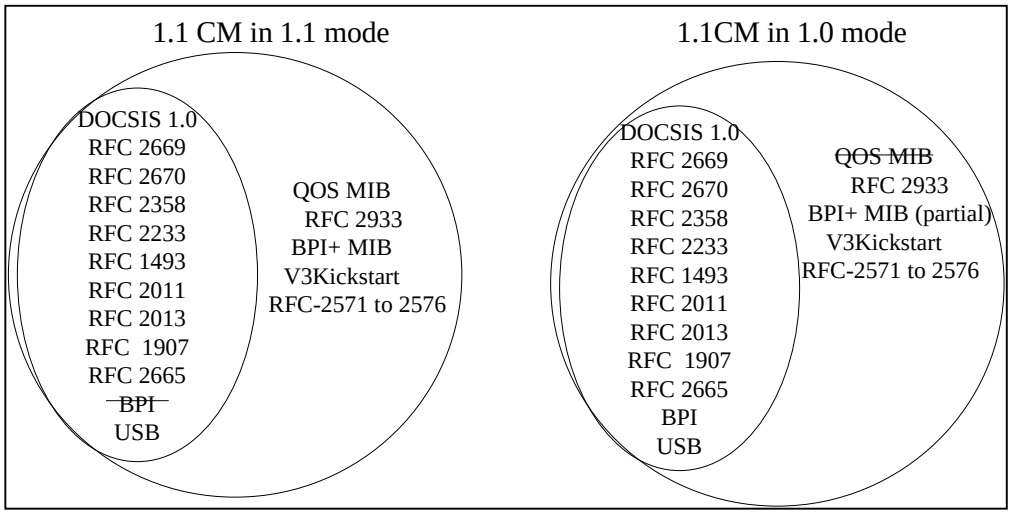


Figure 4. CM DOCSIS Mode and MIBs Requirement

4.6.1.1.1 Requirements for 1.1 CM operating in 1.1 mode

- RFC 2669
- RFC 2670
- RFC 2665
- RFC 1493
- RFC 2011
- RFC 2013
- RFC 2933
- USB MIB
- QOS MIB
- BPI+ MIB
- V3Kickstart [RFC-2786] (When CM is in SNMP V1/V2c NmAccess mode, CM MUST respond with "NoSuchName" for all the request to tables and objects in V3Kickstart.)
- RFC 2571 to 2576 (When CM is in SNMP v1/v2c NmAccess mode, CM MUST respond with "NoSuchName" for all the request to tables and objects defined in RFC 2571 to 2576.)

When DOCSIS 1.1 compliant CM operates in 1.1 mode, it MUST NOT support the following MIB(s):

- BPI MIB

BPI MIB MUST not be available for any access from SNMP manager. DOCSIS 1.1 compliant CM MUST respond with "NoSuchName" for all requests to tables and objects in BPI MIB.

4.6.1.1.2 Requirements for 1.1 CM operating in 1.0 mode

When DOCSIS 1.1 compliant CM operates in 1.0 mode, it MUST support the following MIBs:

- RFC 2669
- RFC 2670
- RFC 2665
- RFC 1493
- RFC 2011
- RFC 2013
- RFC 2933 (IF CM in 1.0 mode supports IGMP, it must implement RFC 2933)
- USB MIB
- BPI MIB
- BPI+ MIB. Part of the BPI+ MIB MUST be supported. Refer to Appendix A for specific MIB object requirements.
- V3Kickstart [RFC-2786] (When CM is in SNMP V1/V2c NmAccess mode, CM MUST respond with "NoSuchName" for all the request to tables and objects in V3Kickstart.)
- RFC 2571 to 2576 (When CM is in SNMP v1/v2c NmAccess mode, CM MUST respond with "NoSuchName" for all the request to tables and objects defined in RFC 2571 to 2576.)

When DOCSIS 1.1 compliant CM operates in 1.0 mode, it MUST NOT support the following MIB(s):

- QOS MIB

- BPI+ (part of the BPI+ MIB MUST still be supported to enable secure software download. Refer to Appendix A for specific MIB object requirements.)
- QOS MIB and BPI+ MIB, MUST not be available for any access from SNMP manager. DOCSIS 1.1 compliant CM MUST respond with “NoSuchName” for all requests to tables and objects in QOS MIB, and BPI+ MIB.

4.6.2 Coexistence and SNMP

DOCSIS 1.1 compliant CM MUST support SNMPv3 and SNMPv1/v2c functionality as specified in Section 2 regardless of what mode (DOCSIS 1.0 or DOCSIS 1.1) CM operates in.

5 OSS for BPI+

The X.509 Digital Certificates management policy and the requirements related to the management of those Digital Certificates will be specified in this section by the ECR/ECO/ECN process.

6 OSSI for CMCI

This section defines the operational mechanisms needed to support the transmission of data over cable services between a cable modem and the customer premise equipment. More specifically, this section will outline the following:

- SNMP access via CMCI
- Console Access
- CM diagnostic capabilities
- Protocol Filtering
- Required MIBs

Currently, the CMCI is categorized as internal, external, and CPE Controlled cable modem functional reference models. The external cable modems MAY have either an Ethernet 10BASE-T or Universal Serial Bus (USB) CMCI interface or both. If both interfaces are present on a CM, they MAY be active at the same time.

The internal cable modems MUST utilize the Peripheral Component Interface (PCI) bus for transparent bi-directional IP traffic forwarding. The PCI interface MUST be defined and accessible from an SNMP manager for both operational and security purposes.

The CPE Controlled Cable modems (CCCM) CMCI MAY be either a Peripheral Component Interface (PCI) or Universal Serial Bus (USB) interface. If PCI is utilized, the interface MUST be defined and accessible from an SNMP manager for both operational and security purposes.

6.1 SNMP Access via CMCI

A CM device providing CPE SNMP access, prior to completing of the CMTS registration process, MUST comply with the SNMP access requirement specified in section 2.2

6.2 Console Access

An external cable modem MUST NOT allow access to the CM functions via a console port. For this specification, a console port is defined as a communication path, either hardware or software that allows a user to issue commands to modify the configuration or operational status of the CM. Access to the external CM MUST only be allowed using DOCSIS 1.1 defined RF interfaces and operator-controlled SNMP access via the CMCI.

6.3 CM Diagnostic Capabilities

The cable modem MAY have read-only diagnostic interfaces for debugging and troubleshooting purposes. The read-only diagnostic interface MUST NOT display any network addressing or operational information.

6.4 Protocol Filtering

The CM MUST be capable of filtering all broadcast traffic from the host CPE, with the exception of DHCP and ARP packets. This filtering function must adhere to section 4.3 (Protocol Filters) of this document. All ICMP type packets MUST be forwarded from the CMCI interface to the RF upstream interface. The CMCI MUST also adhere to the data forwarding rules defined in [MCNS 5].

6.5 Management Information Base (MIB) Requirements

All Cable Modems MUST implement the MIBs detailed in section 3 (Management Information Bases) of this specification, with the following exceptions:

- An external CM with only USB interface(s), MUST NOT implement RFC-2665: Ethernet Interface MIB.
- An external CM with only USB interface(s), MUST implement the IETF Proposed Standard RFC version of USB MIB.
- An internal CM MAY implement RFC-2665: Ethernet Interface MIB.

Appendix A. Detailed MIB Requirements

NOTE:

ACC-FN- Accessible for Notify
D - Deprecated
M - Mandatory
N-Acc - Not accessible
NA - Not Applicable
N-Sup - MUST not support
O - Optional
Ob - Obsolete
RC - Read-Create
RO - Read-Only
RW - Read-Write
RC/RO – Read-Create or Read-Only
RW/RO – Read-Write or Read-Only

General rules:

D - Deprecated – It is 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 instantiated such object and MUST respond with the appropriate error/exception condition. (e.g., no such object for SNMPv2c)

M - Mandatory – The object MUST be implemented correctly according to the MIB definition.

N-Acc - Not Accessible – The object is not accessible and is usually an index in a table.

NA - Not Applicable – Not applicable to the device.

N-Sup - MUST Not Support – Device MUST NOT support the object. That is, an agent MUST NOT instantiated such object and MUST respond with the appropriate error/exception condition. (e.g., no such object for SNMPv2c)

O - 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 instantiated such object and MUST respond with the appropriate error/exception condition. (e.g., no such object for

SNMPv2c)

Ob - Obsolete – It is optional. Though in SNMP convention, obsolete objects should not be implemented, DOCSIS 1.1 OSSI lets vendors choose whether or not to support the obsolete object. That is, a vendor can choose to implement or not implement the obsolete 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, SNMP agent MUST NOT instantiated such object and MUST respond with the appropriate error/exception condition. (e.g., no such object for SNMPv2c)

RC – Read-Create – The access of the object MUST be implemented as Read-Create.

RO – Read-Only – The access of the object MUST be implemented as Read-Only.

RW – Read-Write – The access of the object MUST be implemented as Read-Write.

RC/RO – Read-Create or Read-Only – The access of the object MUST be implemented as either Read-Create or Read-Only as described in the MIB definition.

RW/RO – Read-Write or Read-Only – The access of the object MUST be implemented as either Read-Write or Read-Only as described in the MIB definition.

DOCS-IF-MIB (RFC 2670)				
docsIfDownstreamChannelTable				
Object	**CM**	**Access**	**CMTS**	**Access**
docsIfDownChannelId	M	RO	M	RO
docsIfDownChannelFrequency	M	RO	M	RW/RO
docsIfDownChannelWidth	M	RO	M	RW/RO
docsIfDownChannelModulation	M	RO	M	RW
docsIfDownChannelInterleave	M	RO	M	RW
docsIfDownChannelPower	M	RO	M	RW/RO

docsIfUpstreamChannelTable						
Object			**CM**	**Access**	**CMTS**	**Access**
docsIfUpChannelId			M	RO	M	RO
docsIfUpChannelFrequency			M	RO	M	RW
docsIfUpChannelWidth			M	RO	M	RW
docsIfUpChannelModulationProfile			M	RO	M	RW
docsIfUpChannelSlotSize			M	RO	M	RW/RO
docsIfUpChannelTxTimingOffset			M	RO	M	RO
docsIfUpChannelRangingBackoffStart			M	RO	M	RW
docsIfUpChannelRangingBackoffEnd			M	RO	M	RW
docsIfUpChannelTxBackoffStart			M	RO	M	RW
docsIfUpChannelTxBackoffEnd			M	RO	M	RW
docsIfQosProfileTable						
Object	**1.1CM in 1.0 mode**	**Access**	**1.1 CM in 1.1 mode**	**Access**	**CMTS**	**Access**
docsIfQosProfileIndex	M	N-Acc	O	N-Acc	O	N-Acc
docsIfQosProfPriority	M	RO	O	RO	O	RC/RO
docsIfQosProfMaxUpBandwidth	M	RO	O	RO	O	RC/RO
docsIfQosProfGuarUpBandwidth	M	RO	O	RO	O	RC/RO
docsIfQosProfMaxDownBandwidth	M	RO	O	RO	O	RC/RO
docsIfQosProfMaxTxBurst	M	RO	O	RO	O	RC/RO
docsIfQosProfBaselinePrivacy	M	RO	O	RO	O	RC/RO
docsIfQosProfStatus	M	RO	O	RO	O	RC/RO
docsIfSignalQualityTable						
Object			**CM**	**Access**	**CMTS**	**Access**
docsIfSigQIncludesContention			M	RO	M	RO
docsIfSigQUnerroreds			M	RO	M	RO
docsIfSigQCorrecteds			M	RO	M	RO
docsIfSigQUncorrectables			M	RO	M	RO
docsIfSigQSignalNoise			M	RO	M	RO
docsIfSigQMicroreflections			M	RO	M	RO
docsIfSigQEqualizationData			M	RO	M	RO
docsIfCmMacTable						
Object			**CM**	**Access**	**CMTS**	**Access**
docsIfCmCmtsAddress			M	RO	NA	NA
docsIfCmCapabilities			M	RO	NA	NA
docsIfCmRangingRespTimeout			Ob	N-Sup	NA	NA
docsIfCmRangingTimeout			M	RW	NA	NA
docsIfCmStatusTable						
Object			**CM**	**Access**	**CMTS**	**Access**
docsIfCmStatusValue			M	RO	NA	NA
docsIfCmStatusCode			M	RO	NA	NA

docsIfCmStatusTxPower	M	RO	NA	NA
docsIfCmStatusResets	M	RO	NA	NA
docsIfCmStatusLostSyncs	M	RO	NA	NA
docsIfCmStatusInvalidMaps	M	RO	NA	NA
docsIfCmStatusInvalidUcds	M	RO	NA	NA
docsIfCmStatusInvalidRangingResponses	M	RO	NA	NA
docsIfCmStatusInvalidRegistrationResponses	M	RO	NA	NA
docsIfCmStatusT1Timeouts	M	RO	NA	NA
docsIfCmStatusT2Timeouts	M	RO	NA	NA
docsIfCmStatusT3Timeouts	M	RO	NA	NA
docsIfCmStatusT4Timeouts	M	RO	NA	NA
docsIfCmStatusRangingAborteds	M	RO	NA	NA
docsIfCmServiceTable				
Object	**CM**	**Access**	**CMTS**	**Access**
docsIfCmServiceId	M	N-Acc	NA	NA
docsIfCmServiceQosProfile	M	RO	NA	NA
docsIfCmServiceTxSlotsImmed	M	RO	NA	NA
docsIfCmServiceTxSlotsDed	M	RO	NA	NA
docsIfCmServiceTxRetries	M	RO	NA	NA
docsIfCmServiceTxExceededs	M	RO	NA	NA
docsIfCmServiceRqRetries	M	RO	NA	NA
docsIfCmServiceRqExceededs	M	RO	NA	NA
docsIfCmtsMacTable				
Object	**CM**	**Access**	**CMTS**	**Access**
docsIfCmtsCapabilities	NA	NA	M	RO
docsIfCmtsSyncInterval	NA	NA	M	RW/RO
docsIfCmtsUcdInterval	NA	NA	M	RW/RO
docsIfCmtsMaxServiceIds	NA	NA	M	RO
docsIfCmtsInsertionInterval	NA	NA	Ob	N-Sup
docsIfCmtsInvitedRangingAttempts	NA	NA	M	RW/RO
docsIfCmtsInsertInterval	NA	NA	M	RW/RO
docsIfCmtsStatusTable				
Object	**CM**	**Access**	**CMTS**	**Access**
docsIfCmtsStatusInvalidRangeReqs	NA	NA	M	RO
docsIfCmtsStatusRangingAborteds	NA	NA	M	RO
docsIfCmtsStatusInvalidRegReqs	NA	NA	M	RO
docsIfCmtsStatusFailedRegReqs	NA	NA	M	RO
docsIfCmtsStatusInvalidDataReqs	NA	NA	M	RO
docsIfCmtsStatusT5Timeouts	NA	NA	M	RO

docsIfCmtsCmStatusTable				
Object	**CM**	**Access**	**CMTS**	**Access**
docsIfCmtsCmStatusIndex	NA	NA	M	N-Acc
docsIfCmtsCmStatusMacAddress	NA	NA	M	RO
docsIfCmtsCmStatusIpAddress	NA	NA	M	RO
docsIfCmtsCmStatusDownChannelIfIndex	NA	NA	M	RO
docsIfCmtsCmStatusUpChannelIfIndex	NA	NA	M	RO
docsIfCmtsCmStatusRxPower	NA	NA	M	RO
docsIfCmtsCmStatusTimingOffset	NA	NA	M	RO
docsIfCmtsCmStatusEqualizationData	NA	NA	M	RO
docsIfCmtsCmStatusValue	NA	NA	M	RO
docsIfCmtsCmStatusUnerroreds	NA	NA	M	RO
docsIfCmtsCmStatusCorrecteds	NA	NA	M	RO
docsIfCmtsCmStatusUncorrectables	NA	NA	M	RO
docsIfCmtsCmStatusSignalNoise	NA	NA	M	RO
docsIfCmtsCmStatusMicroreflections	NA	NA	M	RO
docsIfCmtsServiceTable				
Object	**CM**	**Access**	**CMTS**	**Access**
docsIfCmtsServiceId	NA	NA	M	N-Acc
docsIfCmtsServiceCmStatusIndex	NA	NA	M	RO
DocsIfCmtsServiceAdminStatus	NA	NA	M	RW/RO
docsIfCmtsServiceQosProfile	NA	NA	M	RO
docsIfCmtsServiceCreateTime	NA	NA	M	RO
docsIfCmtsServiceInOctets	NA	NA	M	RO
docsIfCmtsServiceInPackets	NA	NA	M	RO
docsIfCmtsModulationTable				
Object	**CM**	**Access**	**CMTS**	**Access**
docsIfCmtsModIndex	NA	NA	M	N-Acc
docsIfCmtsModIntervalUsageCode	NA	NA	M	N-Acc
docsIfCmtsModControl	NA	NA	M	RC
docsIfCmtsModType	NA	NA	M	RC
docsIfCmtsModPreambleLen	NA	NA	M	RC
docsIfCmtsModDifferentialEncoding	NA	NA	M	RC
docsIfCmtsModFECErrorCorrection	NA	NA	M	RC
docsIfCmtsModFECCodewordLength	NA	NA	M	RC
docsIfCmtsModScramblerSeed	NA	NA	M	RC
docsIfCmtsModMaxBurstSize	NA	NA	M	RC
docsIfCmtsModGuardTimeSize	NA	NA	M	RO
docsIfCmtsModLastCodewordShortened	NA	NA	M	RC
docsIfCmtsModScrambler	NA	NA	M	RC
Object				
docsIfCmtsQosProfilePermissions	NA	NA	M	RW /RO

DocsIfCmtsMacToCmTable				
Object	**CM**	**Access**	**CMTS**	**Access**
docsIfCmtsCmMac	NA	NA	M	N-Acc
docsIfCmtsCmPtr	NA	NA	M	RO
IF-MIB (RFC 2233)				
Object	**CM**	**Access**	**CMTS**	**Access**
ifNumber	M	RO	M	RO
IfTableLastChange	M	RO	M	RO
ifTable				
Object	**CM**	**Access**	**CMTS**	**Access**
ifIndex	M	RO	M	RO
ifDescr	M	RO	M	RO
ifType	M	RO	M	RO
ifMtu	M	RO	M	RO
ifSpeed	M	RO	M	RO
ifPhysAddress	M	RO	M	RO
ifAdminStatus	M	RW	M	RW
ifOperStatus	M	RO	M	RO
ifLastChange	M	RO	M	RO
ifInOctets	M	RO	M	RO
ifInUcastPkts	M	RO	M	RO
ifInNUcastPkts	D	RO	D	RO
ifInDiscards	M	RO	M	RO
ifInErrors	M	RO	M	RO
ifInUnknownProtos	M	RO	M	RO
ifOutOctets	M	RO	M	RO
ifOutUcastPkts	M	RO	M	RO
ifOutNUcastPkts	D	RO	D	RO
ifOutDiscards	M	RO	M	RO
ifOutErrors	M	RO	M	RO
ifOutQLen	D	RO	D	RO
ifSpecific	D	RO	D	RO
ifXTable				
Objects	**CM**	**Access**	**CMTS**	**Access**
ifName	M	RO	M	RO
ifInMulticastPkts	M	RO	M	RO
ifInBroadcastPkts	M	RO	M	RO
ifOutMulticastPkts	M	RO	M	RO
ifOutBroadcastPkts	M	RO	M	RO

ifHCInOctets	O	RO	O	RO
ifHCInUcastPkts	O	RO	O	RO
ifHCInMulticastPkts	O	RO	O	RO
ifHCInBroadcastPkts	O	RO	O	RO
ifHCOutOctets	O	RO	O	RO
ifHCOutUcastPkts	O	RO	O	RO
ifHCOutMulticastPkts	O	RO	O	RO
ifHCOutBroadcastPkts	O	RO	O	RO
ifLinkUpDownTrapEnable	M	RW	M	RW
ifHighSpeed	M	RO	M	RO
ifPromiscuousMode	M	RW/RO	M	RW/RO
ifConnectorPresent	M	RO	M	RO
ifAlias	M	RW/RO	M	RW/RO
ifCounterDiscontinuityTime	M	RO	M	RO
ifStackTable				
Objects	**CM**	**Access**	**CMTS**	**Access**
ifStackHigherLayer	M	N-Acc	M	N-Acc
ifStackLowerLayer	M	N-Acc	M	N-Acc
ifStackStatus	M	RC/RO	M	RC/RO
Object	**CM**	**Access**	**CMTS**	**Access**
ifStackLastChange	O	N-Acc	O	N-Acc
ifRcvAddressTable				
Object	**CM**	**Access**	**CMTS**	**Access**
ifRcvAddressAddress	O	N-Acc	O	N-Acc
ifRcvAddressStatus	O	RC	O	RC
IfRcvAddressType	O	RC	O	RC
6.5.1.1 Notification				
linkUp	M		M	
linkDown	M		M	
ifTestTable				
Objects	**CM**	**Access**	**CMTS**	**Access**
ifTestId	O	RW	O	RW
ifTestStatus	O	RW	O	RW
ifTestType	O	RW	O	RW
ifTestResult	O	RO	O	RO
ifTestCode	O	RO	O	RO
ifTestOwner	O	RW	O	RW

BRIDGE-MIB (RFC 1493)				
NOTE: Implementation of BRIDGE MIB is required ONLY if device is a bridging device				
dot1dBase group				
Objects	**CM**	**Access**	**CMTS**	**Access**
dot1dBaseBridgeAddress	M	RO	M	RO
dot1dBaseNumPorts	M	RO	M	RO
dot1dBaseType	M	RO	M	RO
dot1dBasePortTable				
Objects	**CM**	**Access**	**CMTS**	**Access**
dot1dBasePort	M	RO	M	RO
dot1dBasePortIfIndex	M	RO	M	RO
dot1dBasePortCircuit	M	RO	M	RO
dot1dBasePortDelayExceededDiscards	M	RO	M	RO
dot1dBasePortMtuExceededDiscards	M	RO	M	RO
dot1dStp group				
NOTE: This group is required ONLY if STP is implemented				
Objects	**CM**	**Access**	**CMTS**	**Access**
dot1dStpProtocolSpecification	M	RO	M	RO
dot1dStpPriority	M	RW	M	RW
dot1dStpTimeSinceTopologyChange	M	RO	M	RO
dot1dStpTopChanges	M	RO	M	RO
dot1dStpDesignatedRoot	M	RO	M	RO
dot1dStpRootCost	M	RO	M	RO
dot1dStpRootPort	M	RO	M	RO
dot1dStpMaxAge	M	RO	M	RO
dot1dStpHelloTime	M	RO	M	RO
dot1dStpHoldTime	M	RO	M	RO
dot1dStpForwardDelay	M	RO	M	RO
dot1dStpBridgeMaxAge	M	RW	M	RW
dot1dStpBridgeHelloTime	M	RW	M	RW
dot1dStpBridgeForwardDelay	M	RW	M	RW
dot1dStpPortTable				
NOTE:This table is required ONLY if STP is implemented				
Objects	**CM**	**Access**	**CMTS**	**Access**
dot1dStpPort	M	RO	M	RO
dot1dStpPortPriority	M	RW	M	RW
dot1dStpPortState	M	RO	M	RO
dot1dStpPortEnable	M	RW	M	RW
dot1dStpPortPathCost	M	RW	M	RW
dot1dStpPortDesignatedRoot	M	RO	M	RO

dot1dStpPortDesignatedCost	M	RO	M	RO
dot1dStpPortDesignatedBridge	M	RO	M	RO
dot1dStpPortDesignatedPort	M	RO	M	RO
dot1dStpPortForwardTransitions	M	RO	M	RO
dot1dTp group				
Note: This group is required ONLY if transparent bridging is implemented.				
Objects	**CM**	**Access**	**CMTS**	**Access**
dot1dTpLearnedEntryDiscards	M	RO	M	RO
dot1dTpAgingTime	M	RW	M	RW
dot1dTpFdbTable				
Objects	**CM**	**Access**	**CMTS**	**Access**
dot1dTpFdbAddress	M	RO	M	RO
dot1dTpFdbPort	M	RO	M	RO
dot1dTpFdbStatus	M	RO	M	RO
dot1dTpPortTable				
Objects	**CM**	**Access**	**CMTS**	**Access**
dot1dTpPort	M	RO	M	RO
dot1dTpPortMaxInfo	M	RO	M	RO
dot1dTpPortInFrames	M	RO	M	RO
dot1dTpPortOutFrames	M	RO	M	RO
dot1dTpPortInDiscards	M	RO	M	RO
dot1dStaticTable				
Note: Implementation of dot1dStaticTable is OPTIONAL				
Objects	**CM**	**Access**	**CMTS**	**Access**
dot1dStaticAddress	O	RW	O	RW
dot1dStaticReceivePort	O	RW	O	RW
dot1dStaticAllowedToGoTo	O	RW	O	RW
dot1dStaticStatus	O	RW	O	RW
DOCS-CABLE-DEVICE-MIB (RFC 2669)				
docsDevBaseGroup				
Objects	**CM**	**Access**	**CMTS**	**Access**
docsDevRole	M	RO	O	RO
docsDevDateTime	M	RW	O	RW
docsDevResetNow	M	RW	O	RW
docsDevSerialNumber	M	RO	O	RO
docsDevSTPControl	M	RW/RO	O	RW/RO
docsDevNmAccessGroup				

NOTE: docsDevNmAccessGroup is NOT accessible when the device is in SNMP Coexistence mode.				
docsDevNmAcessTable				
Objects	**CM**	**Access**	**CMTS**	**Access**
docsDevNmAccessIndex	M	N-Acc	O	N-Acc
docsDevNmAccessIp	M	RC	O	RC
docsDevNmAccessIpMask	M	RC	O	RC
docsDevNmAccessCommunity	M	RC	O	RC
docsDevNmAccessControl	M	RC	O	RC
docsDevNmAccessInterfaces	M	RC	O	RC
docsDevNmAccessStatus	M	RC	O	RC
DocsDevNmAccessTrapVersion (Note: This object is currently not in RFC 2669)	M	RC	O	RC
docsDevSoftwareGroup				
Objects	**CM**	**Access**	**CMTS**	**Access**
docsDevSwServer	M	RW	O	RW
docsDevSwFilename	M	RW	O	RW
docsDevSwAdminStatus	M	RW	O	RW
docsDevSwOperStatus	M	RO	O	RO
docsDevSwCurrentVers	M	RO	O	RO
docsDevServerGroup				
Objects	**CM**	**Access**	**CMTS**	**Access**
docsDevServerBootState	M	RO	N-Sup	
docsDevServerDhcp	M	RO	N-Sup	
docsDevServerTime	M	RO	N-Sup	
docsDevServerTftp	M	RO	N-Sup	
docsDevServerConfigFile	M	RO	N-Sup	
docsDevEventGroup				
Objects	**CM**	**Access**	**CMTS**	**Access**
docsDevEvControl	M	RW	M	RW
docsDevEvSyslog	M	RW	M	RW
docsDevEvThrottleAdminStatus	M	RW	M	RW
docsDevEvThrottleInhibited	M	RO	M	RO
docsDevEvThrottleThreshold	M	RW	M	RW
docsDevEvThrottleInterval	M	RW	M	RW
docsDevEvControlTable				
Objects	**CM**	**Access**	**CMTS**	**Access**
docsDevEvPriority	M	N-Acc	M	N-Acc
docsDevEvReporting (Mandatory RW by DOCSIS 1.1; exception to RFC-2669)	M	RW	M	RW

docsDevEventTable				
Objects	**CM**	**Access**	**CMTS**	**Access**
docsDevEvIndex	M	N-Acc	M	N-Acc
docsDevEvFirstTime	M	RO	M	RO
docsDevEvLastTime	M	RO	M	RO
docsDevEvCounts	M	RO	M	RO
docsDevEvLevel	M	RO	M	RO
docsDevEvId	M	RO	M	RO
docsDevEvText	M	RO	M	RO
docsDevFilterGroup				
Objects	**CM**	**Access**	**CMTS**	**Access**
docsDevFilterLLCUnmatchedAction	M	RW	O	RW
docsDevFilterLLCTable				
Objects	**CM**	**Access**	**CMTS**	**Access**
docsDevFilterLLCIndex	M	N-Acc	O	N-Acc
docsDevFilterLLCStatus	M	RC	O	RC
docsDevFilterLLCIfIndex	M	RC	O	RC
docsDevFilterLLCProtocolType	M	RC	O	RC
docsDevFilterLLCProtocol	M	RC	O	RC
docsDevFilterLLCMatches	M	RO	O	RO
Objects	**CM**	**Access**	**CMTS**	**Access**
docsDevFilterIpDefault	M	RW	O	RW
docsDevFilterIpTable				
Objects	**CM**	**Access**	**CMTS**	**Access**
docsDevFilterIpIndex	M	N-Acc	O	N-Acc
docsDevFilterIpStatus	M	RC	O	RC
docsDevFilterIpControl	M	RC	O	RC
docsDevFilterIpIfIndex	M	RC	O	RC
docsDevFilterIpDirection	M	RC	O	RC
docsDevFilterIpBroadcast	M	RC	O	RC
docsDevFilterIpSaddr	M	RC	O	RC
docsDevFilterIpSmask	M	RC	O	RC
docsDevFilterIpDaddr	M	RC	O	RC
docsDevFilterIpDmask	M	RC	O	RC
docsDevFilterIpProtocol	M	RC	O	RC
docsDevFilterIpSourcePortLow	M	RC	O	RC
docsDevFilterIpSourcePortHigh	M	RC	O	RC
docsDevFilterIpDestPortLow	M	RC	O	RC
docsDevFilterIpDestPortHigh	M	RC	O	RC
docsDevFilterIpMatches	M	RO	O	RO
docsDevFilterIpTos	M	RC	O	RC

docsDevFilterIpTosMask	M	RC	O	RC
docsDevFilterIpContinue	M	RC	O	RC
docsDevFilterIpPolicyId	M	RC	O	RC
docsDevFilterPolicyTable				
Objects	**CM**	**Access**	**CMTS**	**Access**
docsDevFilterPolicyIndex	M	N-Acc	O	N-Acc
docsDevFilterPolicyId	M	RC	O	RC
docsDevFilterPolicyStatus	M	RC	O	RC
docsDevFilterPolicyPtr	M	RC	O	RC
docsDevFilterTosTable				
Objects	**CM**	**Access**	**CMTS**	**Access**
docsDevFilterTosIndex	M	N-Acc	O	N-Acc
docsDevFilterTosStatus	M	RC	O	RC
docsDevFilterTosAndMask	M	RC	O	RC
docsDevFilterTosOrMask	M	RC	O	RC
docsDevCpeGroup				
NOTE: CM supporting IP spoofing function MUST implement this group. CM not supporting IP spoofing filter MUST NOT implement this group.				
Objects	**CM**	**Access**	**CMTS**	**Access**
docsDevCpeEnroll	O	RW	N-Sup	
docsDevCpeIpMax	O	RW	N-Sup	
docsDevCpeTable				
Objects	**CM**	**Access**	**CMTS**	**Access**
docsDevCpeIp	O	N-Acc	N-Sup	
docsDevCpeSource	O	RO	N-Sup	
docsDevCpeStatus	O	RC	N-Sup	
IP-MIB (RFC 2011)				
IP Group				
Objects	**CM**	**Access**	**CMTS**	**Access**
ipForwarding	M	RW	M	RW
ipDefaultTTL	M	RW	M	RW
ipInreceives	M	RO	M	RO
ipInHdrErrors	M	RO	M	RO
ipInAddrErrors	M	RO	M	RO
ipForwDatagrams	M	RO	M	RO
ipinUnknownProtos	M	RO	M	RO

ipInDiscards	M	RO	M	RO
ipInDelivers	M	RO	M	RO
ipOutRequest	M	RO	M	RO
ipOutDiscard	M	RO	M	RO
ipOutNoRoutes	M	RO	M	RO
ipReasmTimeout	M	RO	M	RO
ipReasmReqds	M	RO	M	RO
ipReasmOKs	M	RO	M	RO
ipReasmFails	M	RO	M	RO
ipFragOKs	M	RO	M	RO
ipFragFails	M	RO	M	RO
ipFragCreates	M	RO	M	RO
ipAddrTable				
Objects	**CM**	**Access**	**CMTS**	**Access**
ipAdEntAddr	M	RO	M	RO
ipAdEntIfIndex	M	RO	M	RO
ipAdEntNetMask	M	RO	M	RO
ipAdEntBcastAddr	M	RO	M	RO
ipAdEntReasmMaxSize	M	RO	M	RO
IpNetToMediaTable				
Objects	**CM**	**Access**	**CMTS**	**Access**
ipNetToMediaIfIndex	M	RC	M	RC
ipNetToMediaPhysAddress	M	RC	M	RC
ipNetToMediaNetAddress	M	RC	M	RC
ipNetToMediaType	M	RC	M	RC
6.5.1.1.1 Objects				
ipRoutingDiscards	M	RO	M	RO
ICMP Group				
Objects	**CM**	**Access**	**CMTS**	**Access**
icmpInMsgs	M	RO	M	RO
icmpInErrors	M	RO	M	RO
icmpInDestUnreachs	M	RO	M	RO
icmpInTimeExcds	M	RO	M	RO
icmpInParmProbs	M	RO	M	RO
icmpInSrcQuenchs	M	RO	M	RO
icmpInRedirects	M	RO	M	RO
icmpInEchos	M	RO	M	RO
icmpInEchosReps	M	RO	M	RO
icmpInTimestamps	M	RO	M	RO
icmpInTimeStampreps	M	RO	M	RO
icmpInAddrMasks	M	RO	M	RO
icmpInAddrMaskReps	M	RO	M	RO

icmpOutMsgs	M	RO	M	RO
icmpOutErrors	M	RO	M	RO
icmpOutDestUnreachs	M	RO	M	RO
icmpOutTimeExcds	M	RO	M	RO
icmpOutParmProbs	M	RO	M	RO
icmpOutSrcQuenchs	M	RO	M	RO
icmpOutRedirects	M	RO	M	RO
icmpOutEchoes	M	RO	M	RO
icmpOutEchoReps	M	RO	M	RO
icmpOutTimestamps	M	RO	M	RO
icmpOutTimestampReps	M	RO	M	RO
icmpOutAddrMasks	M	RO	M	RO
icmpOutAddrMaskReps	M	RO	M	RO
UDP-MIB (RFC 2013)				
UDP Group				
Objects	**CM**	**Access**	**CMTS**	**Access**
udpInDatagrams	M	RO	M	RO
udpNoPorts	M	RO	M	RO
udpInErrors	M	RO	M	RO
udpOutDatagrams	M	RO	M	RO
UDP Listener Table				
Objects	**CM**	**Access**	**CMTS**	**Access**
udpLocalAddress	M	RO	M	RO
udpLocalPort	M	RO	M	RO
SNMPv2-MIB (RFC 1907)				
System Group				
Objects	**CM**	**Access**	**CMTS**	**Access**
sysDescr	M	RO	M	RO
sysObjectID	M	RO	M	RO
sysUpTime	M	RO	M	RO
sysContact	M	RW	M	RW
sysName	M	RW	M	RW
sysLocation	M	RW	M	RW
sysServices	M	RO	M	RO
sysORLastChange	M	RO	M	RO

sysORTable				
Object	**CM**	**Access**	**CMTS**	**Access**
sysORIndex	M	N-Acc	M	N-Acc
sysORID	M	RO	M	RO
sysORDescr	M	RO	M	RO
sysORUpTime	M	RO	M	RO
SNMP Group				
Objects	**CM**	**Access**	**CMTS**	**Access**
snmpInPkts	M	RO	M	RO
SnmpInBadVersions	M	RO	M	RO
snmpOutPkts	Ob	RO	Ob	RO
snmpInBadCommunityNames	M	RO	M	RO
snmpInBadCommunityUses	M	RO	M	RO
snmpInASNParseErrs	M	RO	M	RO
snmpInTooBigs	Ob	RO	Ob	RO
snmpInNoSuchNames	Ob	RO	Ob	RO
snmpInBadValues	Ob	RO	Ob	RO
snmpInReadOnlys	Ob	RO	Ob	RO
snmpInGenErrs	Ob	RO	Ob	RO
snmpInTotalReqVars	Ob	RO	Ob	RO
snmpInTotalSetVars	Ob	RO	Ob	RO
snmpInGetRequests	Ob	RO	Ob	RO
snmpInGetNexts	Ob	RO	Ob	RO
snmpInSetRequests	Ob	RO	Ob	RO
snmpInGetResponses	Ob	RO	Ob	RO
snmpInTraps	Ob	RO	Ob	RO
snmpOutTooBigs	Ob	RO	Ob	RO
snmpOutNoSuchNames	Ob	RO	Ob	RO
snmpOutBadValues	Ob	RO	Ob	RO
snmpOutGenErrs	Ob	RO	Ob	RO
snmpOutGetRequests	Ob	RO	Ob	RO
snmpOutGetNexts	Ob	RO	Ob	RO
snmpOutSetRequests	Ob	RO	Ob	RO
snmpOutGetResponses	Ob	RO	Ob	RO
snmpOutTraps	Ob	RO	Ob	RO
snmpEnableAuthenTraps	M	RW	M	RW
snmpSilentDrops	M	RO	M	RO
snmpProxyDrops	M	RO	M	RO
6.5.1.1.1.1 Object	CM	Access	CMTS	Access
6.5.1.1.1.2 snmpSetSerialNo	M	RW	M	RW

Etherlike-MIB (RFC 2665)				
dot3StatsTable				
Objects	**CM**	**Access**	**CMTS**	**Access**
dot3StatsIndex	M	RO	M	RO
dot3StatsAlignmentErrors	M	RO	M	RO
dot3StatsFCSErrors	M	RO	M	RO
dot3StatsSingleCollisionFrames	M	RO	M	RO
dot3StatsMultipleCollisionFrames	M	RO	M	RO
dot3StatsSQETestErrors	M	RO	M	RO
dot3StatsDeferredTransmissions	M	RO	M	RO
dot3StatsLateCollisions	M	RO	M	RO
dot3StatsExcessiveCollisions	M	RO	M	RO
dot3StatsInternalMacTransmitErrors	M	RO	M	RO
dot3StatsCarrierSenseErrors	M	RO	M	RO
dot3StatsFrameTooLongs	M	RO	M	RO
dot3StatsInternalMacReceiveErrors	M	RO	M	RO
dot3StatsEtherChipSet	D	RO	D	RO
dot3StatsSymbolErrors	M	RO	M	RO
dot3StatsDuplexStatus	M	RO	M	RO
dot3CollTable				
Objects	**CM**	**Access**	**CMTS**	**Access**
dot3CollCount	O	NA	O	NA
dot3CollFrequencies	O	RO	O	RO
dot3ControlTable				
Objects	**CM**	**Access**	**CMTS**	**Access**
dot3ControlFunctionsSupported	O	RO	O	RO
dot3ControlInUnknownOpcodes	O	RO	O	RO
dot3PauseTable				
Objects	**CM**	**Access**	**CMTS**	**Access**
dot3PauseAdminMode	O	RW	O	RW
dot3PauseOperMode	O	RO	O	RO
dot3InPauseFrames	O	RO	O	RO
dot3OutPauseFrames	O	RO	O	RO
USB MIB				
NOTE: This MIB is required for CM that supports USB only.				
Object	**CM**	**Access**	**CMTS**	**Access**
usbNumber	M	RO	NA	

usbPortTable				
Object	**CM**	**Access**	**CMTS**	**Access**
usbPortIndex	M	RO	NA	
usbPortType	M	RO	NA	
usbPortRate	M	RO	NA	
usbDeviceTable				
Object	**CM**	**Access**	**CMTS**	**Access**
usbDeviceIndex	M	RO	NA	
usbDevicePower	M	RO	NA	
usbDeviceVendorID	M	RO	NA	
usbDeviceProductID	M	RO	NA	
usbDeviceNumberConfigurations	M	RO	NA	
usbDeviceActiveClass	M	RO	NA	
usbDeviceStatus	M	RO	NA	
usbDeviceEnumCounter	M	RO	NA	
usbDeviceRemoteWakeup	M	RO	NA	
usbDeviceRemoteWakeupOn	M	RO	NA	
usbCDCTable				
Object	**CM**	**Access**	**CMTS**	**Access**
usbCDCIndex	M	RO	NA	
usbCDCIfIndex	M	RO	NA	
usbCDCSubclass	M	RO	NA	
usbCDCVersion	M	RO	NA	
usbCDCDataTransferType	M	RO	NA	
usbCDCDataEndpoints	M	RO	NA	
usbCDCStalls	M	RO	NA	
usbCDCEtherTable				
Object	**CM**	**Access**	**CMTS**	**Access**
usbCDCEtherIndex	M	RO	NA	
usbCDCEtherIfIndex	M	RO	NA	
usbCDCEtherMacAddress	M	RO	NA	
usbCDCEtherPacketFilter	M	RO	NA	
usbCDCEtherDataStatisticsCapabilities	M	RO	NA	
usbCDCEtherDataCheckErrs	M	RO	NA	
DOCS-QOS-MIB (draft-ietf-ipcdn-qos-mib-04.txt)				
NOTE: 1.1 CM in 1.0 mode MUST NOT support this MIB.				
docsQosPktClassTable				
Object	**1.1 CM in 1.1 mode**	**Access**	**CMTS**	**Access**
docsQosPktClassId	M	N-Acc	M	N-Acc
docsQosPktClassDirection	M	RO	M	RO

docsQosPktClassPriority	M	RO	M	RO
docsQosPktClassIpTosLow	M	RO	M	RO
docsQosPktClassIpTosHigh	M	RO	M	RO
docsQosPktClassIpTosMask	M	RO	M	RO
docsQosPktClassIpProtocol	M	RO	M	RO
docsQosPktClassIpSourceAddr	M	RO	M	RO
docsQosPktClassIpSourceMask	M	RO	M	RO
docsQosPktClassIpDestAddr	M	RO	M	RO
docsQosPktClassIpDestMask	M	RO	M	RO
docsQosPktClassSourcePortStart	M	RO	M	RO
docsQosPktClassSourcePortEnd	M	RO	M	RO
docsQosPktClassDestPortStart	M	RO	M	RO
docsQosPktClassDestPortEnd	M	RO	M	RO
docsQosPktClassDestMacAddr	M	RO	M	RO
docsQosPktClassDestMacMask	M	RO	M	RO
docsQosPktClassSourceMacAddr	M	RO	M	RO
docsQosPktClassEnetProtocolType	M	RO	M	RO
docsQosPktClassEnetProtocol	M	RO	M	RO
docsQosPktClassUserPriApplies	M	RO	M	RO
docsQosPktClassUserPriLow	M	RO	M	RO
docsQosPktClassUserPriHigh	M	RO	M	RO
docsQosPktClassVlanId	M	RO	M	RO
docsQosPktClassState	M	RO	M	RO
docsQosPktClassPkts	M	RO	M	RO
DocsQosParamSetTable when **docsQosParamSetRowType** = **serviceFlow (1)**				
Object	**1.1 CM in 1.1 mode**	**Access**	**CMTS**	**Access**
docsQosParamSetRowType **= serviceFlow (1)**	M	N-Acc	M	N-Acc
docsQosParamSetIndex	M	N-Acc	M	N-Acc
docsQosParamSetRowStatus	M	RO	M	RO
docsQosParamSetServiceClassName	M	RO	M	RO
docsQosParamSetPriority	M	RO	M	RO
docsQosParamSetMaxTrafficRate	M	RO	M	RO
docsQosParamSetMaxTrafficBurst	M	RO	M	RO
docsQosParamSetMinReservedRate	M	RO	M	RO
docsQosParamSetMinReservedPkt	M	RO	M	RO
docsQosParamSetActiveTimeout	M	RO	M	RO
docsQosParamSetAdmittedTimeout	M	RO	M	RO
docsQosParamSetMaxConcatBurst	M	RO	M	RO
DocsQosParamSetSchedulingType	M	RO	M	RO
docsQosParamSetRequestPolicy	M	RO	M	RO
docsQosParamSetNomPollInterval	M	RO	M	RO
docsQosParamSetTolPollJitter	M	RO	M	RO
docsQosParamSetUnsolicitGrantSize	M	RO	M	RO
docsQosParamSetNomGrantInterval	M	RO	M	RO

docsQosParamSetTolGrantJitter	M	RO	M	RO
docsQosParamSetGrantsPerInterval	M	RO	M	RO
docsQosParamSetTosAndMask	M	RO	M	RO
docsQosParamSetTosOrMask	M	RO	M	RO
docsQosParamSetMaxLatency	M	RO	M	RO
docsQosParamSetTable when **docsQosParamSetRowType** = **serviceClass (2).**				
Object	**1.1 CM in 1.1 mode**	**Access**	**CMTS**	**Access**
DocsQosParamSetRowType **= serviceClass (2)**	M	N-Acc	M	N-Acc
docsQosParamSetIndex	M	N-Acc	M	N-Acc
docsQosParamSetRowStatus	M	RO	M	RC
docsQosParamSetServiceClassName	M	RO	M	RO
docsQosParamSetPriority	M	RO	M	RC
docsQosParamSetMaxTrafficRate	M	RO	M	RC
docsQosParamSetMaxTrafficBurst	M	RO	M	RC
docsQosParamSetMinReservedRate	M	RO	M	RC
docsQosParamSetMinReservedPkt	M	RO	M	RC
docsQosParamSetActiveTimeout	M	RO	M	RC
docsQosParamSetAdmittedTimeout	M	RO	M	RC
docsQosParamSetMaxConcatBurst	M	RO	M	RC
DocsQosParamSetSchedulingType	M	RO	M	RC
docsQosParamSetRequestPolicy	M	RO	M	RC
docsQosParamSetNomPollInterval	M	RO	M	RC
docsQosParamSetTolPollJitter	M	RO	M	RC
docsQosParamSetUnsolicitGrantSize	M	RO	M	RC
docsQosParamSetNomGrantInterval	M	RO	M	RC
docsQosParamSetTolGrantJitter	M	RO	M	RC
docsQosParamSetGrantsPerInterval	M	RO	M	RC
docsQosParamSetTosAndMask	M	RO	M	RC
docsQosParamSetTosOrMask	M	RO	M	RC
docsQosParamSetMaxLatency	M	RO	M	RC
docsQosServiceFlowTable				
Object	**1.1 CM in 1.1 mode**	**Access**	**CMTS**	**Access**
docsQosServiceFlowId	M	N-Acc	M	N-Acc
docsQosServiceFlowProvisionedParamSetIndex	M	RO	M	RO
docsQosServiceFlowAdmittedParamSetIndex	M	RO	M	RO
docsQosServiceFlowActiveParamSetIndex	M	RO	M	RO
docsQosServiceFlowSID	M	RO	M	RO
docsQosServiceFlowDirection	M	RO	M	RO
docsQosServiceFlowPrimary	M	RO	M	RO
DocsQosServiceFlowStatsTable				
Object	**1.1 CM in 1.1 mode**	**Access**	**CMTS**	**Access**
docsQosServiceFlowPkts	M	RO	M	RO
docsQosServiceFlowOctets	M	RO	M	RO

docsQosServiceFlowTimeCreated	M	RO	M	RO
docsQosServiceFlowTimeActive	M	RO	M	RO
docsQosServiceFlowPHSUnknowns	M	RO	M	RO
docsQosServiceFlowPolicedDropPkts	M	RO	M	RO
docsQosServiceFlowPolicedDelayPkts	M	RO	M	RO
docsQosUpstreamStatsTable				
Object	**1.1 CM in 1.1 mode**	**Access**	**CMTS**	**Access**
docsQosSID	N-Sup		M	N-Acc
docsQosUpstreamFragPkts	N-Sup		M	RO
docsQosUpstreamIncompletePkts	N-Sup		M	RO
docsQosUpstreamConcatBursts	N-Sup		M	RO
docsQosDynamicServiceStatsTable				
Object	**1.1 CM in 1.1 mode**	**Access**	**CMTS**	**Access**
docsQosIfDirection	M	N-Acc	M	N-Acc
docsQosDSAReqs	M	RO	M	RO
docsQosDSARsps	M	RO	M	RO
docsQosDSAAcks	M	RO	M	RO
DocsQosDSCReqs	M	RO	M	RO
docsQosDSCRsps	M	RO	M	RO
docsQosDSCAcks	M	RO	M	RO
docsQosDSDReqs	M	RO	M	RO
docsQosDSDRsps	M	RO	M	RO
docsQosDynamicAdds	M	RO	M	RO
docsQosDynamicAddFails	M	RO	M	RO
DocsQosDynamicChanges	M	RO	M	RO
docsQosDynamicChangeFails	M	RO	M	RO
DocsQosDynamicDeletes	M	RO	M	RO
DocsQosDynamicDeleteFails	M	RO	M	RO
docsQosServiceFlowLogTable				
Object	**1.1 CM in 1.1 mode**	**Access**	**CMTS**	**Access**
docsQosServiceFlowLogIndex	N-Sup		M	N-Acc
docsQosServiceFlowLogIfIndex	N-Sup		M	RO
docsQosServiceFlowLogSFID	N-Sup		M	RO
docsQosServiceFlowLogCmMac	N-Sup		M	RO
docsQosServiceFlowLogPkts	N-Sup		M	RO
docsQosServiceFlowLogOctets	N-Sup		M	RO
docsQosServiceFlowLogTimeDeleted	N-Sup		M	RO
DocsQosServiceFlowLogTimeCreated	N-Sup		M	RO
docsQosServiceFlowLogTimeActive	N-Sup		M	RO
DocsQosServiceFlowLogControl	N-Sup		M	RW

docsQosServiceClassTable				
Object	**1.1 CM in 1.1 mode**	**Access**	**CMTS**	**Access**
docsQosServiceClassName	N-Sup		M	N-Acc
docsQosServiceClassParamSetIndex	N-Sup		M	RC
docsQosServiceClassStatus	N-Sup		M	RC
docsQosServiceClassPolicyTable				
Object	**1.1 CM in 1.1 mode**	**Access**	**CMTS**	**Access**
docsQosServiceClassPolicyIndex	O	N-Acc	O	N-Acc
docsQosServiceClassPolicyName	O	RC	O	RC
docsQosServiceClassPolicyRulePriority	O	RC	O	RC
docsQosServiceClassPolicyStatus	O	RC	O	RC
docsQosPHSTable				
Object	**1.1 CM in 1.1 mode**	**Access**	**CMTS**	**Access**
docsQosPHSIndex	O	N-Acc	O	N-Acc
docsQosPHSField	O	RO	O	RO
docsQosPHSMask	O	RO	O	RO
docsQosPHSSize	O	RO	O	RO
docsQosPHSVerify	O	RO	O	RO
docsQosPHSClassifierIndex	O	RO	O	RO
docsQosCmtsMacToSrvFlowTable				
Object	**1.1 CM in 1.1 mode**	**Access**	**CMTS**	**Access**
docsQosCmtsCmMac	N-Sup		M	N-Acc
docsQosCmtsServiceFlowId	N-Sup		M	N-Acc
docsQosCmtsIfIndex	N-Sup		M	RO
DOCS-SUBMGT-MIB (draft-ietf-ipcdn-subscriber-mib-02.txt) Subscriber Management MIB				
docsSubMgtCpeControlTable				
Object	**CM**	**Access**	**CMTS**	**Access**
docsSubMgtCpeControlMaxCpeIp	NA	NA	M	RW
docsSubMgtCpeControlActive	NA	NA	M	RW
docsSubMgtCpeControlLearnable	NA	NA	M	RW
docsSubMgtCpeControlReset	NA	NA	M	RW
docsSubMgtCpeMaxIpDefault	NA	NA	M	RW
DocsSubMgtCpeActiveDefault	NA	NA	M	RW
docsSubMgtCpeIpTable				
Object	**CM**	**Access**	**CMTS**	**Access**
DocsSubMgtCpeIpIndex	NA	NA	M	N-Acc
DocsSubMgtCpeIpAddr	NA	NA	M	RO

DocsSubMgtCpeIpLearned			NA	NA	M	RO
docsSubMgtPktFilterTable						
Object			**CM**	**Access**	**CMTS**	**Access**
DocsSubMgtPktFilterGroup			NA	NA	M	N-Acc
docsSubMgtPktFilterIndex			NA	NA	M	N-Acc
docsSubMgtPktFilterSrcAddr			NA	NA	M	RC
docsSubMgtPktFilterSrcMask			NA	NA	M	RC
DocsSubMgtPktFilterDstAddr			NA	NA	M	RC
docsSubMgtPktFilterDstMask			NA	NA	M	RC
DocsSubMgtPktFilterUlp			NA	NA	M	RC
docsSubMgtPktFilterTosValue			NA	NA	M	RC
docsSubMgtPktFilterTosMask			NA	NA	M	RC
DocsSubMgtPktFilterAction			NA	NA	M	RC
docsSubMgtPktFilterMatches			NA	NA	M	RO
docsSubMgtPktFilterStatus			NA	NA	M	RC
docsSubMgtTcpUdpFilterTable						
Object			**CM**	**Access**	**CMTS**	**Access**
DocsSubMgtTcpUdpSrcPort			NA	NA	M	RC
docsSubMgtTcpUdpDstPort			NA	NA	M	RC
docsSubMgtTcpFlagValues			NA	NA	M	RC
docsSubMgtTcpFlagMask			NA	NA	M	RC
DocsSubMgtTcpUdpStatus			NA	NA	M	RC
docsSubMgtCmFilterTable						
Object			**CM**	**Access**	**CMTS**	**Access**
docsSubMgtSubFilterDownstream			NA	NA	M	RW
docsSubMgtSubFilterUpstream			NA	NA	M	NW
docsSubMgtCmFilterDownstream			NA	NA	M	RW
DocsSubMgtCmFilterUpstream			NA	NA	M	RW
Object			**CM**	**Access**	**CMTS**	**Access**
docsSubMgtSubFilterDownDefault			NA	NA	M	RW
DocsSubMgtSubFilterUpDefault			NA	NA	M	RW
docsSubMgtCmFilterDownDefault			NA	NA	M	RW
docsSubMgtCmFilterUpDefault			NA	NA	M	RW
IGMP-STD-MIB (RFC 2933)						
This MIB is optional for Bridging CMTS						
NOTE: 1.1 CM in 1.0 mode is not required to implement RFC-2933						
IgmpInterfaceTable						
Object	**1.1 CM in 1.0 mode**	**Access**	**1.1 CM in 1.1 mode**	**Access**	**CMTS**	**Access**

IgmpInterfaceIfIndex	O	N-Acc	M	N-Acc	M	N-Acc
igmpInterfaceQueryInterval	O	RC	M	RC	M	RC
igmpInterfaceStatus	O	RC	M	RC	M	RC
igmpInterfaceVersion	O	RC	M	RC	M	RC
igmpInterfaceQuerier	O	RO	M	RO	M	RO
igmpInterfaceQueryMaxResponseTime	O	RO	M	RO	M	RO
igmpInterfaceVersion1QuerierTimer	O	RO	M	RO	M	RO
igmpInterfaceWrongVersionQueries	O	RO	M	RO	M	RO
igmpInterfaceJoins	O	RO	M	RO	M	RO
IgmpInterfaceGroups	O	RO	M	RO	M	RO
igmpInterfaceRobustness	O	RC	M	RC	M	RC
igmpInterfaceLastMembQueryIntvl	O	RC	M	RC	M	RC
igmpInterfaceProxyIfIndex	O	RC	M	RC	M	RC
igmpInterfaceQuerierUpTime	O	RO	M	RO	M	RO
igmpInterfaceQuerierExpiryTime	O	RO	M	RO	M	RO
igmpCacheTable						
Object	**1.1 CM in 1.0 mode**	**Access**	**1.1 CM in 1.1 mode**	**Access**	**CMTS**	**Access**
igmpCacheAddress	O	N-Acc	M	N-Acc	M	N-Acc
igmpCacheIfIndex	O	N-Acc	M	N-Acc	M	N-Acc
IgmpCacheSelf	O	RC	M	RC	M	RC
igmpCacheLastReporter	O	RO	M	RO	M	RO
igmpCacheUpTime	O	RO	M	RO	M	RO
igmpCacheExpiryTime	O	RO	M	RO	M	RO
igmpCacheStatus	O	RC	M	RC	M	RC
igmpCacheVersion1HostTimer	O	RO	M	RO	M	RO
Account Management MIB (MIB defining work is still in progress.)						
docsCpeSegmentTable						
Object			**CM**	**Access**	**CMTS**	**Access**
docsCpeSegmentID			NA	NA	O	RO
docsCpeSegmentIp			NA	NA	O	RC
DocsCpeTrafficData Table						
Object			**CM**	**Access**	**CMTS**	**Access**
docsCpeIpAddress			NA	NA	O	RO
docsCpeTrafficDataUpStreamPackets			NA	NA	O	RC
docsCpeTrafficDataDownStreamPackets			NA	NA	O	RC
docsCpeTrafficDataUpStreamOctets			NA	NA	O	RC
docsCpeTrafficDataDownStreamOctets			NA	NA	O	RC
docsCpeTrafficDataUpStreamDropPackets			NA	NA	O	RC
docsCpeTrafficDataDownStreamDropPackets			NA	NA	O	RC
docsCmCpeTable			**CM**	**Access**	**CMTS**	**Access**
docsCmMacAddress			NA	NA	O	RC

docsCmIpAddress	NA	NA	O	RC
docsCpeMACAddress	NA	NA	O	RC
docsCpeIpAddress	NA	NA	O	RC

Object	**1.1CM in 1.0 mode**	**Access**	**1.1 CM in 1.1 mode**	**Access**	**CMTS**	**Access**
DOCS-BPI-MIB (draft-ietf-ipcdn-mcns-bpi-mib-01.txt)						
docsBpiCmBaseTable						
Object	**1.1CM in 1.0 mode**	**Access**	**1.1 CM in 1.1 mode**	**Access**	**CMTS**	**Access**
docsBpiCmPrivacyEnable	M	RO	N-Sup		NA	
docsBpiCmPublicKey	M	RO	N-Sup		NA	
docsBpiCmAuthState	M	RO	N-Sup		NA	
docsBpiCmAuthKeySequenceNumber	M	RO	N-Sup		NA	
docsBpiCmAuthExpires	M	RO	N-Sup		NA	
docsBpiCmAuthReset	M	RW	N-Sup		NA	
docsBpiCmAuthGraceTime	M	RO	N-Sup		NA	
docsBpiCmTEKGraceTime	M	RO	N-Sup		NA	
docsBpiCmAuthWaitTimeout	M	RO	N-Sup		NA	
docsBpiCmReauthWaitTimeout	M	RO	N-Sup		NA	
docsBpiCmOpWaitTimeout	M	RO	N-Sup		NA	
docsBpiCmRekeyWaitTimeout	M	RO	N-Sup		NA	
docsBpiCmAuthRejectWaitTimeout	M	RO	N-Sup		NA	
docsBpiCmAuthRequests	M	RO	N-Sup		NA	
docsBpiCmAuthReplies	M	RO	N-Sup		NA	
docsBpiCmAuthRejects	M	RO	N-Sup		NA	
docsBpiCmAuthInvalids	M	RO	N-Sup		NA	
docsBpiCmAuthRejectErrorCode	M	RO	N-Sup		NA	
docsBpiCmAuthRejectErrorString	M	RO	N-Sup		NA	
docsBpiCmAuthInvalidErrorCode	M	RO	N-Sup		NA	
docsBpiCmAuthInvalidErrorString	M	RO	N-Sup		NA	

docsBpiCmTEKTable						
Object	**1.1CM in 1.0 mode**	**Access**	**1.1 CM in 1.1 mode**	**Access**	**CMTS**	**Access**
docsBpiCmTEKPrivacyEnable	M	RO	N-Sup		NA	
docsBpiCmTEKState	M	RO	N-Sup		NA	
docsBpiCmTEKExpiresOld	M	RO	N-Sup		NA	
docsBpiCmTEKExpiresNew	M	RO	N-Sup		NA	
docsBpiCmTEKKeyRequests	M	RO	N-Sup		NA	
docsBpiCmTEKKeyReplies	M	RO	N-Sup		NA	
docsBpiCmTEKKeyRejects	M	RO	N-Sup		NA	
docsBpiCmTEKInvalids	M	RO	N-Sup		NA	
docsBpiCmTEKAuthPends	M	RO	N-Sup		NA	
docsBpiCmTEKKeyRejectErrorCode	M	RO	N-Sup		NA	
docsBpiCmTEKKeyRejectErrorString	M	RO	N-Sup		NA	
docsBpiCmTEKInvalidErrorCode	M	RO	N-Sup		NA	
docsBpiCmTEKInvalidErrorString	M	RO	N-Sup		NA	
docsBpiCmtsBaseTable						
Object	**1.1CM in 1.0 mode**	**Access**	**1.1 CM in 1.1 mode**	**Access**	**CMTS**	**Access**
docsBpiCmtsDefaultAuthLifetime	NA		NA		N-Sup	
docsBpiCmtsDefaultTEKLifetime	NA		NA		N-Sup	
docsBpiCmtsAuthRequests	NA		NA		N-Sup	
docsBpiCmtsAuthReplies	NA		NA		N-Sup	
docsBpiCmtsAuthRejects	NA		NA		N-Sup	
docsBpiCmtsAuthInvalids	NA		NA		N-Sup	
docsBpiCmtsAuthTable						
Object	**1.1CM in 1.0 mode**	**Access**	**1.1 CM in 1.1 mode**	**Access**	**CMTS**	**Access**
docsBpiCmtsAuthCmMacAddress	NA		NA		N-Sup	
docsBpiCmtsAuthCmPublicKey	NA		NA		N-Sup	
docsBpiCmtsAuthCmKeySequenceNumber	NA		NA		N-Sup	
docsBpiCmtsAuthCmExpires	NA		NA		N-Sup	
docsBpiCmtsAuthCmLifetime	NA		NA		N-Sup	
docsBpiCmtsAuthCmGraceTime	NA		NA		N-Sup	
docsBpiCmtsAuthCmReset	NA		NA		N-Sup	
docsBpiCmtsAuthCmRequests	NA		NA		N-Sup	
docsBpiCmtsAuthCmReplies	NA		NA		N-Sup	
docsBpiCmtsAuthCmRejects	NA		NA		N-Sup	
docsBpiCmtsAuthCmInvalids	NA		NA		N-Sup	
docsBpiCmtsAuthRejectErrorCode	NA		NA		N-Sup	
docsBpiCmtsAuthRejectErrorString	NA		NA		N-Sup	
docsBpiCmtsAuthInvalidErrorCode	NA		NA		N-Sup	
docsBpiCmtsAuthInvalidErrorString	NA		NA		N-Sup	

docsBpiCmtsTEKTable						
Object	**1.1CM in 1.0 mode**	**Access**	**1.1 CM in 1.1 mode**	**Access**	**CMTS**	**Access**
docsBpiCmtsTEKLifetime	NA		NA		N-Sup	
docsBpiCmtsTEKGraceTime	NA		NA		N-Sup	
docsBpiCmtsTEKExpiresOld	NA		NA		N-Sup	
docsBpiCmtsTEKExpiresNew	NA		NA		N-Sup	
docsBpiCmtsTEKReset	NA		NA		N-Sup	
docsBpiCmtsKeyRequests	NA		NA		N-Sup	
docsBpiCmtsKeyReplies	NA		NA		N-Sup	
docsBpiCmtsKeyRejects	NA		NA		N-Sup	
docsBpiCmtsTEKInvalids	NA		NA		N-Sup	
docsBpiCmtsKeyRejectErrorCode	NA		NA		N-Sup	
docsBpiCmtsKeyRejectErrorString	NA		NA		N-Sup	
docsBpiCmtsTEKInvalidErrorCode	NA		NA		N-Sup	
docsBpiCmtsTEKInvalidErrorString	NA		NA		N-Sup	
docsBpiIpMulticastMapTable						
Object	**1.1CM in 1.0 mode**	**Access**	**1.1 CM in 1.1 mode**	**Access**	**CMTS**	**Access**
docsBpiIpMulticastAddress	NA		NA		N-Sup	
docsBpiIpMulticastprefixLength	NA		NA		N-Sup	
docsBpiIpMulticastServiceId	NA		NA		N-Sup	
docsBpiIpMulticastMapControl	NA		NA		N-Sup	
docsBpiMulticastAuthTable						
Object	**1.1CM in 1.0 mode**	**Access**	**1.1 CM in 1.1 mode**	**Access**	**CMTS**	**Access**
docsBpiMulticastServiceId	NA		NA		N-Sup	
docsBpiMulticastCmMacAddress	NA		NA		N-Sup	
docsBpiMulticastAuthControl	NA		NA		N-Sup	
DOCS-BPI2-MIB (draft-ietf-ipcdn-bpiplus-mib-03.txt)						
docsBpi2CmBaseTable						
Object	**1.1CM in 1.0 mode**	**Access**	**1.1 CM in 1.1 mode**	**Access**	**CMTS**	**Access**
docsBpi2CmPrivacyEnable	O	RO	M	RO	NA	
docsBpi2CmPublicKey	O	RO	M	RO	NA	
docsBpi2CmAuthState	O	RO	M	RO	NA	
docsBpi2CmAuthKeySequenceNumber	O	RO	M	RO	NA	
docsBpi2CmAuthExpiresOld	O	RO	M	RO	NA	
docsBpi2CmAuthExpiresNew	O	RO	M	RO	NA	
docsBpi2CmAuthReset	O	RW	M	RW	NA	

docsBpi2CmAuthGraceTime	O	RO	M	RO	NA	
docsBpi2CmTEKGraceTime	O	RO	M	RO	NA	
docsBpi2CmAuthWaitTimeout	O	RO	M	RO	NA	
docsBpi2CmReauthWaitTimeout	O	RO	M	RO	NA	
docsBpi2CmOpWaitTimeout	O	RO	M	RO	NA	
docsBpi2CmRekeyWaitTimeout	O	RO	M	RO	NA	
docsBpi2CmAuthRejectWaitTimeout	O	RO	M	RO	NA	
docsBpi2CmSAMapWaitTimeout	O	RO	M	RO	NA	
docsBpi2CmSAMapMaxRetries	O	RO	M	RO	NA	
docsBpi2CmAuthentInfos	O	RO	M	RO	NA	
docsBpi2CmAuthRequests	O	RO	M	RO	NA	
docsBpi2CmAuthReplies	O	RO	M	RO	NA	
docsBpi2CmAuthRejects	O	RO	M	RO	NA	
docsBpi2CmAuthInvalids	O	RO	M	RO	NA	
docsBpi2CmAuthRejectErrorCode	O	RO	M	RO	NA	
docsBpi2CmAuthRejectErrorString	O	RO	M	RO	NA	
docsBpi2CmTEKTable						
Object	**1.1CM in 1.0 mode**	**Access**	**1.1 CM in 1.1 mode**	**Access**	**CMTS**	**Access**
docsBpi2CmTEKSAId	O	RO	M	RO	NA	
docsBpi2CmTEKSAType	O	RO	M	RO	NA	
docsBpi2CmTEKDataEncryptAlg	O	RO	M	RO	NA	
docsBpi2CmTEKDataAuthentAlg	O	RO	M	RO	NA	
docsBpi2CmTEKState	O	RO	M	RO	NA	
docsBpi2CmTEKKeySequenceNumber	O	RO	M	RO	NA	
docsBpi2CmTEKExpiresOld	O	RO	M	RO	NA	
docsBpi2CmTEKExpiresNew	O	RO	M	RO	NA	
docsBpi2CmTEKKeyRequests	O	RO	M	RO	NA	
docsBpi2CmTEKKeyReplies	O	RO	M	RO	NA	
docsBpi2CmTEKKeyRejects	O	RO	M	RO	NA	
docsBpi2CmTEKInvalids	O	RO	M	RO	NA	
docsBpi2CmTEKAuthPends	O	RO	M	RO	NA	
docsBpi2CmTEKKeyRejectErrorCode	O	RO	M	RO	NA	
docsBpi2CmTEKKeyRejectErrorString	O	RO	M	RO	NA	
docsBpi2CmTEKInvalidErrorCode	O	RO	M	RO	NA	
docsBpi2CmTEKInvalidErrorString	O	RO	M	RO	NA	
docsBpi2CmIpMulticastMapTable						
Object	**1.1CM in 1.0 mode**	**Access**	**1.1 CM in 1.1 mode**	**Access**	**CMTS**	**Access**
docsBpi2CmIpMulticastAddress	O	N-Acc	M	N-Acc	NA	
docsBpi2CmIpMulticastSAId	O	RO	M	RO	NA	
docsBpi2CmIpMulticastSAMapState	O	RO	M	RO	NA	
docsBpi2CmIpMulticastSAMapRequests	O	RO	M	RO	NA	
docsBpi2CmIpMulticastSAMapReplies	O	RO	M	RO	NA	
docsBpi2CmIpMulticastSAMapRejects	O	RO	M	RO	NA	

docsBpi2CmIpMulticastSAMapRejectErrorCode	O	RO	M	RO	NA	
docsBpi2CmIpMulticastSAMapRejectErrorString	O	RO	M	RO	NA	
docsBpi2CmDeviceCertTable						
Object	**1.1CM in 1.0 mode**	**Access**	**1.1 CM in 1.1 mode**	**Access**	**CMTS**	**Access**
docsBpi2CmDeviceCmCert	M	RW	M	RW	NA	
docsBpi2CmDeviceManufCert	M	RO	M	RO	NA	
docsBpi2CmCryptoSuiteTable						
Object	**1.1CM in 1.0 mode**	**Access**	**1.1 CM in 1.1 mode**	**Access**	**CMTS**	**Access**
docsBpi2CmCryptoSuiteIndex	M	N-Acc	M	N-Acc	NA	
docsBpi2CmCryptoSuiteDataEncryptAlg	M	RO	M	RO	NA	
docsBpi2CmCryptoSuiteDataAuthentAlg	M	RO	M	RO	NA	
docsBpi2CmtsBaseEntryTable						
Object	**1.1CM in 1.0 mode**	**Access**	**1.1 CM in 1.1 mode**	**Access**	**CMTS**	**Access**
docsBpi2CmtsDefaultAuthLifetime	NA		NA		M	RW
docsBpi2CmtsDefaultTEKLifetime	NA		NA		M	RW
docsBpi2CmtsDefaultSelfSignedManufCertTrust	NA		NA		M	RW
docsBpi2CmtsCheckCertValidityPeriods	NA		NA		M	RW
docsBpi2CmtsAuthentInfos	NA		NA		M	RO
docsBpi2CmtsAuthRequests	NA		NA		M	RO
docsBpi2CmtsAuthReplies	NA		NA		M	RO
docsBpi2CmtsAuthRejects	NA		NA		M	RO
docsBpi2CmtsAuthInvalids	NA		NA		M	RO
docsBpi2CmtsSAMapRequests	NA		NA		M	RO
docsBpi2CmtsSAMapReplies	NA		NA		M	RO
docsBpi2CmtsSAMapRejects	NA		NA		M	RO
docsBpi2CmtsAuthEntryTable						
Object	**1.1CM in 1.0 mode**	**Access**	**1.1 CM in 1.1 mode**	**Access**	**CMTS**	**Access**
docsBpi2CmtsAuthCmMacAddress	NA		NA		M	N-Acc
docsBpi2CmtsAuthCmBpiVersion	NA		NA		M	RO
docsBpi2CmtsAuthCmPublicKey	NA		NA		M	RO
docsBpi2CmtsAuthCmKeySequenceNumber	NA		NA		M	RO

docsBpi2CmtsAuthCmExpiresOld	NA		NA		M	RO
docsBpi2CmtsAuthCmExpiresNew	NA		NA		M	RO
docsBpi2CmtsAuthCmLifetime	NA		NA		M	RW
docsBpi2CmtsAuthCmGraceTime	NA		NA		M	RO
docsBpi2CmtsAuthCmReset	NA		NA		M	RW
docsBpi2CmtsAuthCmInfos	NA		NA		M	RO
docsBpi2CmtsAuthCmRequests	NA		NA		M	RO
docsBpi2CmtsAuthCmReplies	NA		NA		M	RO
docsBpi2CmtsAuthCmRejects	NA		NA		M	RO
docsBpi2CmtsAuthCmInvalids	NA		NA		M	RO
docsBpi2CmtsAuthRejectErrorCode	NA		NA		M	RO
docsBpi2CmtsAuthRejectErrorString	NA		NA		M	RO
docsBpi2CmtsAuthInvalidErrorCode	NA		NA		M	RO
docsBpi2CmtsAuthInvalidErrorString	NA		NA		M	RO
docsBpi2CmtsAuthPrimarySAId	NA		NA		M	RO
docsBpi2CmtsAuthBpkmCmCertValid	NA		NA		M	RO
docsBpi2CmtsAuthBpkmCmCert	NA		NA		M	RO
docsBpi2CmtsTEKTable						
Object	**1.1CM in 1.0 mode**	**Access**	**1.1 CM in 1.1 mode**	**Access**	**CMTS**	**Access**
docsBpi2CmtsTEKSAId	NA		NA		M	N-Acc
docsBpi2CmtsTEKSAType	NA		NA		M	RO
docsBpi2CmtsTEKDataEncryptAlg	NA		NA		M	RO
docsBpi2CmtsTEKDataAuthentAlg	NA		NA		M	RO
docsBpi2CmtsTEKLifetime	NA		NA		M	RW
docsBpi2CmtsTEKGraceTime	NA		NA		M	RO
docsBpi2CmtsTEKKeySequenceNumber	NA		NA		M	RO
docsBpi2CmtsTEKExpiresOld	NA		NA		M	RO
docsBpi2CmtsTEKExpiresNew	NA		NA		M	RO
docsBpi2CmtsTEKReset	NA		NA		M	RW
docsBpi2CmtsKeyRequests	NA		NA		M	RO
docsBpi2CmtsKeyReplies	NA		NA		M	RO
docsBpi2CmtsKeyRejects	NA		NA		M	RO
docsBpi2CmtsTEKInvalids	NA		NA		M	RO
docsBpi2CmtsKeyRejectErrorCode	NA		NA		M	RO
docsBpi2CmtsKeyRejectErrorString	NA		NA		M	RO
docsBpi2CmtsTEKInvalidErrorCode	NA		NA		M	RO
docsBpi2CmtsTEKInvalidErrorString	NA		NA		M	RO
docsBpi2CmtsIpMulticastMapTable						
Object	**1.1CM in 1.0 mode**	**Access**	**1.1 CM in 1.1 mode**	**Access**	**CMTS**	**Access**
docsBpi2CmtsIpMulticastAddress	NA		NA		M	N-Acc
docsBpi2CmtsIpMulticastPrefixLength	NA		NA		M	N-Acc
docsBpi2CmtsIpMulticastSAId	NA		NA		M	RC/RO

docsBpi2CmtsIpMulticastSAMapRequests	NA		NA		M	RO
docsBpi2CmtsIpMulticastSAMapReplies	NA		NA		M	RO
docsBpi2CmtsIpMulticastSAMapRejects	NA		NA		M	RO
docsBpi2CmtsIpMulticastSAMapRejectErrorCode	NA		NA		M	RO
docsBpi2CmtsIpMulticastSAMapRejectErrorString	NA		NA		M	RO
docsBpi2CmtsIpMulticastMapControl	NA		NA		M	RC/RO
docsBpi2CmtsMulticastAuthTable						
Object	**1.1CM in 1.0 mode**	**Access**	**1.1 CM in 1.1 mode**	**Access**	**CMTS**	**Access**
docsBpi2CmtsMulticastAuthSAId	NA		NA		M	N-Acc
docsBpi2CmtsMulticastAuthCmMacAddress	NA		NA		M	N-Acc
docsBpi2CmtsMulticastAuthControl	NA		NA		M	RC/RO
docsBpi2CmtsProvisionedCmCertTable						
Object	**1.1CM in 1.0 mode**	**Access**	**1.1 CM in 1.1 mode**	**Access**	**CMTS**	**Access**
docsBpi2CmtsProvisionedCmCertMacAddress	NA		NA		M	N-Acc
docsBpi2CmtsProvisionedCmCertTrust	NA		NA		M	RC
docsBpi2CmtsProvisionedCmCertSource	NA		NA		M	RO
docsBpi2CmtsProvisionedCmCertStatus	NA		NA		M	RC
docsBpi2CmtsProvisionedCmCert}	NA		NA		M	RC
docsBpi2CmtsCACertTable						
Object	**1.1CM in 1.0 mode**	**Access**	**1.1 CM in 1.1 mode**	**Access**	**CMTS**	**Access**
docsBpi2CmtsCACertIndex	NA		NA		M	N-Acc
docsBpi2CmtsCACertSubject	NA		NA		M	RO
docsBpi2CmtsCACertIssuer	NA		NA		M	RO
docsBpi2CmtsCACertSerialNumber	NA		NA		M	RO
docsBpi2CmtsCACertTrust	NA		NA		M	RC
docsBpi2CmtsCACertSource	NA		NA		M	RO
docsBpi2CmtsCACertStatus	NA		NA		M	RC
docsBpi2CmtsCACert	NA		NA		M	RC

docsBpi2CodeDownloadGroup						
Object	**1.1CM in 1.0 mode**	**Access**	**1.1 CM in 1.1 mode**	**Access**	**CMTS**	**Access**
docsBpi2CodeDownloadStatusCode,	M	RO	M	RO	O	RO
docsBpi2CodeMfgCodeDownloadStatusString,	M	RO	M	RO	O	RO
docsBpi2CodeMfgOrgName,	M	RO	M	RO	O	RO
docsBpi2CodeMfgCodeAccessStart,	M	RO	M	RO	O	RO
docsBpi2CodeMfgCvcAccessStart,	M	RO	M	RO	O	RO
docsBpi2CodeCoSignerOrgName,	M	RO	M	RO	O	RO
docsBpi2CodeCoSignerCodeAccessStart,	M	RO	M	RO	O	RO
docsBpi2CodeCoSignerCvcAccessStart,	M	RO	M	RO	O	RO
docsBpi2CodeCvcUpdate	M	RW	M	RW	O	RW
SNMP-USM-DH-OBJECTS-MIB (RFC 2786)						
NOTE: **SNMP-USM-DH-OBJECTS-MIB is only accessible when the device is in SNMP Coexistence Mode.**						
Object			**CM**	**Access**	**CMTS**	**Access**
usmDHParameters			M	RW	O	RW
usmDHUserKeyTable						
Object			**CM**	**Access**	**CMTS**	**Access**
usmDHUserAuthKeyChange			M	RC	O	RC
UsmDHUserOwnAuthKeyChange			M	RC	O	RC
usmDHUserPrivKeyChange			M	RC	O	RC
usmDHUserOwnPrivKeyChange			M	RC	O	RC
usmDHKickstartTable						
Object			**CM**	**Access**	**CMTS**	**Access**
usmDHKickstartIndex			M	N-Acc	O	N-Acc
usmDHKickstartMyPublic			M	RO	O	RO
usmDHKickstartMgrPublic			M	RO	O	RO
usmDHKickstartSecurityName			M	RO	O	RO
SNMP-VIEW-BASED-ACM-MIB (RFC2575)						
(Note: SNMP-VIEW-BASED-ACM-MIB is ONLY accessible when the device is in SNMP Coexistence mode.)						
Object			**CM**	**Access**	**CMTS**	**Access**
vacmContextTable						
vacmContextName			M	RO	M	RO

Object	CM	Access	CMTS	Access
vacmSecurityToGroupTable				
vacmSecurityModel	M	N-Acc	M	N-Acc
vacmSecurityName	M	N-Acc	M	N-Acc
vacmGroupName	M	RC	M	RC
vacmSecurityToGroupStorageType	M	RC	M	RC
vacmSecurityToGroupStatus	M	RC	M	RC
Object	**CM**	**Access**	**CMTS**	**Access**
vacmAccessTable				
vacmAccessContextPrefix	M	N-Acc	M	N-Acc
vacmAccessSecurityModel	M	N-Acc	M	N-Acc
vacmAccessSecurityLevel	M	N-Acc	M	N-Acc
vacmAccessContextMatch	M	RC	M	RC
vacmAccessReadViewName	M	RC	M	RC
vacmAccessWriteViewName	M	RC	M	RC
vacmAccessNotifyViewName	M	RC	M	RC
vacmAccessStorageType	M	RC	M	RC
vacmAccessStatus	M	RC	M	RC
vacmViewSpinLock	M	RW	M	RW
Object	**CM**	**Access**	**CMTS**	**Access**
vacmViewTreeFamilyTable				
vacmViewTreeFamilyViewName	M	N-Acc	M	N-Acc
vacmViewTreeFamilySubtree	M	N-Acc	M	N-Acc
vacmViewTreeFamilyMask	M	RC	M	RC
vacmViewTreeFamilyType	M	RC	M	RC
vacmViewTreeFamilyStorageType	M	RC	M	RC
vacmViewTreeFamilyStatus	M	RC	M	RC
SNMP-COMMUNITY-MIB (RFC2576)				
(Note: SNMP-COMMUNITY-MIB is ONLY accessible when the device is in SNMP Coexistence mode.)				
Object	**CM**	**Access**	**CMTS**	**Access**
snmpCommunityTable				
snmpCommunityIndex	M	N-Acc	M	N-Acc
snmpCommunityName	M	RC	M	RC
snmpCommunitySecurityName	M	RC	M	RC

snmpCommunityContextEngineID	M	RC	M	RC
snmpCommunityContextName	M	RC	M	RC
snmpCommunityTransportTag	M	RC	M	RC
snmpCommunityStorageType	M	RC	M	RC
snmpCommunityStatus	M	RC	M	RC
Object	**CM**	**Access**	**CMTS**	**Access**
SnmpTargetExtTable				
snmpTargetAddrTMask	M	RC	M	RC
snmpTargetAddrMMS	M	RC	M	RC
snmpTrapAddress	O	ACC-FN	O	ACC-FN
snmpTrapCommunity	O	ACC-FN	O	ACC-FN
SNMP Management Framework architecture (RFC2571)				
Object	**CM**	**Access**	**CMTS**	**Access**
snmpEngine Group				
snmpEngineID	M	RO	M	RO
snmpEngineBoots	M	RO	M	RO
snmpEngineTime	M	RO	M	RO
snmpEngineMaxMessageSize	M	RO	M	RO
SNMP Message Processing and Dispatching MIB (RFC-2572)				
(Note: SNMP Message Processing and Dispatching MIB is ONLY accessible when the device is in SNMP Coexistence mode.)				
Object	**CM**	**Access**	**CMTS**	**Access**
snmpMPDStats				
snmpUnknownSecurityModels	M	RO	M	RO
snmpInvalidMsgs	M	RO	M	RO
snmpUnknownPDUHandlers	M	RO	M	RO
(RFC-2573)				
(Note: RFC-2573 is ONLY accessible when the device is in SNMP Coexistence mode.)				
Object	**CM**	**Access**	**CMTS**	**Access**
snmpTargetSpinLock	M	RW	M	RW

snmpTargetAddrTable				
Object	**CM**	**Access**	**CMTS**	**Access**
snmpTargetAddrName	M	N-Acc	M	N-Acc
snmpTargetAddrTDomain	M	RC	M	RC
SnmpTargetAddrTAddress	M	RC	M	RC
SnmpTargetAddrTimeout	M	RC	M	RC
SnmpTargetAddrRetryCount	M	RC	M	RC
SnmpTargetAddrTagList	M	RC	M	RC
SnmpTargetAddrParams	M	RC	M	RC
SnmpTargetAddrStorageType	M	RC	M	RC
SnmpTargetAddrRowStatus	M	RC	M	RC
snmpTargetParamsTable				
Object	**CM**	**Access**	**CMTS**	**Access**
SnmpTargetParamsName	M	N-Acc	M	N-Acc
SnmpTargetParamsMPModel	M	RC	M	RC
SnmpTargetParamsSecurityModel	M	RC	M	RC
SnmpTargetParamsSecurityName	M	RC	M	RC
SnmpTargetParamsSecurityLevel	M	RC	M	RC
SnmpTargetParamsStorageType	M	RC	M	RC
SnmpTargetParamsRowStatus	M	RC	M	RC
SnmpUnavailableContexts		RO	M	RO
snmpUnknownContexts	M	RO	M	RO
snmpNotifyTable				
Object	**CM**	**Access**	**CMTS**	**Access**
snmpNotifyName	M	N-Acc	M	N-Acc
snmpNotifyTag	M	RC	M	RC
SnmpNotifyType	M	RC	M	RC
snmpNotifyStorageType	M	RC	M	RC
SnmpNotifyRowStatus	M	RC	M	RC
snmpNotifyFilterProfileTable				
Object	**CM**	**Access**	**CMTS**	**Access**
SnmpNotifyFilterProfileName	M	RC	M	RC
snmpNotifyFilterProfileStorType	M	RC	M	RC
snmpNotifyFilterProfileRowStatus	M	RC	M	RC
snmpNotifyFilterTable				

Object	**CM**	**Access**	**CMTS**	**Access**
SnmpNotifyFilterSubtree	M	N-Acc	M	N-Acc
SnmpNotifyFilterMask	M	RC	M	RC
SnmpNotifyFilterType	M	RC	M	RC
SnmpNotifyFilterStorageType	M	RC	M	RC
SnmpNotifyFilterRowStatus	M	RC	M	RC
(RFC-2574)				
(Note: RFC-2574 MIB is ONLY accessible when the device is in SNMP Coexistence mode.)				
usmStats				
Object	**CM**	**Access**	**CMTS**	**Access**
usmStatsUnsupportedSecLevels	M	RO	M	RO
UsmStatsNotInTimeWindows	M	RO	M	RO
UsmStatsUnknownUserNames	M	RO	M	RO
UsmStatsUnknownEngineIDs	M	RO	M	RO
UsmStatsWrongDigests	M	RO	M	RO
UsmStatsDecryptionErrors	M	RO	M	RO
usmUser				
Object	**CM**	**Access**	**CMTS**	**Access**
usmUserSpinLock	M	RW	M	RW
usmUserTable				
Object	**CM**	**Access**	**CMTS**	**Access**
usmUserEngineID	M	N-Acc	M	N-Acc
UsmUserName	M	N-Acc	M	N-Acc
UsmUserSecurityName	M	RO	M	RO
UsmUserCloneFrom	M	RC	M	RC
usmUserAuthProtocol	M	RC	M	RC
UsmUserAuthKeyChange	M	RC	M	RC
UsmUserOwnAuthKeyChange	M	RC	M	RC
UsmUserPrivProtocol	M	RC	M	RC
UsmUserPrivKeyChange	M	RC	M	RC
UsmUserOwnPrivKeyChange	M	RC	M	RC
UsmUserPublic	M	RC	M	RC
UsmUserStorageType	M	RC	M	RC
UsmUserStatus	M	RC	M	RC

APPENDIX A2. RFC-2670 ifTable MIB-Object details

RFC-2670 MIB-Object details for Cable Device using 10 Meg Ethernet	CMTS-Ethernet-10	CMTS-MAC	CMTS-Downstream	CMTS-Upstream	CM-Ethernet-10	CM-MAC	CM-Downstream	CM-Upstream	CM-USB	CM-CPE Other Type
ifIndex:"A unique value, greater than zero, for each interface. It is recommended that values are assigned contiguously starting from 1. [The Primary CPE MUST be Interface number 1] The value for each interface sub-layer must remain constant at least from one reinitialization of the entity's network management system to the next reinitialization."	(n)	(n)	(n)	(n)	[1 or 4+(n)]	2	3	4	[1 or 4+(n)]	[1 or 4+(n)]
ifType:"The type of interface. Additional values for ifType are assigned by the Internet Assigned Numbers Authority (IANA), through updating the syntax of the IANAifType textual convention."	6	127	128	129	6	127	128	129	160	(IANA num)

RFC-2670 MIB-Object details for Cable Device using 10 Meg Ethernet	CMTS-Ethernet-10	CMTS-MAC	CMTS-Downstream	CMTS-Upstream	CM-Ethernet-10	CM-MAC	CM-Downstream	CM-Upstream	CM-USB	CM-CPE Other Type
ifSpeed:"An estimate of the interface's current bandwidth in bits per second. [For RF Downstream; This is the symbol rate multiplied with the number of bits per symbol. For RF Upstream; This is the raw bandwidth in bits per second of this interface, regarding the highest speed modulation profile that is defined. This is the symbol rate multiplied with the number of bits per symbol for this modulation profile. For MAC Layer; Return zero.] For interfaces which do not vary in bandwidth or for those where no accurate estimation can be made, this object should contain the nominal bandwidth. If the bandwidth of the interface is greater than the maximum value reportable by this object then this object should report its maximum value (4,294,967,295) and ifHighSpeed must be used to report the interace's speed. For a sub-layer which has no concept of bandwidth, this object should be zero."	10,000,000	0	~64-QAM=30,341,646, ~256-QUAM=42,884,296	(n)	10,000,000	0	~64-QAM=30,341,646, ~256-QUAM=42,884,296	(n)	12,500,000	speed

RFC-2670 MIB-Object details for Cable Device using 10 Meg Ethernet	CMTS-Ethernet-10	CMTS-MAC	CMTS-Downstream	CMTS-Upstream	CM-Ethernet-10	CM-MAC	CM-Downstream	CM-Upstream	CM-USB	CM-CPE Other Type
ifHighSpeed:"An estimate of the interface's current bandwidth in units of 1,000,000 bits per second. If this object reports a value of `n' then the speed of the interface is somewhere in the range of `n-500,000' to `n+499,999'. [For RF Downstream; This is the symbol rate multiplied with the number of bits per symbol. For RF Upstream; This is the raw bandwidth in bits per second of this interface, regarding the highest speed modulation profile that is defined. This is the symbol rate multiplied with the number of bits per symbol for this modulation profile. For MAC Layer; Return zero.] For interfaces which do not vary in bandwidth or for those where no accurate estimation can be made, this object should contain the nominal bandwidth. For a sub-layer which has no concept of bandwidth, this object should be zero."	10	0	~64-QAM=30, ~256-QUAM=42	(n)	10	0	~64-QAM=30, ~256-QUAM=42	(n)	12	speed
ifPhysAddress:"The interface's address at its protocol sub-layer. [For RF Upstream/Downstream; return empty string. For MAC Layer; return the physical address of this interface.] For example, for an 802.x interface, this object normally contains a MAC address. The interface's media-specific MIB must define the bit and byte ordering and the format of the value of this object. For interfaces which do not have such an address (e.g., a serial line), this object should contain an octet string of zero length."	Enet-MAC	CATV-MAC	Empty - String	Empty - String	Enet-MAC	CATV-MAC	Empty - String	Empty - String	USB-Phys Addr.	Phys Addr.
ifAdminStatus:"The desired state of the interface. The testing(3) state indicates that no operational packets can be passed. When a managed system initializes, all interfaces start with ifAdminStatus in the up(1) state.	Up(1), Down(2), Testing(3)	Up(1), Down(2), Testing(3)	Up(1), Down(2), Testing(3)	Up(1), Down(2), Testing(3)	Up(1), Down(2), Testing(3)	Up(1), Down(2), Testing(3)	Up(1), Down(2), Testing(3)	Up(1), Down(2), Testing(3)	Up(1), Down(2), Testing(3)	Up(1), Down(2), Testing(3)

RFC-2670 MIB-Object details for Cable Device using 10 Meg Ethernet	CMTS-Ethernet-10	CMTS-MAC	CMTS-Downstream	CMTS-Upstream	CM-Ethernet-10	CM-MAC	CM-Downstream	CM-Upstream	CM-USB	CM-CPE Other Type
As a result of either explicit management action or per configuration information retained by the managed system, ifAdminStatus is then changed to either the down(2) or testing(3) states (or remains in the up(1) state)."										
ifOperStatus:"The current operational state of the interface. The testing(3) state indicates that no operational packets can be passed. If ifAdminStatus is down(2) then ifOperStatus should be down(2). If ifAdminStatus is changed to up(1) then ifOperStatus should change to up(1) if the interface is ready to transmit and receive network traffic; it should change to dormant(5) if the interface is waiting for external actions (such as a serial line waiting for an incoming connection); it should remain in the down(2) state if and only if there is a fault that prevents it from going to the up(1) state; it should remain in the notPresent(6) state if the interface has missing (typically, hardware) components."	Up(1), Down(2), Testing(3)	Up(1), Down(2), Testing(3)	Up(1), Down(2), Testing(3)	Up(1), Down(2), Testing(3)	Up(1), Down(2), Testing(3)	Up(1), Down(2), Testing(3)	Up(1), Down(2), Testing(3)	Up(1), Down(2), Testing(3)	Up(1), Down(2), Testing(3)	Up(1), Down(2), Testing(3)
ifMtu:"The size of the largest packet which can be sent/received on the interface, specified in octets. [For RF Upstream/Downstream; the value includes the length of the MAC header. For MAC Layer; return 1500.] For interfaces that are used for transmitting network datagrams, this is the size of the largest network datagram that can be sent on the interface."	1500	1500	1764	1764	1500	1500	1764	1764	1500	1500?
ifInOctets:"The total number of octets received on the interface, including framing characters. [For RF Upstream/Downstream (where not zero[2]); This includes MAC packets as	(n)	(n)	MUST be 0	(n)	(n)	(n) = low 32-bits of the	(n) = low 32-bits of the	MUST be 0	(n)	(n)

[2] * The ifEntry for Downstream interfaces supports the ifGeneralInformationGroup and the ifPacketGroup of the Interfaces MIB. This is an output only interface at the CMTS and all input status counters – ifIn* - will return zero. This is an input only interface at the CM and all output status counters – ifOut* - will return

RFC-2670 MIB-Object details for Cable Device using 10 Meg Ethernet	CMTS-Ethernet-10	CMTS-MAC	CMTS-Downstream	CMTS-Upstream	CM-Ethernet-10	CM-MAC	CM-Downstream	CM-Upstream	CM-USB	CM-CPE Other Type
well as data packets, and includes the length of the MAC header. For MAC Layer; The total number of data octets received on this interface, targeted for upper protocol layers.] Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of ifCounterDiscontinuityTime."						64-bit count	64-bit count			
IfHCInOctets: (usage[3]) "The total number of octets received on the interface, including framing characters. [For RF Upstream/Downstream (where not zero[*]); This includes MAC packets as well as data packets, and includes the length of the MAC header. For MAC Layer; The total number of data octets received on this interface, targeted for upper protocol layers.] This object is a 64-bit version of ifInOctets. Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of ifCounterDiscontinuityTime."	0 or (n) = 64-bit count[4]	0 or (n) = 64-bit count ***	MUST be 0	0 or (n) = 64-bit count ***	0 or (n) = 64-bit count ***	(n) = 64-bit count	(n) = 64-bit count	MUST be 0	0 or (n) = 64-bit count ***	0 or (n) = 64-bit count ***
ifOutOctets:"The total number of	(n)	(n) =	(n) =	MUST	(n)	(n)	MUST	(n)	(n)	(n)

zero. The ifEntry for Upstream interfaces supports the ifGeneralInformationGroup and the ifPacketGroup of the Interfaces MIB. This is an input only interface at the CMTS and all output status counters – ifOut* - will return zero. This is an output only interface at the CM and all input status counters – ifIn* - will return zero.

[3] ** For interfaces that operate at 20,000,000 (20 million) bits per second or less, 32-bit byte and packet counters MUST be used. For interfaces that operate faster than 20,000,000 bits/second, and slower than 650,000,000 bits/second, 32-bit packet counters MUST be used and 64-bit octet counters MUST be used. For interfaces that operate at 650,000,000 bits/second or faster, 64-bit packet counters AND 64-bit octet counters MUST be used. When 64-bit counters are in use, the 32-bit counters MUST still be available. The 32-bit counters report the low 32-bits of the associated 64-bit count (e.g., ifInOctets will report the least significant 32 bits of ifHCInOctets). This enhances inter-operability with existing implementations at a very minimal cost to agents.

[4] *** If the optional 64-bit counter is implemented then the corresponding 32-bit counter MUST represent the low 32-bits of the associated 64-bit counter.

RFC-2670 MIB-Object details for Cable Device using 10 Meg Ethernet	CMTS-Ethernet-10	CMTS-MAC	CMTS-Downstream	CMTS-Upstream	CM-Ethernet-10	CM-MAC	CM-Downstream	CM-Upstream	CM-USB	CM-CPE Other Type
octets transmitted out of the interface, including framing characters. [For RF Upstream/ Downstream (where not zero*); This includes MAC packets as well as data packets, and includes the length of the MAC header. For MAC Layer; The total number of octets, received from upper protocol layers and transmitted on this interface.] Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of ifCounterDiscontinuityTime."		low 32-bits of the 64-bit count	low 32-bits of the 64-bit count	be 0			be 0			
ifHCOutOctets: (usage**) "The total number of octets transmitted out of the interface, including framing characters. [For RF Upstream/ Downstream (where not zero*); This includes MAC packets as well as data packets, and includes the length of the MAC header. For MAC Layer; The total number of octets, received from upper protocol layers and transmitted on this interface.] This object is a 64-bit version of ifOutOctets. Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of ifCounterDiscontinuityTime."	0 or (n) = 64-bit count ***	(n) = 64-bit count	(n) = 64-bit count	MUST be 0	0 or (n) = 64-bit count ***	0 or (n) = 64-bit count ***	MUST be 0	0 or (n) = 64-bit count ***	0 or (n) = 64-bit count ***	0 or (n) = 64-bit count ***
ifInUcastPkts:"The number of packets, delivered by this sub-layer to a higher (sub-)layer, which were not addressed to a multicast or broadcast address at this sub-layer. [For RF Upstream/ Downstream (where not zero*); This includes data packets as well as MAC layer packets. For MAC Layer; The number of Unicast packets received on this interface, targeted for upper protocol layers.] Discontinuities in the value of this counter can occur at re-initialization of the management	(n)	(n)	MUST be 0	(n)	(n)	(n)	(n)	MUST be 0	(n)	(n)

RFC-2670 MIB-Object details for Cable Device using 10 Meg Ethernet	CMTS-Ethernet-10	CMTS-MAC	CMTS-Downstream	CMTS-Upstream	CM-Ethernet-10	CM-MAC	CM-Downstream	CM-Upstream	CM-USB	CM-CPE Other Type
system, and at other times as indicated by the value of ifCounterDiscontinuityTime."										
ifHCInUcastPkts:"The number of packets, delivered by this sub-layer to a higher (sub-)layer, which were not addressed to a multicast or broadcast address at this sub-layer. [For RF Upstream/ Downstream (where not zero*); This includes data packets as well as MAC layer packets. For MAC Layer; The number of Unicast packets received on this interface, targeted for upper protocol layers.] This object is a 64-bit version of ifInUcastPkts. Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of ifCounterDiscontinuityTime."	0 or (n) = 64-bit count ***	0 or (n) = 64-bit count ***	MUST be 0	0 or (n) = 64-bit count ***	0 or (n) = 64-bit count ***	0 or (n) = 64-bit count ***	0 or (n) = 64-bit count ***	MUST be 0	0 or (n) = 64-bit count ***	0 or (n) = 64-bit count ***
ifInMulticastPkts:"The number of packets, delivered by this sub-layer to a higher (sub-)layer, which were addressed to a multicast address at this sub-layer. [For RF Upstream/ Downstream (where not zero*); This includes data packets as well as MAC layer packets. For MAC Layer; The number of Multicast packets received on this interface, targeted for upper protocol layers.] For a MAC layer protocol, this includes both Group and Functional addresses. Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of ifCounterDiscontinuityTime."	(n)	(n)	MUST be 0	(n)	(n)	(n)	(n)	MUST be 0	(n)	(n)
ifHCInMulticastPkts:"The number of packets, delivered by this sub-layer to a higher (sub-)layer, which were addressed to a multicast address at this sub-layer. [For RF Upstream/	0 or (n) = 64-bit count ***	0 or (n) = 64-bit count ***	MUST be 0	0 or (n) = 64-bit count ***	0 or (n) = 64-bit count ***	0 or (n) = 64-bit count ***	0 or (n) = 64-bit count ***	MUST be 0	0 or (n) = 64-bit count ***	0 or (n) = 64-bit count ***

RFC-2670 MIB-Object details for Cable Device using 10 Meg Ethernet	CMTS-Ethernet-10	CMTS-MAC	CMTS-Downstream	CMTS-Upstream	CM-Ethernet-10	CM-MAC	CM-Downstream	CM-Upstream	CM-USB	CM-CPE Other Type
Downstream (where not zero*); This includes data packets as well as MAC layer packets. For MAC Layer; The number of Multicast packets received on this interface, targeted for upper protocol layers.] For a MAC layer protocol, this includes both Group and Functional addresses. This object is a 64-bit version of ifInMulticastPkts. Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of ifCounterDiscontinuityTime."	***	***		***	***	***	***		***	***
ifInBroadcastPkts:"The number of packets, delivered by this sub-layer to a higher (sub-)layer, which were addressed to a broadcast address at this sub-layer. [For RF Upstream/ Downstream (where not zero*); This includes data packets as well as MAC layer packets. For MAC Layer; The number of Broadcast packets received on this interface, targeted for upper protocol layers.] Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of ifCounterDiscontinuityTime."	(n)	(n)	MUST be 0	(n)	(n)	(n)	(n)	MUST be 0	(n)	(n)
ifHCInBroadcastPkts:"The number of packets, delivered by this sub-layer to a higher (sub-)layer, which were addressed to a broadcast address at this sub-layer. [For RF Upstream/ Downstream (where not zero*); This includes data packets as well as MAC layer packets. For MAC Layer; The number of Broadcast packets received on this interface, targeted for upper protocol layers.] This object is a 64-bit version of ifInBroadcastPkts. Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as	0 or (n) = 64-bit count ***	0 or (n) = 64-bit count ***	MUST be 0	0 or (n) = 64-bit count ***	0 or (n) = 64-bit count ***	0 or (n) = 64-bit count ***	0 or (n) = 64-bit count ***	MUST be 0	0 or (n) = 64-bit count ***	0 or (n) = 64-bit count ***

RFC-2670 MIB-Object details for Cable Device using 10 Meg Ethernet	CMTS-Ethernet-10	CMTS-MAC	CMTS-Downstream	CMTS-Upstream	CM-Ethernet-10	CM-MAC	CM-Downstream	CM-Upstream	CM-USB	CM-CPE Other Type
indicated by the value of ifCounterDiscontinuityTime."										
ifInDiscards:"The number of inbound packets which were chosen to be discarded even though no errors had been detected to prevent their being deliverable to a higher-layer protocol. One possible reason for discarding such a packet could be to free up buffer space. Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of ifCounterDiscontinuityTime."	(n)	(n)	MUST be 0	(n)	(n)	(n)	(n)	MUST be 0	(n)	(n)
ifInErrors:"For packet-oriented interfaces, the number of inbound packets that contained errors preventing them from being deliverable to a higher-layer protocol. For character-oriented or fixed-length interfaces, the number of inbound transmission units that contained errors preventing them from being deliverable to a higher-layer protocol. Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of ifCounterDiscontinuityTime."	(n)	(n)	MUST be 0	(n)	(n)	(n)	(n)	MUST be 0	(n)	(n)
ifInUnknownProtos:"For packet-oriented interfaces, the number of packets received via the interface which were discarded because of an unknown or unsupported protocol. For character-oriented or fixed-length interfaces that support protocol multiplexing the number of transmission units received via the interface which were discarded because of an unknown or unsupported protocol. For any interface that does not support protocol multiplexing, this counter will	(n)	(n)	MUST be 0	(n)	(n)	(n)	(n)	MUST be 0	(n)	(n)

RFC-2670 MIB-Object details for Cable Device using 10 Meg Ethernet	CMTS-Ethernet-10	CMTS-MAC	CMTS-Downstream	CMTS-Upstream	CM-Ethernet-10	CM-MAC	CM-Downstream	CM-Upstream	CM-USB	CM-CPE Other Type
always be 0. Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of ifCounterDiscontinuityTime."										
ifOutUcastPkts:"The total number of packets that higher-level protocols requested be transmitted, and which were not addressed to a multicast or broadcast address at this sub-layer, including those that were discarded or not sent. [For RF Upstream/ Downstream (where not zero*); This includes MAC packets as well as data packets. For MAC Layer; The number of Unicast packets, received from upper protocol layers and transmitted on this interface.] Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of ifCounterDiscontinuityTime."	(n)	(n)	(n)	MUST be 0	(n)	(n)	MUST be 0	(n)	(n)	(n)
ifHCOutUcastPkts:"The total number of packets that higher-level protocols requested be transmitted, and which were not addressed to a multicast or broadcast address at this sub-layer, including those that were discarded or not sent. [For RF Upstream/ Downstream (where not zero*); This includes MAC packets as well as data packets. For MAC Layer; The number of Unicast packets, received from upper protocol layers and transmitted on this interface.] This object is a 64-bit version of ifOutUcastPkts. Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of ifCounterDiscontinuityTime."	0 or (n) = 64-bit count ***	0 or (n) = 64-bit count ***	0 or (n) = 64-bit count ***	MUST be 0	0 or (n) = 64-bit count ***	0 or (n) = 64-bit count ***	MUST be 0	0 or (n) = 64-bit count ***	0 or (n) = 64-bit count ***	0 or (n) = 64-bit count ***
ifOutMulticastPkts:"The total number of packets that higher-level protocols requested be transmitted, and which	(n)	(n)	(n)	MUST be 0	(n)	(n)	MUST be 0	(n)	(n)	(n)

RFC-2670 MIB-Object details for Cable Device using 10 Meg Ethernet	CMTS-Ethernet-10	CMTS-MAC	CMTS-Downstream	CMTS-Upstream	CM-Ethernet-10	CM-MAC	CM-Downstream	CM-Upstream	CM-USB	CM-CPE Other Type
were addressed to a multicast address at this sub-layer, including those that were discarded or not sent. [For RF Upstream/ Downstream (where not zero*); This includes MAC packets as well as data packets. For MAC Layer; The number of Multicast packets received from upper protocol layers and transmitted on this interface.] For a MAC layer protocol, this includes both Group and Functional addresses. Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of ifCounterDiscontinuityTime."										
ifHCOutMulticastPkts:"The total number of packets that higher-level protocols requested be transmitted, and which were addressed to a multicast address at this sub-layer, including those that were discarded or not sent. [For RF Upstream/ Downstream (where not zero*); This includes MAC packets as well as data packets. For MAC Layer; The number of Multicast packets received from upper protocol layers and transmitted on this interface.] For a MAC layer protocol, this includes both Group and Functional addresses. This object is a 64-bit version of ifOutMulticastPkts. Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of ifCounterDiscontinuityTime."	0 or (n) = 64-bit count ***	0 or (n) = 64-bit count ***	0 or (n) = 64-bit count ***	MUST be 0	0 or (n) = 64-bit count ***	0 or (n) = 64-bit count ***	MUST be 0	0 or (n) = 64-bit count ***	0 or (n) = 64-bit count ***	0 or (n) = 64-bit count ***
ifOutBroadcastPkts:"The total number of packets that higher-level protocols requested be transmitted, and which were addressed to a broadcast address at this sub-layer, including those that were discarded	(n)	(n)	(n)	MUST be 0	(n)	(n)	MUST be 0	(n)	(n)	(n)

RFC-2670 MIB-Object details for Cable Device using 10 Meg Ethernet	CMTS-Ethernet-10	CMTS-MAC	CMTS-Downstream	CMTS-Upstream	CM-Ethernet-10	CM-MAC	CM-Downstream	CM-Upstream	CM-USB	CM-CPE Other Type
or not sent. [For RF Upstream/ Downstream (where not zero*); This includes MAC packets as well as data packets. For MAC Layer; The number of Broadcast packets, received from upper protocol layers and transmitted on this interface.] Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of ifCounterDiscontinuityTime."										
ifHCOutBroadcastPkts:"The total number of packets that higher-level protocols requested be transmitted, and which were addressed to a broadcast address at this sub-layer, including those that were discarded or not sent. [For RF Upstream/ Downstream (where not zero*); This includes MAC packets as well as data packets. For MAC Layer; The number of Broadcast packets, received from upper protocol layers and transmitted on this interface.] This object is a 64-bit version of ifOutBroadcastPkts. Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of ifCounterDiscontinuityTime."	0 or (n) = 64-bit count ***	0 or (n) = 64-bit count ***	0 or (n) = 64-bit count ***	MUST be 0	0 or (n) = 64-bit count ***	0 or (n) = 64-bit count ***	MUST be 0	0 or (n) = 64-bit count ***	0 or (n) = 64-bit count ***	0 or (n) = 64-bit count ***
ifOutDiscards:"The number of outbound packets which were chosen to be discarded even though no errors had been detected to prevent their being transmitted. One possible reason for discarding such a packet could be to free up buffer space. Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of ifCounterDiscontinuityTime."	(n)	(n)	(n)	MUST be 0	(n)	(n)	MUST be 0	(n)	(n)	(n)
ifOutErrors:"For packet-oriented interfaces, the number of outbound packets that could not be transmitted	(n)	(n)	(n)	MUST be 0	(n)	(n)	MUST be 0	(n)	(n)	(n)

RFC-2670 MIB-Object details for Cable Device using 10 Meg Ethernet	CMTS-Ethernet-10	CMTS-MAC	CMTS-Downstream	CMTS-Upstream	CM-Ethernet-10	CM-MAC	CM-Downstream	CM-Upstream	CM-USB	CM-CPE Other Type
because of errors. For character-oriented or fixed-length interfaces, the number of outbound transmission units that could not be transmitted because of errors. Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of ifCounterDiscontinuityTime."										
ifPromiscuousMode:"This object has a value of false(2) if this interface only accepts packets/frames that are addressed to this station. This object has a value of true(1) when the station accepts all packets/frames transmitted on the media. The value true(1) is only legal on certain types of media. If legal, setting this object to a value of true(1) may require the interface to be reset before becoming effective. The value of ifPromiscuousMode does not affect the reception of broadcast and multicast packets/frames by the interface."	true(1) false(2)	true(1) false(2)	false(2)	true(1) false(2)	true(1) false(2)	true(1) false(2)	true(1) false(2)	false(2)	true(1) false(2)	true(1) false(2)

RFC-2670 MIB-Object details for Cable Device using 100 Meg Ethernet **(effected MIB-Objects only; all others same as above table)**	**CMTS-Ethernet-100**	**CMTS-MAC**	**CMTS-Downstream**	**CMTS-Upstream**	**CM-Ethernet-100**	**CM-MAC**	**CM-Downstream**	**CM-Upstream**	**CM-USB**	**CM-CPE Other Type**
ifSpeed:"An estimate of the interface's current bandwidth in bits per second. [For RF Downstream; This is the symbol rate multiplied with the number of bits per symbol. For RF Upstream; This is the raw bandwidth in bits per second of this interface, regarding the highest speed modulation profile that is defined. This is the symbol rate multiplied with the number of bits per symbol for this modulation profile. For MAC Layer; Return zero.] For interfaces which do not vary in bandwidth or for those where no accurate estimation can be made, this object should contain the nominal bandwidth. If the bandwidth of the interface is greater than the maximum value reportable by this object then this object should report its maximum value (4,294,967,295) and ifHighSpeed must be used to report the interace's speed. For a sub-layer which has no concept of bandwidth, this object should be zero."	100,000,000	0	~64-QAM=30,341,646, ~256-QUAM=42,884,296	(n)	100,000,000	0	~64-QAM=30,341,646, ~256-QUAM=42,884,296	(n)	12,500,000	speed

RFC-2670 MIB-Object details for Cable Device using 100 Meg Ethernet **(effected MIB-Objects only; all others same as above table)**	**CMTS-Ethernet-100**	**CMTS-MAC**	**CMTS-Downstream**	**CMTS-Upstream**	**CM-Ethernet-100**	**CM-MAC**	**CM-Downstream**	**CM-Upstream**	**CM-USB**	**CM-CPE Other Type**
ifHighSpeed:"An estimate of the interface's current bandwidth in units of 1,000,000 bits per second. If this object reports a value of `n' then the speed of the interface is somewhere in the range of `n-500,000' to `n+499,999'. [For RF Downstream; This is the symbol rate multiplied with the number of bits per symbol. For RF Upstream; This is the raw bandwidth in bits per second of this interface, regarding the highest speed modulation profile that is defined. This is the symbol rate multiplied with the number of bits per symbol for this modulation profile. For MAC Layer; Return zero.] For interfaces which do not vary in bandwidth or for those where no accurate estimation can be made, this object should contain the nominal bandwidth. For a sub-layer which has no concept of bandwidth, this object should be zero."	100	0	~64-QAM=30, ~256-QUAM=42	(n)	100	0	~64-QAM=30, ~256-QUAM=42	(n)	12	speed
ifInOctets:"The total number of octets received on the interface, including framing characters. [For RF Upstream/Downstream (where not zero*); This includes MAC packets as well as data packets, and includes the length of the MAC header. For MAC Layer; The total number of data octets received on this interface, targeted for upper protocol layers.] Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of ifCounterDiscontinuityTime."	(n) = low 32-bits of the 64-bit count	(n)	MUST be 0	(n)	(n) = low 32-bits of the 64-bit count	(n) = low 32-bits of the 64-bit count	(n) = low 32-bits of the 64-bit count	MUST be 0	(n)	(n)
IfHCInOctets: (usage**) "The total number of octets received on the interface, including framing characters. [For RF Upstream/Downstream (where not	(n) = 64-bit count	0 or (n) = 64-bit count ***	MUST be 0	0 or (n) = 64-bit count ***	(n) = 64-bit count	(n) = 64-bit count	(n) = 64-bit count	MUST be 0	0 or (n) = 64-bit count ***	0 or (n) = 64-bit count ***

RFC-2670 MIB-Object details for Cable Device using 100 Meg Ethernet (effected MIB-Objects only; all others same as above table)	CMTS-Ethernet-100	CMTS-MAC	CMTS-Downstream	CMTS-Upstream	CM-Ethernet-100	CM-MAC	CM-Downstream	CM-Upstream	CM-USB	CM-CPE Other Type
zero*); This includes MAC packets as well as data packets, and includes the length of the MAC header. For MAC Layer; The total number of data octets received on this interface, targeted for upper protocol layers.] This object is a 64-bit version of ifInOctets. Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of ifCounterDiscontinuityTime."		***		***					***	***
ifOutOctets:"The total number of octets transmitted out of the interface, including framing characters. [For RF Upstream/ Downstream (where not zero*); This includes MAC packets as well as data packets, and includes the length of the MAC header. For MAC Layer; The total number of octets, received from upper protocol layers and transmitted on this interface.] Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of ifCounterDiscontinuityTime."	(n) = low 32-bits of the 64-bit count	(n) = low 32-bits of the 64-bit count	(n) = low 32-bits of the 64-bit count	MUST be 0	(n) = low 32-bits of the 64-bit count	(n)	MUST be 0	(n)	(n)	(n)
ifHCOutOctets: (usage**) "The total number of octets transmitted out of the interface, including framing characters. [For RF Upstream/ Downstream (where not zero*); This includes MAC packets as well as data packets, and includes the length of the MAC header. For MAC Layer; The total number of octets, received from upper protocol layers and transmitted on this interface.] This object is a 64-bit version of ifOutOctets. Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of	(n) = 64-bit count	(n) = 64-bit count	(n) = 64-bit count	MUST be 0	(n) = 64-bit count	0 or (n) = 64-bit count ***	MUST be 0	0 or (n) = 64-bit count ***	0 or (n) = 64-bit count ***	0 or (n) = 64-bit count ***

RFC-2670 MIB-Object details for Cable Device using 100 Meg Ethernet (effected MIB-Objects only; all others same as above table)	CMTS-Ethernet-100	CMTS-MAC	CMTS-Downstream	CMTS-Upstream	CM-Ethernet-100	CM-MAC	CM-Downstream	CM-Upstream	CM-USB	CM-CPE Other Type
ifCounterDiscontinuityTime."										

APPENDIX B. Business Process Scenarios For Subscriber Account Management

In order to develop the DOCS-OSS Subscriber Account Management Specification, it is necessary to consider high-level business processes common to cable operators and the associated operational scenarios. The following definitions represent a generic view of key processes involved. It is understood that business process terminology vary among different cable operators, distinguished by unique operating environments and target market segments

For the purpose of this document, Subscriber Account Management 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 subscriber customers.

B.1. The Old Service Model -- "One Class Only" & "Best Effort" Service

The Internet is an egalitarian cyber society in its pure technical form where all Internet Protocol (IP) packets are treated as equals. Given all IP packets have equal right of way over the Internet, it is a "one class fits all", "first come, first serve" type of service level arrangement. The response time and quality of delivery service is promised to be on a "best effort" basis only.

Unfortunately, while all IP packets are theoretically equal, certain classes of IP packets must be processed differently. When transmitting data packets, traffic congestion causes no fatal problems except unpredictable delays and frustrations. However, in a convergent IP world where data packets are mixed with those associated with voice and streaming video, such "one class" service level and "best effort only" quality is not workable.

B.2. The Old Billing Model -- *"Flat Rate" Access*

As high speed data over cable service deployment moves to the next stage, serious considerations must be made by all cable operators to abandon old business practices, most notably "flat rate" fee structure. No service provider can hope to stay in business long by continuing to offer a single, "flat rate" access service to all subscribers, regardless of actual usage.

Imagine your utility bills were the same month after month, whether you used very little water or electricity every day, or if you ran your water and your air conditioning at full blast 24 hours a day. You are entitled, just like everyone else, to consume as much or as little as you wished, anytime you wanted it. Chances are you would not accept such a service agreement. Not only because it is not a fair arrangement, but also because such wasteful consumption would put pressure on the finite supply of water and electricity that most of your normal demands for usage would likely go unfulfilled.

B.3. A Successful New Business Paradigm

The new paradigm for delivering IP-based services over cable networks is forcing all cable operators to adopt a new business paradigm. The retention of customers will require that an operator offer different class of service options and associated access rates with guaranteed provisioning and delivery of subscribed services. "Back Office" usage-based accounting and subscriber billing will become an important competitive differentiation in the emergence of high-speed data over cable services.

B.3.1 Integrating "Front End" Processes Seamlessly with "Back Office" Functions

A long-standing business axiom states that accountability exists only with the right measurements and that business prospers only with the proper management information. An effective subscriber account management system for data over cable services should meet three (3) major requirements:

Automatic & Dynamic Subscriber Provisioning

The 1st requirement is to integrate service subscription orders and changes automatically and dynamically, with the various processes that invoke the provisioning and delivering of subscribed and/or "on demand" services;

Guaranteed Class & Quality of Services

The 2nd requirement is to offer different class of services with varying rates and guarantee the quality of service level associated with each service class;

Data Collection, Warehousing & Usage Billing

The 3rd requirement is to capture a subscriber's actual usage, calculating the bill based on the rate associated with the customer's subscribed service levels.

B.3.2 Designing Class of Services

While designing different class of service offerings, a cable operator might consider the following framework:

Class of Service by Account Type – Business vs. Residential Accounts

Class of Service by Guaranteed Service Levels

Class of Service by Time of Day and/or Day of Week

"On Demand" Service by Special Order

The following is a plausible sample of class of services:

- *"Best Effort" Service Without Minimum Guarantee*

  This class of "Best Effort Only" service is the normal practice of today where subscribers of this class of service are allocated only excess channel bandwidth available at the time while each subscriber's access is capped at a maximum bandwidth (for example at 512 kilobit per second).

- *Platinum Service for Business and High-Access Residential Accounts*

  Business accounts subscribing to this service are guaranteed a minimum data rate of downstream bandwidth – 512 kilobit per second – and if excess bandwidth is available, they are allowed to burst to 10 megabit per second.

- *Gold Service for Business Accounts*

  This class of service guarantees subscribers a 256 kilobit per second downstream data rate during business hours (for example from 8 a.m. to 6 p.m.) and 128 kilobit per second at other times. If excess bandwidth is available at any time, data is allowed to burst to 5 megabit per second.

- *Gold Service for Residential Accounts*

  Residential subscribers of this service are guaranteed 128 kilobit per second downstream bandwidth during business hours and 256 kilobit per second at other times (for example from 6 p.m. to 8 a.m.), and a maximum data burst rate of 5 megabit per second with available excess bandwidth.

- *Silver Service for Business Accounts*

  Business accounts subscribing to this service are guaranteed 128 kilobit per second downstream data rate during business hours and 64 kilobit per second during other times, and a maximum burst rate of 1 megabit per second.

- *Silver Service for Residential Accounts*

  Subscribers are guaranteed 64 kilobit per second downstream bandwidth during business hours and 128 kilobit per second at other times, with a maximum burst rate of 1 megabit per second.

- *"On Demand" Service by Special Order*

  This class of "on demand" service allows a subscriber to request additional bandwidth available for a specific period of time. For example, a subscriber can go to operator's web site and requests for increased guaranteed bandwidth service levels from his registered subscribed class of service from the normal 256 kilobit per second to 1 megabit per second from 2 p.m. to 4 p.m. the following day only, after which his service levels returns to the original subscribed class. The provisioning server will check the bandwidth commitment and utilization history to decide whether such "on demand" service is granted.

B.3.3 Usage-Based Billing

A complete billing solution involves the following processes:

- Design different usage-based billing options
- Capture and manage subscriber account and service subscription information
- Estimate future usage based on past history
- Collect billable event data
- Generate and rate billing records
- Calculate, prepare and deliver bill
- Process and manage bill payment information and records
- Handle customer account inquires
- Manage debt and fraud

This Specification focuses only on various business scenarios on bandwidth-centric usage-based billing options.

B.3.4 Designing Usage-Based Billing Models

In support of the offering of different class of services is a new set of billing processes, which are based on the accounting of actual usage of subscribed service by each subscriber calculated by the associated fee structures.

There are several alternatives to implementing usage-based billing. The following offers a few examples:

- *Billing Based on an Average Bandwidth Usage.*

  The average bandwidth usage is defined as the total bytes transmitted divided by the billing period.

- *Billing Based on Peak Bandwidth Usage.*

  The peak bandwidth usage is the highest bandwidth usage sample during the entire billing period. Each usage sample is defined as the average bandwidth usage over a data collection period (typically 10 minutes).

Since it is usually the peak usage pattern that creates the highest possibility of access problems for the cable operator, therefore it is reasonable to charge for such usage. One scheme of peak usage billing is called "95 percentile billing". The process is as follows -- at the end of each billing period, the billing software examines the usage records of each subscriber and it "throws away" the top five percent of usage records of that period, then charge the subscriber on the next highest bandwidth usage.

- *"Flat Monthly Fee"Plus Usage Billing Based on the Class of Service Subscribed.*

  Any usage beyond the minimum guaranteed bandwidth for that particular subscriber service class is subject to an extra charge based on the number of bytes transmitted.

- *Billing for "On Demand" Service*

  This special billing process is to support the "On Demand" Service offering described above.

Appendix C. IPDR Standards Submission for Cable Data Systems Subscriber Usage Billing Records

C.1 Service Definition

Cable Data Systems consist of Cable Modem Termination Systems (CMTS) (located at a Multiple Service Operator's (MSO) head-end office) that provide broadband Internet access to subscribers connected via Cable Modems (CMs) through the cable plant. These Cable Data Systems comply with the Data Over Cable Service Interface Specifications (DOCSIS) sponsored by Cable Television Laboratories, Inc. The IPDR format for Cable Data Systems Subscriber Usage Billing Records specified herein support the DOCSIS 1.1 Operations Support System Interface specification (OSSI). The DOCSIS 1.1 OSSI requires the CMTS to provide usage-billing records for all bandwidth consumed by the subscribers connected to it via their Cable Modems when polled by the MSO's billing or mediation system.

C.1.1 Service Requirements

1. Cable Data Service is "always on". Thus, from the CMTS perspective, there are no subscriber logon events to track, but rather, in a manner similar to electric power utilities, there are only data traffic flows to meter and police.
2. A Cable Data Subscriber is uniquely identified by their Cable Modem MAC address (i.e. Ethernet address). Note that a CM is usually assigned a dynamic IP address via DHCP, so the IP address of a subscriber changes over time. Since the CM MAC address is constant, it must be used to identify the subscriber's usage billing records. All Internet traffic generated by the subscriber's Customer Premises Equipment (CPE) is bridged by the CM to and from the CMTS. The subscriber's packet and byte (octet) traffic counts are recorded by the CMTS in counters associated with the CM MAC address. Note that the current IP addresses of the CM and all the CPE in use during the collection interval are recorded for auditing purposes.
3. Cable Data Service is metered and enforced against a Service Level Agreement (SLA) that specifies the Quality of Service (QoS) that an MSO provides to a subscriber. An MSO typically has several Service Packages to offer to their subscribers, such as "Gold", "Silver", or "Bronze". Each of the Service Packages implements a specific SLA and is available for a specific price. A Service Package is implemented by a set of Service Flows that are known to the billing system by their Service Flow IDs (SFIDs) and Service Class Names (SCNs). Service Flows are the unit of billing data collection for a Cable Data Subscriber. In addition, since a subscriber may change their Service Package over time, it is very likely that a given subscriber will have several IPDRs, one for each Service Flow they have used during the collection interval.
4. Bandwidth in a Cable Data System is measured separately in both the downstream and upstream directions (relative to the CMTS). Each Service Flow is unidirectional and is associated with packet traffic of a specific type (e.g. TCP or UDP). Since most SLAs provide for asymmetric bandwidth guarantees, it is necessary to separate the downstream and upstream traffic flows in the billing usage records. Bandwidth used is measured in both packets and octets.
5. The bandwidth guarantee component of the SLA is enforced and metered by the CMTS with the assistance of the CM. However, the CM is not considered a trusted device because of its location on the Customer's Premises, so the CMTS is expected to provide all of the usage billing information for each subscriber connected to it.
6. Since an SLA may require the CMTS to enforce bandwidth limits by dropping or delaying packets that exceed the maximum throughput bandwidth for a Service Flow, the SLA dropped packets counters and delayed packets counters are also included in the usage records for each Service Flow. These counters are not used to compute billable subscriber usage but rather are available to the billing and customer care systems to enable "up-selling" to subscribers who try to exceed their subscribed service level. Thus, subscribers whose usage patterns indicate a large number of

dropped octets are probably candidates for an upgrade to a higher SLA that supports their true application bandwidth demands which, in turn, generates more revenue for the MSO.

7. The packet and octet values in the usage billing records are based on absolute 64-bit counters maintained in the CMTS. These counters may be reset when the CMTS system resets, therefore the CMTS System Up Time (sysUpTime) is included in the IPDRdoc so that the billing or mediation system can correlate counters that appear to regress.

C.1.2 Service Usage Attribute List

C.1.2.1 Service Session (SS)

The Service Session records the usage for a Service Consumer (i.e. Subscriber) associated with a specific Service Flow as seen at this collection interval. The standard SS attribute name ***service*** identifies the Service Class Name (SCN) of the Service Flow associated with this bandwidth usage. Note that the SFID for the Service Flow is recorded as a Usage Event (UE) attribute (see section C.1.2.2 below). See Table 1 below for a summary of all service usage attribute value names.

C.1.2.1.1 Service Consumer (SC)

The Service Consumer (Subscriber) is identified by their Cable Modem MAC Address and their current Cable Modem IP address (as assigned by DHCP). The standard usage attribute value names ***subscriberdId*** and ***ipAddress*** are used to record this information. Additionally, each CPE IP address that was in use during the collection interval is also recorded. A new usage attribute value name ***cpeIpAddress*** is used to record these addreses. Note that since many IPDRs in this IPDRdoc are for the same Subscriber, a single SC element is created for each Subscriber that is then referenced within each associated IPDR by an SCRef element. Each Subscriber's SC element is identified by a unique sequential reference value.

C.1.2.1.2 Service Element (SE)

The CMTS is the single Service Element that records all of the subscriber usage in this IPDRdoc. The CMTS is identified by its IP address and its DNS host name. The standard usage attribute value names ***ipAddress*** and ***hostName*** are used to record this information. In addition, the current value of the CMTS System Up Time is included so the billing or mediation system can determine if the CMTS has been reset since the last record collection cycle. A new usage attribute value name ***sysUpTime*** is used to record this information. The format of sysUpTime is a 32-bit integer counting the number of hundredths of a second since the management interface of the CMTS was initialized. Note that since all IPDRs in this IPDRdoc are created by the same CMTS, the SE element in each IPDR is included by reference (SERef) to the single CMTS SE entry at the beginning of the document. The SE reference id is usually the host name of the CMTS.

C.1.2.2 Usage Event (UE)

The Usage Event records the absolute value of the packet and octet counters associated with a single active Service Flow for a given Subscriber (i.e. CM) as seen during this collection interval. The UE ***type*** keyword is ***Interim*** if the Service Flow is currently active or ***Stop*** if the Service Flow has been deleted during this collection interval. Note that the IPDR ***time*** value for an Interim record is always the same as the IPDRDoc ***startTime*** value, but a Stop record always has a ***time*** value earlier than the IPDRDoc.

A single UE represents the absolute bandwidth consumed by the Subscriber since the Service Flow was started. Bandwidth consumed during the interval must be computed by the billing system based on

counters from adjacent collection intervals. The CMTS maintains the absolute values in 64-bit counters which are reported as usage attribute values in the IPDR formatted in ASCII decimal representation as described below. The internal 32-bit Service Flow ID is recorded as the new usage attribute value name ***SFID*** to faciltate correlation of counter sets for the same Service Flow in sequential IPDRDoc files.

Note well in the discussion that follows that ***downstream*** and ***upstream*** are relative to the CMTS while ***receive*** and ***send*** are relative to the CM. A Usage Event is always seen from the Subscriber's (i.e. CM's) frame of reference, therefore receive and send are the directional modifiers of the usage attribute value names in an IPDR. In addition, since a Service Flow is unidirectional there should be either receive-counts or send-counts for that Service Flow, but not both. Note also that the directional modifiers of the usage attribute value names are the only true indicators of the Service Flow direction for the billing system as the SCN is chosen arbitrarily by the MSO and cannot be relied on to encode Service Flow direction in its name.

For an **upstream Service Flow,** packet traffic is recorded as bandwidth sent from the CM to the CMTS. The bandwidth-consumed counters are in both packets and octets so the standard usage attribute value names ***sendPkts*** and ***sendOctets*** are used to record this information.

For a **downstream Service Flow,** packet traffic is recorded as bandwidth received by the CM from the CMTS. The bandwidth-consumed counters are in both packets and octets so the standard usage attribute value names ***recvPkts*** and ***recvOctets*** are used to record this information. In addition, for downstream Service Flows only, the CMTS records the number of received and sent packets dropped and delayed due to the subscriber exceeding the maximum SLA bandwidth limit associated with a Service Flow. Two new usage attribute value names are needed to record this information: ***recvSLADropPkts*** and ***recvSLADelayPkts***.

Table 2 Service Usage Attribute Value Names

Element	Attribute or Usage Attribute Value Name	Type	Units/Values	Remarks
SS	**service**	String	Examples: GoldTCPDown, BronzeUDPUp	Service Class Name (SCN) of the Service Flow
SC	**subscriberId**	String	hh-hh-hh-hh-hh-hh	Cable Modem MAC address in dash delimited hex notation
	ipAddress	String	nnn.nnn.nnn.nnn	CM's current IP Address. Canonical IP address in period delimited decimal notation
	cpeIpAddress*	String	nnn.nnn.nnn.nnn	Current IP address of a CPE using this CM. One per CPE active during the collection interval.
SE	**ipAddress**	String	nnn.nnn.nnn.nnn	CMTS's IP Address. Canonical IP address in period delimited decimal notation
	hostName	String	Example: cmts-01.mso.com	CMTS's fully qualified domain name
	sysUpTime*	Unsigned32	nnnnnnnnn	32-bit count of hundredths of a second since system initialization in decimal notation
UE	**type**	keyword	Interim | Stop	Interim identifies running SFs. Stop identifies deleted SFs.
	SFID*	Unsigned32	nnnnnnnnn	32-bit Service Flow ID of the SF in decimal notation
For Downstream Service Flows only:				
	recvOctets	double	64-bit counter in	Downstream Octets

* New usage attribute value names that need to be defined in the standard IPDR dictionary

			decimal notation	
	recvPkts	"	"	Downstream packets
	recvSLADropPkts*	"	"	Downstream dropped packets exceeding SLA
	recvSLADelayPkts*	"	"	Downstream delayed packets exceeding SLA
For Upstream Service Flows only:				
	sendOctets	double	64-bit counter in decimal notation	Upstream Octets
	sendPkts	"	"	Upstream packets

C.2 Example IPDR XML Subscriber Usage Billing Records

The example Subscriber Usage Billing File can be viewed easily via a standard web browser (such as Microsoft Internet Explorer 5.0) if the IPDR standard Data Type Definition (DTD) file ipdr_1.0.dtd is placed in the same directory as the billing file.

C.2.1 ipdr_1.0.dtd -- Standard IPDRdoc Data Type Definition (DTD) File

```
<!--  The IPDRDoc element is the top-level container of a set of
      IPDR's.  The document will also define the entity which
      recorded these IPDR's via the IPDRRecorder element.
  -->
<!ELEMENT IPDRDoc  (IPDRRec , IPDRRecList? ,  (IPDR | IPDRTable )+ ,
IPDRDoc.End? )>
<!ATTLIST IPDRDoc  seqNum    CDATA     #IMPLIED
                   version   CDATA     #IMPLIED
                   startTime CDATA     #IMPLIED
                   info      CDATA     #IMPLIED
                   a-dtype   NMTOKENS  'seqNum    int
                                        startTime dateTime.tz' >
<!--  The IPDRDoc.End element optionally marks the end of the IPDR block.
      It may contain some check information like a count of IPDR's.
  -->
<!ELEMENT IPDRDoc.End EMPTY>
<!ATTLIST IPDRDoc.End  count   CDATA     #IMPLIED
                       endTime CDATA     #IMPLIED
                       a-dtype NMTOKENS  'count   int
                                          endTime dateTime.tz' >
<!-- The IPDRRec element describes the entity that is responsible for
```

```
     creating (recording) the IPDRDocument.
  -->
<!ELEMENT IPDRRec EMPTY>
<!ATTLIST IPDRRec  id        ID        #IMPLIED
                   startTime CDATA     #IMPLIED
                   info      CDATA     #IMPLIED
                   a-dtype   NMTOKENS  'startTime dateTime.tz' >
<!-- The IPDRRecRef element may be used to associate common references
     to the same IPDRRec element without repeating its other attributes.
  -->
<!ELEMENT IPDRRecRef EMPTY>
<!ATTLIST IPDRRecRef  ref IDREF  #REQUIRED >
<!-- The IPDRRecList identifies contributing IPDR recording entities
     which were used in the construction of the current IPDR Document.
     A typical example use would be for an aggregator of IPDR documents
     to identify the set of initial recorders presenting IPDRs
  -->
<!ELEMENT IPDRRecList  (IPDRRec+ )>

<!-- An IPDR describes an event between a Service Consumer (SC) and
     a Service Element (SE).  The SC and SE elements are contained
     beneath an entity called the ServiceSession (SS).  Details of
     the event is contained in the Usage Event (UE) element.  All IPDR's
     have a time indicating when the event occurred.
  -->
<!ELEMENT IPDR  ( (IPDRRec | IPDRRecRef )? ,  (SS | SSRef ) , UE , BaseIPDR?
)>
<!ATTLIST IPDR  id      ID        #IMPLIED
                time    CDATA     #REQUIRED
                seqNum  CDATA     #IMPLIED
                a-dtype NMTOKENS  'time   dateTime.tz
                                   seqNum int' >
<!-- The Service Session (SS) element groups the Service Consumer
     and Service Element information.  This grouping allows
     an SC/SE pair to be associated with other IPDR's via
     a single reference (the SSRef).
  -->
<!ELEMENT SS  ( (SC | SCRef ) ,  (SE | SERef ) )>
<!ATTLIST SS  id      ID     #IMPLIED
              service CDATA  #IMPLIED >
<!-- The SSRef element may be used to associate common references
     to the same pairing of a Service Consumer and a Service Element.
```

```
  -->
<!ELEMENT SSRef EMPTY>
<!ATTLIST SSRef  ref IDREF  #REQUIRED >
<!-- An IPDR's ServiceConsumer, ServiceElement and UsageEvent
     sections are used to partition a set of attributes <v> elements
     into their appropriate categories.  The SC and SE components
     also have Reference analogs to associate common groups of
     attributes, and reduce the number of elements in a document.

     All Service events indicate what type of metrics they carry.
     By default a 'Start-Stop' type indicates that it is a
     complete measurement and does not rely on other
     records to form a complete picture of an activity.
  -->
<!ELEMENT SC  (v* )>
<!ATTLIST SC  id ID  #IMPLIED >
<!ELEMENT SE  (v* )>
<!ATTLIST SE  id ID  #IMPLIED >
<!ELEMENT UE  (v* )>
<!ATTLIST UE  type  (Start | Stop | Start-Stop | Interim )  'Start-Stop'
              name CDATA  #IMPLIED >
<!-- The SERef and SCRef elements may be used to associate common
references to the Session Element or Session Consumer.
  -->
<!ELEMENT SERef EMPTY>
<!ATTLIST SERef  ref IDREF  #REQUIRED >
<!ELEMENT SCRef EMPTY>
<!ATTLIST SCRef  ref IDREF  #REQUIRED >
<!-- The BaseIPDR element allows reference to be made to IPDRs which
     contributed to the construction of the current IPDR element.
  -->
<!ELEMENT BaseIPDR EMPTY>
<!ATTLIST BaseIPDR  refs IDREFS  #REQUIRED >
<!-- The attribute value <v> element is the main extensibility mechanism
     of the IPDR record.  The IPDR group will define a set of named
     attributes that follow a hierarchical naming convention.
     All IPDR derived attribute names will reside under the
     org.ipdr.* hierarchy.  Based on their position in an element,
     the leading org.ipdr name of the attribute may be omitted
     for compactness.  Third party attributes must be fully
     qualified.
  -->
<!ELEMENT v  (#PCDATA )>
<!ATTLIST v  name CDATA  #IMPLIED >
```

```
<!-- An IPDRTable element is used to more compactly represent a set
     of homogenous IPDR's.  Homogenous in the sense that each IPDR
     in a table contains the same set of describing attribute
     elements <v> elements.  Use of IPDRTable is optional.  IPDR
     elements may directly live under the IPDRDoc if they
     do not share a common structure or verbosity is desired
     over compactness.
  -->
<!ELEMENT IPDRTable  ( (IPDRRec | IPDRRecRef )? , IPDRSchema ,  (IPDR )+ ,
IPDRTable.End? )>
<!ATTLIST IPDRTable  startTime CDATA     #IMPLIED
                     a-dtype   NMTOKENS  'startTime dateTime.tz' >
<!--  The IPDRTable.End element optionally marks the end of a set of IPDR's
      in an IPDRTable element.  It may contain some check information
like a count of IPDR's.
  -->
<!ELEMENT IPDRTable.End EMPTY>
<!ATTLIST IPDRTable.End  count   CDATA     #REQUIRED
                         endTime CDATA     #IMPLIED
                         a-dtype NMTOKENS  'count   int
                                            endTime dateTime.tz' >
<!-- The IPDRSchema element within an IPDRTable defines the attribute
     elements common to all the IPDR's in the given table
  -->
<!ELEMENT IPDRSchema  (SCSchema , SESchema , UESchema )>

<!-- The following schema components define the sets of attributes
     for the ServiceConsumer (SC), ServiceElement (SE) and
     Usage Event (UE) components of the IPDR respectively.
  -->
<!ELEMENT SCSchema  (AttrDesc+ )>

<!ELEMENT SESchema  (AttrDesc+ )>

<!ELEMENT UESchema  (AttrDesc+ )>

<!-- An AttrDesc element describes an attribute that will appear in
     in a particular position in a subsequent set of IPDR's contained
     in a table.  One can think of this as being analogous to the
     table header <TH> elements in HTML.
  -->
<!ELEMENT AttrDesc  (#PCDATA )>
```

C.2.2 Example IPDRDoc XML File Containing Subscriber Usage IPDRs

```
<?xml version="1.0"?>
<!DOCTYPE IPDRDoc SYSTEM "ipdr_1.0.dtd">
<IPDRDoc version="1.0" startTime="2000-06-21T21:33:17Z">
   <IPDRRec info="CMTS-01.mso.com"/>
   <IPDR time="2000-06-21T21:33:17Z">
         <SS>
                        <SC/>
                        <SE id="CMTS-01">
                              <v name="ipAddress">192.168.100.30</v>
                              <v name="hostName">CMTS-01.mso.com</v>
                              <v name="sysUpTime">12345678</v>
                        </SE>
         </SS>
         <UE/>
   </IPDR>
   <IPDR time="2000-06-21T21:20:11Z">
         <SS service="BronzeTCPDown">
                        <SC id="SC00001">
                              <v name="subscriberId">11-11-11-11-11-11</v>
                              <v name="ipAddress">192.168.100.111</v>
                              <v name="cpeIpAddress">192.168.200.111</v>
                              <v name="cpeIpAddress">192.168.200.112</v>
                              <v name="cpeIpAddress">192.168.222.113</v>
                        </SC>
                        <SERef ref="CMTS-01"/>
         </SS>
         <UE type="Stop">
                        <v name="SFID">33333333</v>
                        <v name="recvOctets">34567890</v>
                        <v name="recvPkts">567890</v>
                        <v name="recvSLADropPkts">7890123</v>
                        <v name="recvSLADelayPkts">7890</v>
         </UE>
   </IPDR>
   <IPDR time="2000-06-21T21:20:12Z">
         <SS service="BronzeTCPUp">
                        <SCRef ref="SC00001"/>
                        <SERef ref="CMTS-01"/>
         </SS>
```

```
      <UE type="Stop">
                  <v name="SFID">44444444</v>
                  <v name="sendOctets">34567</v>
                  <v name="sendPkts">4567</v>
      </UE>
</IPDR>
<IPDR time="2000-06-21T21:20:13Z">
      <SS service="BronzeUDPDown">
                  <SCRef ref="SC00001"/>
                  <SERef ref="CMTS-01"/>
      </SS>
      <UE type="Stop">
                  <v name="SFID">55555555</v>
                  <v name="recvOctets">345678</v>
                  <v name="recvPkts">5678</v>
                  <v name="recvSLADropPkts">78677</v>
                  <v name="recvSLADelayPkts">890</v>
      </UE>
</IPDR>
<IPDR time="2000-06-21T21:20:14Z">
      <SS service="BronzeUDPUp">
                  <SCRef ref="SC00001"/>
                  <SERef ref="CMTS-01"/>
      </SS>
      <UE type="Stop">
                  <v name="SFID">66666666</v>
                  <v name="sendOctets">345</v>
                  <v name="sendPkts">45</v>
      </UE>
</IPDR>
<IPDR time="2000-06-21T21:33:17Z">
      <SS service="PrimaryDown">
                  <SCRef ref="SC00001"/>
                  <SERef ref="CMTS-01"/>
      </SS>
      <UE type="Interim">
                  <v name="SFID">1111111</v>
                  <v name="recvOctets">4567</v>
                  <v name="recvPkts">34</v>
                  <v name="recvSLADropPkts">0</v>
                  <v name="recvSLADelayPkts">0</v>
      </UE>
</IPDR>
```

```
<IPDR time="2000-06-21T21:33:17Z">
      <SS service="PrimaryUp">
                  <SCRef ref="SC00001"/>
                  <SERef ref="CMTS-01"/>
      </SS>
      <UE type="Interim">
                  <v name="SFID">22222222</v>
                  <v name="sendOctets">12345</v>
                  <v name="sendPkts">345</v>
      </UE>
</IPDR>
<IPDR time="2000-06-21T21:33:17Z">
      <SS service="GoldTCPDown">
                  <SCRef ref="SC00001"/>
                  <SERef ref="CMTS-01"/>
      </SS>
      <UE type="Interim">
                  <v name="SFID">77777777</v>
                  <v name="recvOctets">1234567890</v>
                  <v name="recvPkts">34567890</v>
                  <v name="recvSLADropPkts">67890</v>
                  <v name="recvSLADelayPkts">9090</v>
      </UE>
</IPDR>
<IPDR time="2000-06-21T21:33:17Z">
      <SS service="GoldTCPUp">
                  <SCRef ref="SC00001"/>
                  <SERef ref="CMTS-01"/>
      </SS>
      <UE type="Interim">
                  <v name="SFID">88888888</v>
                  <v name="sendOctets">1234567</v>
                  <v name="sendPkts">34567</v>
      </UE>
</IPDR>
<IPDR time="2000-06-21T21:33:17Z">
      <SS service="GoldUDPDown">
                  <SCRef ref="SC00001"/>
                  <SERef ref="CMTS-01"/>
      </SS>
      <UE type="Interim">
```

```
                        <v name="SFID">99999999</v>
                        <v name="recvOctets">12345678</v>
                        <v name="recvPkts">345678</v>
                        <v name="recvSLADropPkts">678</v>
                        <v name="recvSLADelayPkts">789</v>
      </UE>
</IPDR>
<IPDR time="2000-06-21T21:33:17Z">
      <SS service="GoldUDPUp">
                        <SCRef ref="SC00001"/>
                        <SERef ref="CMTS-01"/>
      </SS>
      <UE type="Interim">
                        <v name="SFID">10101010</v>
                        <v name="sendOctets">12345</v>
                        <v name="sendPkts">345</v>
      </UE>
</IPDR>
<IPDR time="2000-06-21T21:25:01Z">
      <SS service="VoiceDown">
                        <SC id="SC00002">
                              <v name="subscriberId">22-22-22-22-22-22</v>
                              <v name="ipAddress">192.168.100.222</v>
                              <v name="cpeIpAddress">192.168.200.222</v>
                              <v name="cpeIpAddress">192.168.200.223</v>
                        </SC>
                        <SERef ref="CMTS-01"/>
      </SS>
      <UE type="Stop">
                        <v name="SFID">12121212</v>
                        <v name="recvOctets">1234</v>
                        <v name="recvPkts">445</v>
                        <v name="recvSLADropPkts">0</v>
                        <v name="recvSLADelayPkts">0</v>
      </UE>
</IPDR>
<IPDR time="2000-06-21T21:25:02Z">
      <SS service="VoiceUp">
                        <SCRef ref="SC00002"/>
                        <SERef ref="CMTS-01"/>
      </SS>
      <UE type="Stop">
                        <v name="SFID">13131313</v>
```

```
                        <v name="sendOctets">1345</v>
                        <v name="sendPkts">565</v>
      </UE>
</IPDR>
<IPDR time="2000-06-21T21:33:17Z">
      <SS service="PrimaryDown">
                        <SCRef ref="SC00002"/>
                        <SERef ref="CMTS-01"/>
      </SS>
      <UE type="Interim">
                        <v name="SFID">14141414</v>
                        <v name="recvOctets">4567</v>
                        <v name="recvPkts">34</v>
                        <v name="recvSLADropPkts">0</v>
                        <v name="recvSLADelayPkts">0</v>
      </UE>
</IPDR>
<IPDR time="2000-06-21T21:33:17Z">
      <SS service="PrimaryUp">
                        <SCRef ref="SC00002"/>
                        <SERef ref="CMTS-01"/>
      </SS>
      <UE type="Interim">
                        <v name="SFID">15151515</v>
                        <v name="sendOctets">12345</v>
                        <v name="sendPkts">345</v>
      </UE>
</IPDR>
<IPDR time="2000-06-21T21:33:17Z">
      <SS service="SilverTCPDown">
                        <SCRef ref="SC00002"/>
                        <SERef ref="CMTS-01"/>
      </SS>
      <UE type="Interim">
                        <v name="SFID">16161616</v>
                        <v name="recvOctets">1234567890</v>
                        <v name="recvPkts">34567890</v>
                        <v name="recvSLADropPkts">67890</v>
                        <v name="recvSLADelayPkts">7089</v>
      </UE>
</IPDR>
```

```
   <IPDR time="2000-06-21T21:33:17Z">
         <SS service="SilverTCPUp">
                        <SCRef ref="SC00002"/>
                        <SERef ref="CMTS-01"/>
         </SS>
         <UE type="Interim">
                        <v name="SFID">17171717</v>
                        <v name="sendOctets">1234567</v>
                        <v name="sendPkts">34567</v>
         </UE>
   </IPDR>
   <IPDR time="2000-06-21T21:33:17Z">
         <SS service="SilverUDPDown">
                        <SCRef ref="SC00002"/>
                        <SERef ref="CMTS-01"/>
         </SS>
         <UE type="Interim">
                        <v name="SFID">18181818</v>
                        <v name="recvOctets">12345678</v>
                        <v name="recvPkts">345678</v>
                        <v name="recvSLADropPkts">678</v>
                        <v name="recvSLADelayPkts">8907</v>
         </UE>
   </IPDR>
   <IPDR time="2000-06-21T21:33:17Z">
         <SS service="SilverUDPUp">
                        <SCRef ref="SC00002"/>
                        <SERef ref="CMTS-01"/>
         </SS>
         <UE type="Interim">
                        <v name="SFID">19191919</v>
                        <v name="sendOctets">12345</v>
                        <v name="sendPkts">345</v>
         </UE>
   </IPDR>
   <IPDRDoc.End count="19" endTime="2000-06-21T21:33:21Z"/>
</IPDRDoc>
```

Appendix D. SNMPv2c INFORM Request Definition for Subscriber Account Management (SAM)

The INFORM Request definition of account management will be specified in this section by the ECR/ECO/ECN process.

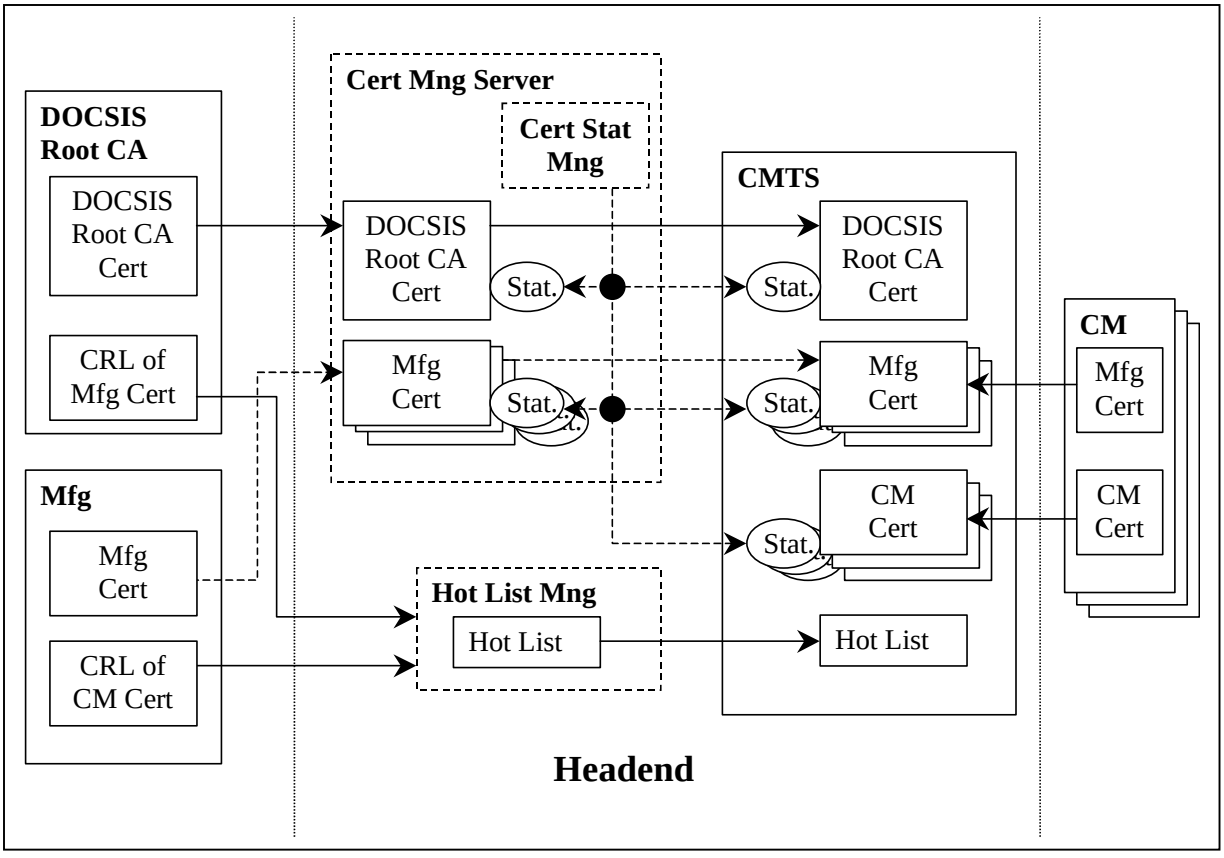
This page intentionally blank

Appendix E. Summary of the CM Authentication and the Code File Authentication

The purpose of this appendix is to provide the overview of the two authentication mechanisms defined by BPI+ specification [SP-BPI+-I03] and also to provide a example of the responsibility assignment for actual operation but not to add any new requirements for the CMTS or the CM. Please refer BPI+ specification [SP-BPI+-I03] regarding the requirement for the CMTS and the CM.

E.1 Authentication of the DOCSIS 1.1 compliant CM

If the CMTS is compliant to the DOCSIS 1.1/BPI+ and a DOCSIS 1.1 compliant CM is provisioned to run BPI+ by the CM configuration file, the CMTS authenticate the CM during the CM initialization by verifying the CM certificate and the manufacturer CA certificate. These certificates are contained in Auth Info message and Auth Request message separately and sent from the CM to the CMTS just after the CM registration. Only the CM with the valid certificates will be authorized by the CMTS and become ready to forward the user traffic. Note that this CM authentication won't be applied if the CMTS and/or the CM is not compliant to BPI+, or the CM is not provisioned to run BPI+.



E.1.1. Responsibility of the DOCSIS Root CA

The DOCSIS Root CA is responsible for the following:

- Store the DOCSIS Root private key in secret.
- Maintain the DOCSIS Root CA certificate.
- Issue the manufacturer CA certificates signed by the DOCSIS Root CA.
- Maintain the CRL of the manufacturer CA.
- Provide the operators with the CRL.

It is not yet decided whether a manufacturer CA certificate signed by the DOCSIS Root CA is provided to the CM manufacturer before applying for the CableLabs' certification process or after achieving the certified status.

E.1.2 Responsibility of the CM manufacturers

The CM manufacturers are responsible for the following:

- Store the manufacturer CA private key in secret,
- Maintain the manufacturer CA certificate. The manufacturer CA certificate is usually signed by the DOCSIS Root CA but can be self-signed until the DOCSIS Root CA issues it based on the CableLabs policy.
- Issue the CM certificates,
- Put the manufacturer CA certificate in the CM's software,
- Put each CM certificate in the CM's permanent, write-once memory.
- Provide the operators with the hot list of the CM certificate. The hot list may be in the CRL format. However, the detail of the format and the way of delivery are TBD.

E.1.3 Responsibility of the operators

The operators are responsible for the following:

- Maintain that the CMTS(s) have an accurate date and time. If a CMTS has a wrong date or time, the invalid certificate may be authenticated or the valid certificate may not be authenticated.
- Put the DOCSIS Root CA certificate in the CMTS during the CMTS provisioning using BPI+ MIB or the CMTS's proprietary function. The operator may have a server to manage this certificate for one or more CMTS(s).
- Put the manufacturer CA certificate(s) in the CMTS during the CMTS provisioning using BPI+ MIB or the CMTS's proprietary function (optional). The operator may have a server to manage this certificate for one or more CMTS(s).

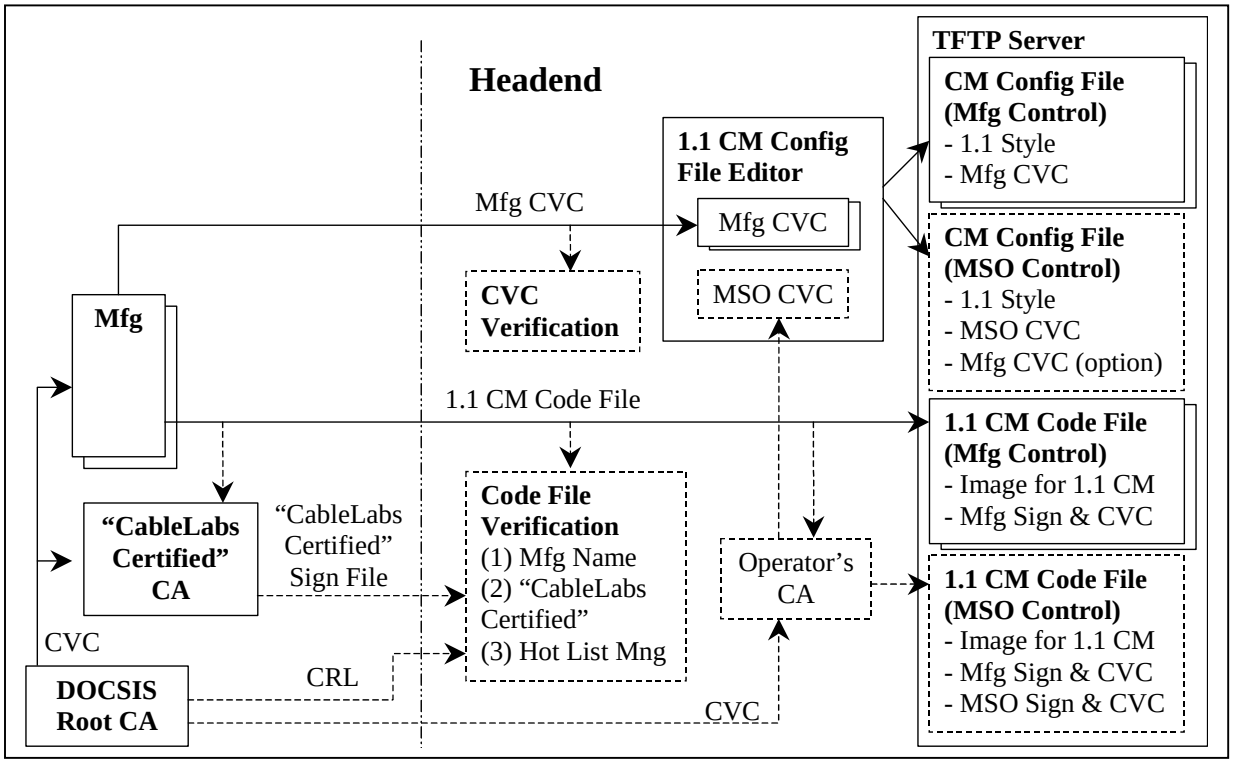
- Maintain the status of the certificates in the CMTS(s) if desired using BPI+ MIB or the CMTS's proprietary function (optional). The operator may have a server to manage all the status of the certificates recorded in one or more CMTS(s).

The operator may have a server to manage the DOCSIS Root CA certificate, manufacturer CA certificate(s) and also the status of the certificates recorded in one or more CMTS(s).

- Maintain the hot list for the CMTS based on the CRLs provided by the DOCSIS Root CA and the CM manufacturers (optional). The operator may have a server to manage the hot list based on the CRLs provided by the DOCSIS Root CA and manufacturer CAs. The CMTS may have a function to automatically download the DOCSIS Root CA certificate and the CRLs via the Internet or other method. The DOCSIS Root CA or CableLabs is likely to put the DOCSIS Root CA on their Web or TFTP server in order to let the operators (or the CMTS on behalf of the operator) download it but this is not yet decided.

E.2 Authentication of the code file for the DOCSIS 1.1 compliant CM

When the DOCSIS 1.1/BPI+ compliant CM downloads the code file from TFTP server, the CM must always authenticate the code file as defined in the appendix D of [SP-BPI+-I03] regardless of whether the CM is provisioned to run BPI+, BPI or none of them by the CM configuration file. The CM installs the new image and restart using it only if the CVC(s) and the signature(s) in the code file are verified. If the authentication fails because of the invalid CVC(s) or signature(s) in the code file, the CM rejects the code file downloaded from the TFTP server and continues to operate using the current code. The CM accepts the order of the software downloading via the CM configuration file or the MIB only if the CM is properly initialised by the CVC(s) in the CM configuration file. In addition to the code file authentication by the CM, the operators may authenticate the code file before they put it on the TFTP sever. The following figure shows the summary of these mechanisms.



E.2.1 Responsibility of the DOCSIS Root CA

The DOCSIS Root CA is responsible for the following:

- Store the DOCSIS Root private key in secret,
- Maintain the DOCSIS Root CA certificate, and
- Issue the code verification certificates (CVCs) for the CM manufacturers, for the operators, and for "CableLabs Certified(TM)".
- May maintain the CRL of the CVCs and provide it with the operators but not yet decided.

E.2.2 Responsibility of the CM manufacturer

The CM manufacturers are responsible for the following:

- Store the manufacturer CVC private key in secret,
- Put the DOCSIS Root CA certificate in the CM's software,
- Maintain the manufacturer CVC. (Current BPI+ specification only allows the CVC signed by the DOCSIS Root CA and does not accept the self-signed CVC.)
- Generate the code file with the manufacturer's CVC and signature, and
- Provide the operators with the code file and the manufacturer CVC,

E.2.3 Responsibility of CableLabs

CableLabs is responsible for the following:

- Store the "CableLabs Certified(TM)" CVC private key in secret,
- Maintain the "CableLabs Certified(TM)" CVC signed by the DOCSIS Root CA.
- Issue the "CableLabs Certified(TM)" signature file for the DOCSIS 1.1 CM code file certified by CableLabs.

E.2.4 Responsibility of the operators

The operator has the following responsibility and options:

- Check the manufacturer of the code file by verifying the manufacturer's CVC and signature in the code file provided by the CM manufacturer before the operator load the code file on the TFTP server (optional). The code file may be rejected and won't be loaded on the TFTP server if the unexpected manufacturer signs it or the CVC and/or the signature in it are invalid.
- Check if the code file provided by the CM manufacturer is "CableLabs Certified(TM)" by verifying the "CableLabs Certified(TM)"'s CVC and signature in the "CableLabs Certified(TM)" signature file against the code file before the operator load the code file on the TFTP server (optional). CableLabs is likely to post all the "CableLabs Certified(TM)" signature files and also the corresponding certified code files on the web or FTP server while this is not yet decided. Whether this information is open to only the CableLabs members, all the operators, all the vendors, or pubic is not yet decided

- Operate the operator CA by storing the operator CA private key in secret and maintaining the operator's (co-signer) CVC issued by the DOCSIS Root CA (optional).
- Generate the MSO-controlled code file by adding the operator's CVC and signature to the original code file provided by the CM manufacturer (optional).
- Check if the CVC provided by the CM manufacturer is valid (optional).
- Put the appropriate CVC(s) in the CM configuration file. In case that the original code file is to be downloaded to the CMs, the CM configuration file must contain the valid CVC from the CM's manufacturer. In case that the operator-controlled code file is to be downloaded, the CM configuration file must contain the valid CVC of the operator and may contain the valid CVC from the CM manufacturer. If there is no CVC in the CM configuration file or all the CVC(s) in the CM configuration file is invalid, the CM won't accept any order of the software downloading via the CM configuration file and the MIB. Note that the DOCSIS 1.1 compliant CM may be registered and authorized by the CMTS and becomes operational regardless of whether the CM configuration file contains the valid CVC(s).

Appendix F. Events for Notification

PROCESS	CM PRIORITY	CMTS PRIORITY	EVENT MESSSAGE	EVENT DETAILS	ERROR CODE SET	Trap OID (docsdev.2.xxx) where xxx is (below) is used to identify a specific event
			DHCP and TOD FAILED before registration		D00	
Init	Critical		DHCP FAILED - Discover sent, no offer received		D01.0	1140850944
Init	Critical		DHCP FAILED - Request sent, No response		D02.0	1140851200
Init	Critical		DHCP FAILED - Requested Info not supported.		D03.0	1140851456
Init	Critical		DHCP FAILED - Response doesn't contain ALL the valid fields as describe in the RFI spec Appendix D		D03.1	1140851457
			DOWNSTREAM ACQUISITION FAILED		T00	
Init	Critical		SYNC Timing Synchronization failure, Failed to acquire QAM/QPSK symbol timing, Error stats Retry #'s		T01.0	1409286400
Init	Critical		SYNC Timing Synchronization failure, Failed to acquire FEC framing. Error stats Retry #'s # of bad frames		T02.0	1409286656
Init	Critical		SYNC Timing Synchronization failure, Acquired FEC framing. Failed to acquire MPEG2 Sync. Retry #'s		T02.1	1409286657
Init	Critical		SYNC Timing Synchronization failure, Failed to acquire MAC framing. Error stats Retry #'s Of bad frames		T03.0	1409286912
Init	Critical		SYNC Timing Synchronization failure, Failed to receive MAC SYNC frame within time-out period.		T04.0	1409287168

PROCESS	CM PRIORITY	CMTS PRIORITY	EVENT MESSSAGE	EVENT DETAILS	ERROR CODE SET	Trap OID (docsdev.2.xxx) where xxx is (below) is used to identify a specific event
Init	Critical		SYNC Timing Synchronization failure, Loss of Sync. (Missed 5 in a row, after having SYNC'd at one time)		T05.0	1409287424
			FAILED TO OBTAIN UPSTREAM PARAMETERS		U00	
Init	Critical		No UCD's Received. Timeout.		U01.0	1426063616
Init	Critical		UCD invalid or channel unusable.		U02.0	1426063872
Init	Critical		UCD, & SYNC valid, NO MAPS for this channel		U04.0	
Init	Critical		US channel wide parameters not set before Burst Descriptors.		U06.0	1426064896
			RANGING FAILED : RNG-REQ RANGING REQUEST		R00	
Init	Critical		No Maintenance Broadcasts for Ranging opportunities received T2 time-out.		R01.0	1375731968
Init	Critical		Received Response to Broadcast Maintenance Request, But no Unicast Maintenance opportunities received. T4 timeout.		R04.0	1375732736
			RANGING FAILED : RNG-REQ RANGING RESPONSE			
Init	Critical		No Ranging Response received, T3 time-out.		R02.0	1375732224
Init	Critical		Ranging Request Retries exhausted		R03.0	1375732480
Init	Critical		Started Unicast Maintenance Ranging no Response received. T3 time-out.		R05.0	1375732992
Init	Critical		Unicast Maintenance Ranging attempted. No response. Retries		R06.0	1375733248

PROCESS	CM PRIORITY	CMTS PRIORITY	EVENT MESSSAGE	EVENT DETAILS	ERROR CODE SET	Trap OID (docsdev.2.xxx) where xxx is (below) is used to identify a specific event
			exhausted.			
Init	Critical		Unicast Ranging Received Abort Response. Re-initializing MAC.		R07.0	1375733504
			ToD FAILED			
Init	Error		Time of Day Request sent no Response received.		D04.1	1140851713
Init	Error		Time of Day Response received but invalid data/format.		D04.2	1140851714
			DHCP and TOD FAILED before registration		D00	
Init	Critical		TFTP failed, request sent, No Response/No server.		D05.0	1140851968
Init	Critical		TFTP failed, configuration file NOT FOUND.		D06.0	1140852224
Init	Critical		TFTP Failed, OUT OF ORDER packets.		D07.0	1140852480
Init	Critical		TFTP complete, but failed Message Integrity check (MIC)		D08.0	1140852736
			REGISTRATION FAILED (REG-REQ REGISTRATION REQUEST)		I10	
Init	Critical	Warning	Registration Failed, Service not available	Unrecognized configuration setting	I04.1	1224737793
	Critical	Warning		Temporarily unavailable	I04.2	1224737794
	Critical	Warning		Permanent.	I04.3	1224737795
Init	Critical	Warning	Registration Failed - Registration request	Invalid MAC header.	I101.0	1224762624
	Critical	Warning		Invalid SID, not in use.	I102.0	1224762880
	Critical	Warning		Required TLV's not present.	I104.0	1224763392
Init	Critical	Warning	Registration Failed, Bad Downstream Frequency	Format Invalid	I105.0	1224763648
	Critical	Warning		Not in use	I105.1	1224763649

PROCESS	CM PRIORITY	CMTS PRIORITY	EVENT MESSSAGE	EVENT DETAILS	ERROR CODE SET	Trap OID (docsdev.2.xxx) where xxx is (below) is used to identify a specific event
	Critical	Warning		Not multiple of 62500Hz	I105.2	1224763650
Init	Critical	Warning	Registration Failed, Bad Upstream Channel	Invalid, unassigned	I106.0	1224763904
	Critical	Warning		Change followed with (RE-) Registeration REQ.	I106.1	1224763905
Init	Critical	Warning	Registration Failed, Network Access configuration has invalid parameter.		I108.0	1224764416
Init	Critical	Warning	Registration Failed, Bad Class of Service	configuration is invalid.	I109.0	1224764672
	Critical	Warning		Service ID unsupported	I110.0	1224764928
	Critical	Warning		Service ID invalid or out of range.	I111.0	1224765184
Init	Critical	Warning	Registration Failed, Bad Max Downstream Bit	configuration is invalid format	I112.0	1224765440
	Critical	Warning		configuration setting is unsupported.	I112.1	1224765441
Init	Critical	Warning	Registration Failed,Bad Max Upstream Bit Rate	Configuration setting invalid format	I113.0	1224765696
	Critical	Warning		Configuration setting unsupported	I113.1	1224765697
Init	Critical	Warning	Registration Failed,Bad Upstream Priority configuration	invalid format.	I114.0	1224765952
	Critical	Warning		setting out of range.	I114.1	1224765953
Init	Critical	Warning	Registration Failed,Bad Guaranteed Min Upstream Channel Bit rate configuration setting	invalid format.	I115.0	1224766208
	Critical	Warning		exceeds Max Upstream Bit rate.	I115.1	1224766209
	Critical	Warning		out of range.	I115.2	1224766210
Init	Critical	Warning	Registration Failed,Bad Max Upstream Channel	invalid format.	I116.0	1224766464

PROCESS	CM PRIORITY	CMTS PRIORITY	EVENT MESSSAGE	EVENT DETAILS	ERROR CODE SET	Trap OID (docsdev.2.xxx) where xxx is (below) is used to identify a specific event
			Transmit Burst configuration setting			
	Critical	Warning		out of range	I116.1	1224766465
Init	Critical	Warning	Registration Failed,Modem Capabilities configuration setting invalid format.		I117.0	1224766720
Init	Critical	Warning	Registration Failed, Config file parameters outside the range e.g. # of CPE's given in config file more than the Spec.		I118.0	1224766976
			VERSION 1.1 SPECIFIC REG-REQ REGISTRATION REQUEST		I200.0	
Init	Critical	Warning	DOCSIS 1.1 Registration rejected	unspecified reason.	I201.0	1224788224
	Critical	Warning		unrecognized configuration setting.	I201.0	1224788224
	Critical	Warning		temporary no resource.	I201.2	1224788226
	Critical	Warning		permanent administrative.	I201.3	1224788227
	Critical	Warning		required parameter not present.	I201.4	1224788228
	Critical	Warning		header suppression setting not supported.	I201.5	1224788229
			REG-RSP REGISTRATION RESPONSE		I00.0	
Init	Critical	Warning	Registration RESP Bad.	invalid format or not recognized.	I01.0	1224737024
	Critical	Warning		bad SID.	I03.0	1224737536
Init	Critical	Warning	Registration RESP not received.		I02.0	1224737280
			REG-ACK REGISTRATION ACKNOWLEDGEMENT		I300.0	
Init	Critical	Warning	Registration aborted no REG-ACK.		I301.0	1224813824
			TLV-11 ERRORS		I400.0	

PROCESS	CM PRIORITY	CMTS PRIORITY	EVENT MESSSAGE	EVENT DETAILS	ERROR CODE SET	Trap OID (docsdev.2.xxx) where xxx is (below) is used to identify a specific event
Init	Error		TLV-11 error	CM SNMP Object(s) ignored	I401.0	1224839424
SW Upgrade	Information		SW download initiated	NMS_Initiated <IP address>, <TFTP server address>, < Filename>	E101.0	1157653760
	Information			CFG_File_Initiated <TFTP server address>, < Filename>	E102.0	1157654016
SW Upgrade	Error		SW download Failed	Max retry exceeded	E103.0	1157654272
SW Upgrade	Error		SW upgrade Error - before sucessful TFTP	Server not present <TFTP server addr>	E104.0	1157654528
	Error			File not present <Filename>	E105.0	1157654784
	Error			Tftp Max retry exceeded	E106.0	1157655040
SW Upgrade	Error		SW upgrade Error - after sucessful TFTP	Incompatible file	E107.0	1157655296
				File corruption	E108.0	1157655552
SW Upgrade	Error		Disruption during SW download.	Power lost	E109.0	1157655808
				RF removed.	E110.0	1157656064
SW Upgrade	Notice		SW download Successful	NMS_Initiated <IP address>, <TFTP server address>, < Filename>	E111.0	1157656320
				CFG_File_Initiated <TFTP server address>, < Filename>	E112.0	1157656576
DHCP			DHCP renewal failure		D100.0	
	Error			Renew sent, No response	D101.0	1140876544
	Error			Rebind Sent, No response	D102.0	1140876800
DHCP	Error		Invalid DHCP Options	Renew	D103.0	1140877056
	Error			Rebind	D104.0	1140877312

PROCESS	CM PRIORITY	CMTS PRIORITY	EVENT MESSSAGE	EVENT DETAILS	ERROR CODE SET	Trap OID (docsdev.2.xxx) where xxx is (below) is used to identify a specific event
				DYNAMIC SERVICES	S00	
DYNAMIC SERVICES	Error	Warning	Dynamic Service requests - Service add rejected	unspecified reason	S01.0	1392509184
	Error	Warning		unrecognized configuration setting	S01.1	1392509185
	Error	Warning		permanent administrative	S01.3	1392509187
	Error	Warning		required parameter not present	S01.4	1392509188
	Error	Warning		HMAC authentication failure	S01.7	1392509191
DYNAMIC SERVICES	Error	Warning	Dynamic Service requests - Service change rejected	unspecified reason	S02.0	1392509440
	Error	Warning		unrecognized configuration setting	S02.1	1392509441
	Error	Warning		requestor not owner of service flow	S02.4	1392509444
	Error	Warning		service flow not found	S02.5	1392509445
	Error	Warning		HMAC authentication failure	S02.8	1392509448
DYNAMIC SERVICES	Error	Warning	Dynamic Service requests - Service delete rejected	unspecified reason	S03.0	1392509696
	Error	Warning		service flow not found.	S03.2	1392509698
	Error	Warning		HMAC authentication failure	S03.3	1392509699
			DYNAMIC SERVICE RESPONSES			
DYNAMIC SERVIC	Error	Warning	Service add response rejected invalid transaction ID		S101.0	1392534784

PROCESS	CM PRIORITY	CMTS PRIORITY	EVENT MESSSAGE	EVENT DETAILS	ERROR CODE SET	Trap OID (docsdev.2.xxx) where xxx is (below) is used to identify a specific event
ES						
DYNAMIC SERVICES	Error	Warning	Service change response rejected invalid transaction ID.		S102.0	1392535040
DYNAMIC SERVICES	Error	Warning	Service delete response rejected invalid transaction ID		S103.0	1392535296
			DYNAMIC SERVICE ACKNOWLEDGEMENTS			
DYNAMIC SERVICES	Error	Warning	Dynamic Service Acknowledgements - Service add	ACK rejected invalid transaction ID	S201.0	1392560384
	Error	Warning		aborted no ACK	S201.1	1392560385
DYNAMIC SERVICES	Error	Warning	Dynamic Service Acknowledgements - Service change	ACK rejected invalid transaction ID	S202.0	1392560640
			CM CONFIGURATION FILE		B100	
Init (BPI+)	Warning	Error	Missing BP Configuration Setting TLV		B101.0	1107322112
Init (BPI+)	Warning	Error	Invalid BP Configuration Setting Value		B102.0	1107322368
			CERTIFICATE VERIFICATION		B200	
Init (BPI+)	Error	Error	CM Certificate Format Error		B201.0	1107347712
Init (BPI+)	Error	Error	Manufactrue CA Certificate Format Error		B202.0	1107347968
Init (BPI+)	Error	Error	CM Certificate Self-Verification Failure		B203.0	1107348224
			AUTH FSM		B300	
BPKM	Warning	Error	Auth Reject -- No		B301.2	1107373314

PROCESS	CM PRIORITY	CMTS PRIORITY	EVENT MESSSAGE	EVENT DETAILS	ERROR CODE SET	Trap OID (docsdev.2.xxx) where xxx is (below) is used to identify a specific event
			Information			
BPKM	Warning	Error	Auth Reject -- Unauthorized CM		B301.3	1107373315
BPKM	Warning	Error	Auth Reject -- Unauthorized SAID		B301.4	1107373316
BPKM	Warning	Error	Auth Reject -- Parmanent Authorization Failure		B301.8	1107373320
BPKM	Warning	Error	Auth Invalid -- No Information		B302.2	1107373570
BPKM	Warning	Error	Aufh Invalid -- Unauthorized CM		B302.3	1107373571
BPKM	Warning	Error	Auth Invalid -- Unsolicited		B302.5	1107373573
BPKM	Warning	Error	Auth Invalid -- Invalid Key Sequence Number		B302.6	1107373574
BPKM	Warning	Error	Auth Invalid -- Message (Key Request) Authentication Failure		B302.7	1107373575
BPKM	Warning	Error	Unsupported Crypto Suite		B303.0	1107373824
			EVENT BETWEEN AUTH & TEK FSM		B400	
BPKM	Informational		Authorized		B401.0	1107398912
BPKM	Informational		Auto Pend		B402.0	1107399168
BPKM	Informational		Auth Comp		B403.0	1107399424
BPKM	Informational		Stop		B404.0	1107399680
			TEK FSM		B500	
BPKM	Warning	Error	Key Reject -- No Information		B501.2	1107424514
BPKM	Warning	Error	Key Reject -- Unauthorized SAID		B501.3	1107424515
BPKM	Warning	Error	TEK Invalid -- No Information		B502.3	1107424771
BPKM	Warning	Error	TEK Invalid -- Invalid Key Sequence Number		B502.6	1107424774
			SA MAP FSM		B600	

PROCESS	CM PRIORITY	CMTS PRIORITY	EVENT MESSSAGE	EVENT DETAILS	ERROR CODE SET	Trap OID (docsdev.2.xxx) where xxx is (below) is used to identify a specific event
Dynamic SA	Informational		SA Map State Machine Started		B601.0	1107450112
Dynamic SA	Warning	Error	Unsupported Crypto Suite		B602.0	1107450368
Dynamic SA	Error		Map Request Retry Timeout		B603.0	1107450624
Dynamic SA	Notice		Unmap		B604.0	1107450880
Dynamic SA	Warning	Error	Map Reject -- Not Authorized for Requested Dwonstream Traffic Flow (EC=7)		B605.9	1107451145
Dynamic SA	Warning	Error	Map Reject -- Dwonstream Traffic Flow Not Mapped to BPI+ SAID (EC=8)		B605.10	1107451137
Dynamic SA	Warning	Error	Mapped to Existing SAID		B606.0	1107451392
Dynamic SA	Warning	Error	Mapped to New SAID		B607.0	1107451648
			VERIFICAITON OF CODE FILE		E200	
SW Upgrade	Error		Improper Code File Controls		E201.0	1157679360
SW Upgrade	Error		Code File Manufacturer CVC Validation Failure		E202.0	1157679616
SW Upgrade	Error		Code File Manufacturer CVS Validation Failure		E203.0	1157679872
SW Upgrade	Error		Code File Co-Signer CVC Validation Failure		E204.0	1157680128
SW Upgrade	Error		Code File Co-Signer CVS Validation Failure		E205.0	1157680384
SW Upgrade	Error		Improper Configuration File CVC Format		E206.0	1157680640
SW Upgrade	Error		Configuration File CVC Validation Failure		E207.0	1157680896
SW Upgrade	Error		Improper SNMP CVC Format		E208.0	1157681152

Appendix G. Trap Definitions for Cable Device

The trap definition for cable device will be specified in this section by the ECR/ECO/ECN process.

Appendix I. Application of RFC-2933 to DOCSIS 1.1 active/passive IGMP devices

I.1 DOCSIS 1.1 IGMP MIBs

DOCSIS 1.1 devices, CM and CMTS, that support IGMP (in active or passive mode), MUST support the IDMR IGMP MIB (RFC-2933). As such, this section describes the application of the IETF IDMR sub-committee IGMP MIB to DOCSIS 1.1 active/passive IGMP devices.

The IDMR IGMP MIB is organized into two distinct tables, the interface and cache tables. The IGMP Interface Table contains entries for each interface that supports IGMP on a device. For DOCSIS 1.1 this includes the NSI and HFC for the CMTS and the HFC and CMCI on the CM. The IGMP Cache Table contains one row for each IP Multicast Group for which there are active members on a given interface. Active membership MUST only exist on the CMCI of a Cable Modem. However, active membership MAY exist on both the NSI and HFC side interfaces of the CMTS. This is because a CMTS may be implemented as a Multicast Router on which other network side devices are actively participating in a multicast session.

Support of the IDMR IGMP MIB by DOCSIS 1.1 devices is presented in terms of IGMP capabilities, the device type (CM or CMTS), and the interface on which IGMP is supported. This is followed by a set of new IGMP MIB conformance, compliance and group statements for DOCSIS 1.1 devices.

I.1.1 IGMP Capabilities: Active and Passive Mode

There are two basic modes of IGMP capability that are applicable to a DOCSIS 1.1 device. The first mode is a *passive* operation in which the device selectively forwards IGMP based upon the known state of multicast session activity on the subscriber side (an example of this is described in Appendix L of [MCNS 5]). In *passive* mode, the device derives its IGMP timers based on the rules specified in section 3.3.1 of the RFI. The second mode is an *active* operation in which the device terminates and initiates IGMP based upon the known state of multicast session activity on the subscriber side. One example of the latter, active, mode is commonly referred to as an IGMP-Proxy implementation side (as described in [ID-IGMP]). A more complete example of an active IGMP device is that of a Multicast Router. Although a specific implementation is not imposed by the DOCSIS 1.1 specification, the device MUST meet the requirements stated in section 3.3.1 of [MCNS 5] and MUST support the IDMR IGMP MIB as described herein. As presently specified in the DOCSIS 1.1, active CMs are explicitly prohibited from transmitting IGMP Queries upstream onto the HFC. However, active CMTSs may transmit IGMP Queries onto the NSI as mentioned previously.

I.1.2 IGMP Interfaces

A description of the application of the IDMR IGMP MIB to DOCSIS 1.1 devices follows. This description is organized by CM and CMTS device type.

I.2 Docsis 1.1 CM Support for the IGMP MIB

There are two types of interfaces applicable to IGMP on the DOCSIS 1.1 CM. These are the HFC-Side and CMCI-Side interfaces, respectively. Application of the IGMP MIB to DOCSIS 1.1 CMs is presented in terms of *passive* and *active* CM operation and these two interface types.

I.2.1 igmpInterfaceTable- igmpInterfaceEntry

I.2.1.1 igmpInterfaceIfIndex

The ifIndex value of the interface for which IGMP is enabled.

I.2.1.1.1 All Modes

This is the same for passive and active modes.

HFC-side: not-accessible. ifIndex of docsCableMaclayer(127), CATV MAC Layer
CMCI-side: not-accessible. ifIndex of CMCI-Side interface.

I.2.1.2 igmpInterfaceQueryInterval

The frequency at which IGMP Host-Query packets are transmitted on this interface.

I.2.1.2.1 Passive Mode

HFC-side: n/a, read-only. The CM MUST not transmit queries upstream. Return a value of zero.
CMCI-side: read only . This value is derived based on the interval of queries received from an upstream querier.

I.2.1.2.2 Active Mode

HFC-side: n/a, read-only. The CM MUST not transmit queries upstream. Return a value of zero.
CMCI-side: read-create. Min = 0; Max = $(2^{\wedge 32}-1)$; Default = 125

I.2.1.3 igmpInterfaceStatus

The activation of a row enables IGMP on the interface. The destruction of a row disables IGMP on the interface.

I.2.1.3.1 All Modes

MUST be enabled on both interfaces for all DOCSIS 1.1 CM interfaces.

I.2.1.4 igmpInterfaceVersion

The version of IGMP which is running on this interface. MUST be version 2 for all DOCSIS 1.1 CM interfaces.

I.2.1.5 igmpInterfaceQuerier

The address of the IGMP Querier on the IP subnet to which this interface is attached.

I.2.1.5.1 Passive Mode

HFC-side: read-only. MUST be the address of an upstream IGMP Querier device for both active and passive CMs.

CMCI-side: read-only. Same as HFC-side value.

I.2.1.5.2 Active Mode

HFC-side: read-only. MUST be the address of an upstream IGMP Querier device for both active and passive CMs.

CMCI-side: read-only. Active CMs may report it as the HFC-side value. However, active CM's that participate in IGMP Querier negotiation on the CMCI may report it as a different CPE.

I.2.1.6 igmpInterfaceQueryMaxResponseTime

The maximum query response time advertised in IGMPv2 queries on this interface.

I.2.1.6.1 Passive Mode

HFC-side: n/a, read-only. return a value of zero.

CMCI-side: read-only. This value is derived from observation of queries received from an upstream querier

I.2.1.6.2 Active Mode

HFC-side: n/a, read-only. return a value of zero.

CMCI-side: read-create. Min = 0; Max = 255; Default = 100.

I.2.1.7 igmpInterfaceQuerierUpTime

The time since igmpInterfaceQuerier was last changed.

I.2.1.7.1 PassiveMode

HFC-side: read-only.

CMC-side: n/a, read-only. Return a value of zero.

I.2.1.7.2 Active Mode

HFC-side: read-only.

CMCI-side: read-only.

I.2.1.8 igmpInterfaceQuerierExpiryTime

The amount of time remaining before the other querier present timer expires. If the local system is the queier, the value of this object is zero.

I.2.1.8.1 Passive Mode

Both interfaces: n/a, read-only. The CM is never the querier, return 0.

I.2.1.8.2 Active Mode

HFC-side: n/a, read-only. Return 0.

CMCI-side: read-only. The CM may only be the querier on the CMCI.

I.2.1.9 igmpInterfaceVersion1QuerierTimer

The time remaining until the host assumes that there are no IGMPv1 routers present on the interface. While this is non-zero, the host will reply to all queries with version 1 membership reports.

I.2.1.9.1 Passive Mode

HFC-side: n/a read-only. Return a value of zero.

CMCI-side: n/a read-only. Return a value of zero.

I.2.1.9.2 Active Mode

HFC-side: read-only.

CMCI-side: read-only.

I.2.1.10 igmpInterfaceWrongVersionQueries

The number of queries recived whose IGMP version does not match igmpInterfaceVersion, over the lifetime of the row entry. IGMP requires that all routers on a LAN be configured to run the same version of IGMP. Although, DOCSIS 1.1 requires that all CM and CMTS devices support IGMPv2, it is possible for an upstream querier to be an IGMPv1 querier.

I.2.1.10.1 All Modes

All interfaces: read-only. The number of non-v2 queries received on this interface.

I.2.1.11 igmpInterfaceJoins

The number of times a group membership has been added on this interface; that is, the number of times an entry for this interface has been added to the Cache Table. This object gives an indication of the amount of IGMP activity over the lifetime of the row entry.

I.2.1.11.1 All HFC-side: n/a, read-only. Always return a value of zero (see CMCI-side).

I.2.1.11.2 All CMCI-side: read-only. Group membership is defined to only exist on the CMCI.

I.2.1.12 igmpInterfaceProxyIfIndex

Some devices implement a form of IGMP proxying whereby memberships learned on the interface represented by this row, cause IGMP Host Membership Reports to be sent on the interface whose ifIndex value is given by this object. Such a device would implement the igmpV2RouterMIBGroup only on its router interfaces (those interfaces with non-zero igmpInterfaceProxyIfIndex). Typically, the value of this object is 0, indicating that no proxying is being done.

I.2.1.12.1 Passive Mode

All Interfaces: read-only. Always return a value of zero.

I.2.1.12.2 Active Mode

HFC-side: read-only. Always return a value of zero.

CMCI-side: read-only. Always return a ifIndex for HFC-side interface.

I.2.1.13 igmpInterfaceGroups

The current number of entries for this interface in the Cache Table.

I.2.1.13.1 All HFC-side: n/a, read-only. Always return a value of zero (see CMCI-side).

I.2.1.13.2 All CMCI-side: read-only. Group membership is defined to only exist on the CMCI.

Number of active sessions Proxied or Active on this Interface.

I.2.1.14 igmpInterfaceRobustness

The robustness variable allows tuning for the expected packet loss on a subnet. If a subnet is expected to be lossy, the robustness variable may be increased. IGMP is robust to (robustness variable – 1) packet losses.

I.2.1.14.1 Passive Mode

HFC-side: n/a read-only. Return a value of zero.

CMCI-side: n/a read-only. Return a value of zero.

I.2.1.14.2 Active Mode

All interfaces: read-create. Min = 1; Max = ($2^{\wedge 32}$-1); Default = 2

I.2.1.15 igmpInterfaceLastMemberQueryIntvl

The last member query interval is the max response time inserted into group specific queries sent in response to leave group messages, and is also the amount of time between group specific query messages. This value may be tuned to modify the leave latency of the network. A reduced value results in reduced time to detect the loss of the last member of a group.

I.2.1.15.1 Passive Mode

HFC-side: n/a, read-only. return a value of zero.

CMCI-side: read-only. This value is derived from observation of queries received from an upstream querier

I.2.1.15.2 Active Mode

HFC-side: n/a, read-only. return a value of zero.

CMCI-side: read-create. Min = 0; Max = 255; Default = 100.

I.2.2 igmpCacheTable - igmpCacheEntry

I.2.2.1 igmpCacheAddress

The IP multicast group address for which this entry contains information.

I.2.2.1.1 All Modes

Not-accessible (index). Report the address of active IP Multicast on the CMCI interface.

I.2.2.2 igmpCacheIfIndex

The interface for which this entry contains information for an IP multicast group address.

I.2.2.2.1 All Modes

MUST only apply to CMCI interface (e.g., membership is only active on subscriber side of CM).

I.2.2.3 igmpCacheSelf

An indication of whether the local system is a member of this group address on this interface.

I.2.2.3.1 Passive Mode

read-only. MUST be set to FALSE. The CM is not a member of any group.

I.2.2.3.2 Active Mode

read-create. Implementation specific. If the CM is configured to be a member of the group, then membership reports are sent with the CM's IP Address but MUST ONLY be sent in proxy for active sessions on the CMCI (e.g., the CM MUST NOT be a member of a multicast group that is not active on the CMCI). If the CM is not configured to be a member, then the source IP Address of membership reports MUST be set to the current value of the igmpCacheLastReporter address.

I.2.2.4 igmpCacheLastReporter

The IP address of the source of the last membership report received for this IP Multicast group address on this interface. If no membership report has been received, this object has the value of 0.0.0.0.

I.2.2.4.1 All Modes

MUST only apply to last reporter on CMCI interface (e.g., membership is only active on subscriber side of CM).

I.2.2.5 igmpCacheUpTime

The time elapsed since this entry was created.

I.2.2.5.1 All Modes

read-only. MUST only apply to duration of membership on CMCI interface (e.g., membership is only active on subscriber side of CM).

I.2.2.6 igmpCacheExpiryTime

The minimum amount of time remaining before this entry will be aged out.

I.2.2.6.1 All Modes

read-only. MUST only apply to duration of membership on CMCI interface (e.g., membership is only active on subscriber side of CM).

I.2.2.7 igmpCacheStatus

The status of this entry.

I.2.2.7.1 All Modes

read-create. MUST only apply to membership on CMCI interface (e.g., membership is only active on subscriber side of CM). Deletion of a row results in preventing downstream forwarding to this IP Multicast group address on this interface.

I.2.2.8 igmpCacheVersion1HostTimer

The time remaining until the local querier will assume that there are no longer any IGMP version 1 members on thie IP subnet attached to this interface. Upon hearing any IGMPv1 membership report, this value is reset to the group membership timer. While this time remaining is non-zero, the local querier ignores any IGMPv2 leave messages for this group that it receives on this interface.

I.2.2.8.1 Passive Mode

All interfaces: n/a, read-only. Return a value of zero.

I.2.2.8.2 Active Mode

HFC-side: n/a, read-only. Return a value of zero.

CMCI-side: read-only.

I.3 Docsis 1.1 CMTS Support for the IGMP MIB

There are two types of interfaces applicable to IGMP on the DOCSIS 1.1 CMTS. These are the NSI-Side and NSI-Side interfaces, respectively. Application of the IGMP MIB to DOCSIS 1.1 CMTS's is presented in terms of *passive* and *active* CMTS operation and these two interface types.

It is important to note that an *active* IGMP capable CMTS may be implemented as a proxy, router, or hybrid device. As such, the CMTS may be capable of querying on both its NSI and HFC side interfaces and may manage membership for devices on its NSI interfaces (e.g., as a multicast router). This is different than an *active* CM, which MUST NOT query on its HFC side interface (e.g., it may only query on its CMCI). This capability is accounted for in the application of the IGMP MIB to the CMTS.

I.3.1 igmpInterfaceTable- igmpInterfaceEntry

I.3.1.1 igmpInterfaceIfIndex

The ifIndex value of the interface for which IGMP is enabled.

I.3.1.1 All Modes

This is the same for passive and active modes.

NSI-side: not-accessible. ifIndex of applicable network side interface(s).
HFC-side: not-accessible. ifIndex of docsCableMaclayer(127), CATV MAC Layer interface.

I.3.1.2 igmpInterfaceQueryInterval

The frequency at which IGMP Host-Query packets are transmitted on this interface.

I.3.1.2.1 Passive Mode

NSI-side: n/a, read-only. Return a value of zero.
HFC-side: read only . This value is derived based on the interval of queries received from a Network Side querier.

I.3.1.2.2 Active Mode

NSI-side: read-create. Min = 0; Max = $(2^{\wedge 32}-1)$; Default = 125
HFC-side: read-create. Min = 0; Max = $(2^{\wedge 32}-1)$; Default = 125

I.3.1.3 igmpInterfaceStatus

I.3.1.3.1 All Modes

The activation of a row enables IGMP on the interface. The destruction of a row disables IGMP on the interface.

I.3.1.4 igmpInterfaceVersion

The version of IGMP which is running on this interface. MUST be version 2 for all DOCSIS 1.1 CMTS interfaces.

I.3.1.5 igmpInterfaceQuerier

The address of the IGMP Querier on the IP subnet to which this interface is attached.

I.3.1.5.1 Passive Mode

NSI-side: read-only. This is the address of a network side device.

HFC-side: read-only. Same as NSI-side value.

I.3.1.5.2 Active Mode

NSI-side: read-only.

HFC-side: read-only. Active CMTSs MUST report this as an IP Address assigned to the CMTS' HFC-side interface. That is, queries MUST not originate from CMs or CPE.

I.3.1.6 igmpInterfaceQueryMaxResponseTime

The maximum query response time advertised in IGMPv2 queries on this interface.

I.3.1.6.1 Passive Mode

NSI-side: n/a, read-only. return a value of zero.

HFC-side: read-only. This value is derived from observation of queries received from a network side querier.

I.3.1.6.2 Active Mode

NSI-side: read-create. Min = 0; Max = 255; Default = 100.

HFC-side: read-create. Min = 0; Max = 255; Default = 100.

I.3.1.7 igmpInterfaceQuerierUpTime

The time since igmpInterfaceQuerier was last changed.

I.3.1.7.1 PassiveMode

NSI-side: read-only.

HFC-side: n/a, read-only. Return a value of zero.

I.3.1.7.2 Active Mode

NSI-side: read-only.

HFC-side: read-only.

I.3.1.8 igmpInterfaceQuerierExpiryTime

The amount of time remaining before the other querier present timer expires. If the local system is the queier, the value of this object is zero.

I.3.1.8.1 Passive Mode

Both interfaces: n/a, read-only. The CMTS is not the querier, return 0.

I.3.1.8.2 Active Mode

NSI-side: read-only.

HFC-side: read-only. The CMTS MUST be the only querier on the HFC.

I.3.1.9 igmpInterfaceVersion1QuerierTimer

The time remaining until the host assumes that there are no IGMPv1 routers present on the interface. While this is non-zero, the host will reply to all queries with version 1 membership reports.

I.3.1.9.1 Passive Mode

NSI-side: n/a read-only. Return a value of zero.

HFC-side: n/a read-only. Return a value of zero.

I.3.1.9.2 Active Mode

NSI-side: read-only.

HFC-side: read-only.

I.3.1.10 igmpInterfaceWrongVersionQueries

The number of queries recived whose IGMP version does not match igmpInterfaceVersion, over the lifetime of the row entry. IGMP requires that all routers on a LAN be configured to run the same version of IGMP. Although, DOCSIS 1.1 requires that all CMTS and CMTSTS devices support IGMPv2, it is possible for a network side querier to be an IGMPv1 querier.

I.3.1.10.1 All Modes

All interfaces: read-only. The number of non-v2 queries received on this interface.

I.3.1.11 igmpInterfaceJoins

The number of times a group membership has been added on this interface; that is, the number of times an entry for this interface has been added to the Cache Table. This object gives an indication of the amount of IGMP activity over the lifetime of the row entry.

I.3.1.11.1 Passive Mode

NSI-side: n/a read-only. Return a value of zero.

HFC-side: n/a read-only. Return a value of zero.

I.3.1.11.2 Active Mode

NSI-side: read-only.

HFC-side: read-only.

I.3.1.12 igmpInterfaceProxyIfIndex

Some devices implement a form of IGMP proxying whereby memberships learned on the interface represented by this row, cause IGMP Host Membership Reports to be sent on the interface whose ifIndex value is given by this object. Such a device would implement the igmpV2RouterMIBGroup only on its router interfaces (those interfaces with non-zero igmpInterfaceProxyIfIndex). Typically, the value of this object is 0, indicating that no proxying is being done.

I.3.1.12.1 Passive Mode

All Interfaces: read-only. Always return a value of zero.

I.3.1.12.2 Active Mode

NSI-side: read-only.

HFC-side: read-only. Always return an ifIndex for a NSI-side interface.

I.3.1.13 igmpInterfaceGroups

The current number of entries for this interface in the Cache Table.

I.3.1.13.1 Passive Mode

NSI-side: n/a read-only. Return a value of zero.

HFC-side: n/a read-only. Group membership of HFC-side devices.

I.3.1.13.2 Active Mode

NSI-side: read-only.

HFC-side: read-only.

I.3.1.14 igmpInterfaceRobustness

The robustness variable allows tuning for the expected packet loss on a subnet. If a subnet is expected to be lossy, the robustness variable may be increased. IGMP is robust to (robustness variable – 1) packet losses.

I.3.1.14.1 Passive Mode

NSI-side: n/a read-only. Return a value of zero.

HFC-side: n/a read-only. Return a value of zero.

I.3.1.14.2 Active Mode

All interfaces: read-create. Min = 1; Max = ($2^{\wedge 32}$-1); Default = 2

I.3.1.15 igmpInterfaceLastMemberQueryIntvl

The last member query interval is the max response time inserted into group specific queries sent in response to leave group messages, and is also the amount of time between group specific query messages. This value may be tuned to modify the leave latency of the network. A reduced value results in reduced time to detect the loss of the last member of a group.

I.3.1.15.1 Passive Mode

NSI-side: n/a, read-only. return a value of zero.

HFC-side: read-only. This value is derived from observation of queries received from a network side querier.

I.3.1.15.2 Active Mode

NSI-side: read-create. Min = 0; Max = 255; Default = 100.

HFC-side: read-create. Min = 0; Max = 255; Default = 100.

I.3.2 igmpCacheTable - igmpCacheEntry

I.3.2.1 igmpCacheAddress

The IP multicast group address for which this entry contains information.

I.3.2.1.1 All Modes

Not-accessible (index). Report the address of active IP Multicast on the interface.

I.3.2.2 igmpCacheIfIndex

The interface for which this entry contains information for an IP multicast group address.

I.3.2.2.1 Passive Mode

MUST only apply to HFC side interface (e.g., membership is only active on subscriber side of CMTS).

I.3.2.2.2 Active Mode

NSI-side: not-accessible

HFC-side: not-accessible

I.3.2.3 igmpCacheSelf

An indication of whether the local system is a member of this group address on this interface.

I.3.2.3.1 Passive Mode

read-only. MUST be set to FALSE. The CMTS is not a member of any group.

I.3.2.3.2 Active Mode

NSI-side: read-create. Implementation specific (i.e., may apply to RIPv2 or OSPF)

HFC-side: MUST be set to FALSE. The CMTS is not a member of any group on the HFC.

I.3.2.4 igmpCacheLastReporter

The IP address of the source of the last membership report received for this IP Multicast group address on this interface. If no membership report has been received, this object has the value of 0.0.0.0.

I.3.2.4.1 Passive Mode

MUST only apply to last reporter on HFC-side interface (e.g., membership is only active on subscriber side of CMTS).

I.3.2.4.2 Active Mode

NSI-side: read-only

HFC-side: read-only

I.3.2.5 igmpCacheUpTime

The time elapsed since this entry was created.

I.3.2.5.1 Passive Mode

MUST only apply to duration of membership on HFC-side interface (e.g., membership is only active on subscriber side of CMTS).

I.3.2.5.2 Active Mode

NSI-side: read-only

HFC-side: read-only

I.3.2.6 igmpCacheExpiryTime

The minimum amount of time remaining before this entry will be aged out.

I.3.2.6.1 Passive Mode

MUST only apply to duration of membership on HFC-side interface (e.g., membership is only active on subscriber side of CMTS).

I.3.2.6.2 Active Mode

NSI-side: read-only

HFC-side: read-only

I.3.2.7 igmpCacheStatus

The status of this entry.

I.3.2.7.1 Passive Mode

read-create MUST only apply to membership on HFC-side interface (e.g., membership is only active on subscriber side of CMTS). Deletion of a row results in preventing downstream forwarding to this IP Multicast group address on this interface.

I.3.2.7.2 Active Mode

NSI-side: read-create

HFC-side: read-create

I.3.2.8 igmpCacheVersion1HostTimer

The time remaining until the local querier will assume that there are no longer any IGMP version 1 members on thie IP subnet attached to this interface. Upon hearing any IGMPv1 membership report, this value is reset to the group membership timer. While this time remaining is non-zero, the local querier ignores any IGMPv2 leave messages for this group that it receives on this interface.

I.3.2.8.1 Passive Mode

All interfaces: n/a, read-only. Return a value of zero.

I.3.2.8.2 Active Mode

NSI-side: read-only.
HFC-side: read-only.

I.3.3 IGMP MIB Compliance

I.3.3.1 docsIgmpV2PassiveDeviceCompliance

```
docsIgmpV2PassiveDeviceCompliance MODULE-COMPLIANCE
    STATUS current
    DESCRIPTION
    "The compliance statement for DOCSIS Devices passively running IGMPv2 and
     implementing the IGMP MIB."
    MODULE – this module
    MANDATORY-GROUPS { igmpBaseMIBGroup,
                                        igmpRouterMIBGroup,
                                        igmpV2RouterMIBGroup
                                      }
    OBJECT igmpInterfaceStatus
    MIN-ACCESS read-only
    DESCRIPTION
    "Write access is not required."

    OBJECT igmpCacheStatus
    MIN-ACCESS read-only
    DESCRIPTION
    "Write access is not required."

    ::= {docsIgmpMIBCompliances 1}
```

I.3.3.2 docsIgmpV2ActiveDeviceCompliance

```
docsIgmpV2ActiveCmCompliance MODULE-COMPLIANCE
    STATUS current
    DESCRIPTION
    "The compliance statement for DOCSIS Devices actively running IGMPv2 and
     implementing the IGMP MIB."
    MODULE – this module
    MANDATORY-GROUPS { igmpBaseMIBGroup,
                                        igmpV2HostMIBGroup,
```

```
                igmpRouterMIBGroup,
                igmpV2RouterMIBGroup
            }
    OBJECT igmpInterfaceStatus
    MIN-ACCESS read-only
    DESCRIPTION
    “Write access is not required.”

    OBJECT igmpCacheStatus
    MIN-ACCESS read-only
    DESCRIPTION
    “Write access is not required.”

    ::= {docsIgmpMIBCompliances 2}
```

I.3.4 MIB Groups

See IGMP MIB for a description of the objects included in each group.

I.3.4.1 igmpV2HostMIBGroup

Active Devices only (optional – see notes for igmpCacheSelf).

I.3.4.2 igmpV2RouterMIBGroup

Active and Passive Devices

I.3.4.3 igmpBaseMIBGroup

Active and Passive Devices

I.3.4.4 igmpV2RouterMIBGroup

Active and Passive Devices

I.3.4.5 igmpRouterMIBGroup

Active and Passive Devices

I.3.4.6 igmpV2HostOptMIBGroup

Active and Passive Devices

I.3.4.7 igmpV2ProxyMIBGroup

Active Devices only.

Appendix J. Expected Behaviors for DOCSIS 1.1 modem in 1.0 and 1.1 modes in OSS area

The OSSI Appendix J table identifies DOCSIS OSSI 1.1 CM features that MAY and MUST be implemented in 1.0 mode.

Specific requirement	Required behavior, DOCSIS 1.1 Modem in 1.0 Mode	Required behavior, DOCSIS 1.1 Modem in 1.1 Mode
Assignment of event-id	SHOULD support a 32-bit number with the following requirement: 1) Top bit is set to 0 for DOCSIS standard events; 2) top bit is set to 1 for vendor proprietary events.	MUST be a 32-bit number. Top bit is set to 0 for DOCSIS standard events. Top bit is set to 1 for vendor proprietary events.
Event Definitions	CM SHOULD support DOCSIS standard events defined in the OSSI 1.1 specification.	CM MUST support DOCSIS standard events defined in the OSSI 1.1 specification.
Default handling of events by priority. (Whether to store locally, send trap, or syslog message)	CM SHOULD behave as follow: Error and notice events are stored locally and sent as traps and syslog messages. Other event levels are stored only to the local log, except for informational and debug which are not stored or sent as traps or syslog messages.	CM MUST behave as follows: Error and notice events are stored locally and send traps and syslog messages. Other event levels store only to the local log, except for informational and debug which are not stored or cause any traps or syslog messages.
Meaning of event levels	CM SHOULD support event level definitions specified by the OSSI 1.1 specification.	CM MUST support event level definitions specified by the OSSI 1.1 specification.

Event storage in docsDevEventTable	Each entry in the dosDevEventTable contains an event-ID (identical to the Eventid requirement specified in section 4.4.2.2.2), event time stamp when the event occurred first time and last time, number of appearances and event description in human-readable English format. Total length of the each event description entry MUST not be longer then 255 characters (max. defined for SNMPadminString). Identical events mean that the event-IDs are identical. For those events that may have different display text strings (i.e. display different IP address), only the string from the last event occurred MUST be stored in docsDevEvText.	Each occurrence of an event MUST be logged in a separate row. The first and last times columns MUST be set identically to indicate when the event occurred. The event count MUST always be set to 1.
Number of rows in docsDevEventTable	CM MUST support a minimum of 10 rows of docsDevEventTable.	CM MUST support a minimum of 10 rows of docsDevEventTable.
Event log persistence	Event log MUST persist across reboots	Event log MUST persist across reboots.
SNMP Version of Trap Control (when CM is in SNMP v1/v2c DocsDevNmAccess mode)	CM MUST implement docsDevNmAccessTrapVersion, which controls whether SNMP V1 or V2 traps are sent.	CM MUST implement docsDevNmAccessTrapVersion, which controls whether SNMP V1 or V2 traps are sent.
Syslog message format	CM SHOULD support the syslog message with the format: <level>CABLEMODEM [vendor]: <eventId> text OR <level>Cablemodem [vendor]: text	CM MUST support the syslog message with the format: <level>CABLEMODEM [vendor]: <eventId> text
SNMP Protocol Requirement	CM MUST support SNMP v1/v2c and SNMPv3 with DH. CM must support SNMP requirements specified in section 2.2 of the OSSI.	CM MUST support SNMP v1/v2c and SNMPv3 with DH
MIBs to implement	CM MUST support MIB objects as specified by Appendix A.	CM MUST support MIB objects as specified by Appendix A.
Deprecated MIB objects	Deprecated object is optional. If supported, the object MUST be implemented correctly. If not supported, the object MUST return appropriate SNMP error notifying that the object does not exist.	Deprecated object is optional. If supported, the object MUST be implemented correctly. If not supported, the object MUST return appropriate SNMP error notifying that the object does not exist.

Configuration Management	CM MUST support configuration management requirement as specified by Section 4.2 of the OSSI 1.1 specification.	CM MUST support configuration management requirement as specified by Section 4.2 of the OSSI 1.1 specification.
IP/LLC filters	CM SHOULD support LLC/IP filter requirement as specified by OSSI 1.1 specification.	CM MUST support LLC/IP filter requirement as specified by OSSI 1.1 specification.
CM interaction with CM configuration file	CM MUST process TLV type 11 entries in a configuration file as specified by Section 3.4 of the OSSI 1.1 specification.	CM MUST process TLV type 11 entries in a configuration file as specified by Section 3.4 of the OSSI 1.1 specification.
Additional MIB objects requirement	CM MUST implement additional MIB object requirements (on top of RFCs) as specified in Section 3.3 of the OSSI 1.1 specification.	CM MUST implement additional MIB object requirements (on top of RFCs) as specified in Section 3.3 of the OSSI 1.1 specification.
Performance management	CM MUST support performance management requirements as specified by Section 4.5 of the OSSI 1.1 specification.	CM MUST support performance management requirements as specified by Section 4.5 of the OSSI 1.1 specification.
OSS for CMCI	CM MUST support CMCI requirements as specified by Section 6 of the OSSI 1.1 specification.	CM MUST support CMCI requirements as specified by Section 6 of the OSSI 1.1 specification.

Appendix P. References

[IETF3] draft-ietf-ipcdn-igmp-mib-00.txt, H. Abramson, "Docsis 1.1 IGMP MIB", June 1999

[IETF4] draft-ietf-ipcdn-qos-mib-04.txt, Mike Patrick, "Data Over Cable System Quality of Service Management Information Base", Oct 18, 2000

[IETF6] Proposed Standard RFC version of BPI+ MIB, "**draft-ietf-ipcdn-bpiplus-03.txt"**

[IETF7] Proposed Standard RFC version of USB MIB, "**draft-ietf-xxxx-xxxx-xxxx-00.txt"**

[IETF8] Proposed Standard RFC version of BPI MIB, "**draft-ietf-mcns-bpi-mib-01.txt"**

[IETF9] Proposed Standard RFC version of Customer Management MIB, "**draft-ietf-ipcdn-subscriber-mib-02.txt"**

[MCNS1] MCNS Cable Modem Termination System - Network-Side Interface Specification SP-CMTS-NSI-I01-960702

[MCNS2] MCNS Cable Modem to Customer Premise Equipment Interface SpecificationSP-CMCI-D02c-991015

[MCNS 3] MCNS Operations Support System Framework TR-OSSF-W08-961016

[MCNS 4] MCNS Data Over Cable Services Cable Modem Telephony Return Interface Specification SP-CMTRI-I01-970804

[MCNS 5] MCNS Data Over Cable Services Cable Modem Radio Frequency Interface Specification SP-RFIv1.1-I05-000714

[MCNS 6] MCNS Data Over Cable Services Security Specification SP-SS-I01-970506

[RFC-1157] Schoffstall, M., Fedor, M., Davin, J. and Case, J., A Simple Network Management Protocol (SNMP), IETF RFC-1157, May, 1990

[RFC-1213] K. McCloghrie and M. Rose. Management Information Base for Network Management of TCP/IP-base internets: MIB-II, IETF RFC-1213, March, 1991

[RFC-1224] L. Steinberg., Techniques for Managing Asynchronously Generated Alerts, IETF RFC-1224, May, 1991

[RFC-1493] E. Decker, P. Langille, A. Rijsinghani, and K.McCloghrie., Definitions ofManaged Objects for Bridges, IETF RFC-1493, July, 1993

[RFC-1901] Case, J., McCloghrie, K., Rose, M. and S. Waldbusser, "Introduction to Community-based SNMPv2", RFC 1901, January 1996.

[RFC-1903] Case, J., McCloghrie, K., Rose, M., and S. Waldbusser, "Textual Conventions for Version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1903, January 1996.

[RFC-1905] Case, J., McCloghrie, K., Rose, M. and S. Waldbusser, "Protocol Operations for Version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1905, January 1996.

[RFC-1906] Case, J., McCloghrie, K., Rose, M. and S. Waldbusser, "Transport Mappings for Version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1906, January 1996

[RFC-1907] Case, J., McCloghrie, K., Rose, M. and S. Waldbusser, "Management Information Base for Version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1907, January 1996.

[RFC-2011] K. McCloghrie, "Category: Standards Track SNMPv2 Management Information Base for the Internet Protocol using SMIv2", November 1996

[RFC-2013] K. McCloghrie, "Category: Standards Track SNMPv2 Management Information Base for the User Datagram Protocol using SMIv2", November 1996

[RFC-2132] S. Alexander, R. Droms. DHCP Options and BOOTP Vendor Extensions. IETF RFC-2132. March, 1997.

[RFC-2233] K. McCloghrie, F. Kastenholz, "The Interfaces Group MIB using SMIv2 ", November 1997

[RFC-2358] J. Flick, J. Johnson, "Definitions of Managed Objects for the Ethernet-like Interface Types", June 1998

[RFC-2570] J. Case, R. Mundy, D. Partain, B. Stewart, "Introduction to Version 3 of the Internet-standard Network Management Framework", April 1999

[RFC-2571] Harrington, D., Presuhn, R. and B. Wijnen, "An Architecture for Describing SNMP Management Frameworks", RFC 2571, April 1999.

[RFC-2572] Case, J., Harrington, D., Presuhn, R. and B. Wijnen, "Message Processing and Dispatching for the Simple Network Management Protocol (SNMP)", RFC 2572, April 1999

[RFC-2573] Levi, D., Meyer, P. and B. Stewart, "SNMP Applications", RFC 2573, April 1999.

[RFC-2574] Blumenthal, U. and B. Wijnen, "The User-Based Security Model for Version 3 of the Simple Network Management Protocol (SNMPv3)", RFC 2574, April 1999

[RFC-2575] Wijnen, B., Presuhn, R. and K. McCloghrie, "View-based Access Control Model for the Simple Network Management Protocol (SNMP)", RFC 2575, April 1999

[RFC-2576] R. Frye, D. Levi, S. Routhier, B. Wijnen, "Coexistence between Version 1, Version 2, and Version 3 of the Internet-Standard and Network Management Framework", RFC 2576, March 2000.

[RFC-2578] McCloghrie, K., Perkins, D., Schoenwaelder, J., Case, J., Rose, M. and S. Waldbusser, "Structure of Management Information Version 2 (SMIv2)", STD 58, RFC 2578, April 1999

[RFC-2579] McCloghrie, K., Perkins, D., Schoenwaelder, J., Case, J., Rose, M. and S. Waldbusser, "Textual Conventions for SMIv2", STD 58, RFC 2579, April 1999

[RFC-2580] McCloghrie, K., Perkins, D., Schoenwaelder, J., Case, J., Rose, M. and S. Waldbusser, "Conformance Statements for SMIv2", STD 58, RFC 2580, April 1999

[RFC-2669] M. St. Johns, "DOCSIS Cable Device MIB Cable Device Management Information Base for DOCSIS compliant Cable Modems and Cable Modem Termination Systems", August 1999

[RFC-2670] M. St. Johns, "Radio Frequency (RF) Interface Management Information Base for MCNS/DOCSIS compliant RF interfaces", August 1999

[RFC-2786] stjohns-snmpv3-dhkeychange-mib-01.txt, Michael C. StJohns, "Diffie-Helman USM Key MIB August 1999 Diffie-Helman USM Key Management Information Base and Textual Convention", Aug 6, 1999

[RFC-2933] McCloghrie, K., Farinacci, D., Thaler, D., “Internet Group Management Protocol MIB”, RFC 2933

[ID-IGMP] Fenner, W., IGMP-based Multicast Forwarding (``IGMP Proxying''), IETF Internet Draft, http://www.ietf.org/internet-drafts/draft-fenner-igmp-proxy-03.txt.

This page intentionally blank

Appendix Q. Acknowledgements

The following contributors deserve genuine gratitude for their efforts in the development of the OSSI 1.1 specification.

Pak Siripunkaw of MediaOne

Taft Singletary of Cox Communications

Jason Schnitzer of Shaw

Mike St. Johns of @Home

Asha Hegde of Cisco

Daniel Chuang of 3Com

Minnie Lu of Cisco

Bill Yost of TCE

Kaz Ozawa of Toshiba

Bob Himlin of TurboNet

Douglas Jones of MediaOne

Tasheer Syed of TCE

Benjamin Dolnik of 3Com

Randy Demuynck of Ericsson

Raymond Hou of Amplifynet

Fred Kiremidjian of Amplifynet

Adam Parmelee of Terayon

Dan Smith of Broadband Access Systems, Inc.

Pavaz Kokan of TCE

CableLabs and the cable industry as a whole are grateful to these individuals and organizations for their contributions. Their diligent work and professional approach should be commended and their continued enthusiam will be invaluable as the OSSI specification evolves.

Appendix R. Revisions

R.1 ECNs included in SP-OSSIv1.1-I01-000407

ECN	Date Accepted	Author
oss-n-00011	03/01/00	Pak Siripunkaw
oss-n-00027	04/19/00	Kaz Ozawa

R.2 ECNs included in SP-OSSIv1.1-I02-000714

ECN	Date Accepted	Author
oss-n-00039	05/10/00	William Yost
oss-n-00040	05/17/00	William Yost
oss-n-00041	05/17/00	Minnie Lu
oss-n-00054	06/21/00	Pak Siripunkaw

R.3 ECNs included in SP-OSSIv1.1-I03-001215

ECN	Date Accepted	Author
oss-n-00063	07/26/00	Pak Siripunkaw
oss-n-00065	08/02/00	Dan Smith
oss-n-00066	08/02/00	Pak Siripunkaw
oss-n-00067	08/09/00	Pak Siripunkaw
oss-n-00068	08/16/00	Kaz Ozawa
oss-n-00074	09/13/00	Erich Arnold
oss-n-00077	10/11/00	William H. Yost
oss-n-	11/15/00	Pak

00078		Siripunkaw
oss-n-00080	10/04/00	Dan Smith
oss-n-00081	09/27/00	David Raftus
oss-n-00087	11/08/00	Pak Siripunkaw
oss-n-00090	10/25/00	Dan Smith
oss-n-00091	11/08/00	Erich Arnold
oss-n-00094	11/08/00	Dan Smith
oss-n-00096	11/22/00	Pak Siripunkaw
oss-n-00097	11/08/00	Pak Siripunkaw
oss-n-00102	11/15/00	Bruce Braidek
oss-n-00106	11/15/00	Greg Nakanishi
oss-n-00107	11/15/00	Dan Smith
oss-n-00109	11/22/00	David Raftus
oss-n-00110	11/22/00	Pak Siripunkaw
oss-n-00111	11/22/00	Pak Siripunkaw
oss-n-00117	11/22/00	Dan Smith