PacketCable™ MTA Device Provisioning Specification

PKT-SP-PROV-I09-040402

ISSUED

Notice

Document Status Sheet

Document Control Number:	PKT-SP-PROV-I09-040402
Document Title:	PacketCable™ MTA Device Provisioning Specification
Revision History:	I01 – Released December 01, 1999 I02 – Released March 23, 2001 I03 – Released December 21, 2001 I04 – Released October 18, 2002 I05 – Released November 27, 2002 I06 – Released April 15, 2003 I07 – Released July 28, 2003 I08 – Released January 13, 2004 I09 – Released April 2, 2004
Date:	April 2, 2004
Status:	~~Work in Progress~~ ~~Draft~~ Issued ~~Closed~~
Distribution Restrictions:	~~Author Only~~ ~~CL/Member~~ ~~CL/ PacketCable/ Vendor~~ Public

Key to Document Status Codes

Work in Progress	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 reviews by Members and vendors. Drafts are susceptible to substantial change during the review process.
Issued	A stable document, which has undergone rigorous member and vendor review and is suitable for product design and development, cross-vendor interoperability, and for certification testing.
Closed	A static document, reviewed, tested, validated, and closed to further engineering change requests to the specification through CableLabs.

Contents

Figures

Tables

1 INTRODUCTION

1.1 Purpose

This specification describes the PacketCable™ 1.0 embedded-MTA device initialization and provisioning. This specification is issued to facilitate design and field-testing leading to manufacturability and interoperability of conforming hardware and software by multiple vendors.

1.2 Scope

The scope of this document is limited to the provisioning of a PacketCable 1.0 embedded-MTA device by a single provisioning and network management provider. An attempt has been made to provide enough detail to enable vendors to build an embedded-MTA device that is interoperable in a PacketCable 1.0 network configuration. This document defines the provisioning of MTA components of the embedded MTA device (unless stated otherwise).

1.3 Document Overview

This specification describes provisioning of a PacketCable 1.0 embedded-MTA. The document is structured as follows:

- Section 2 – References.
- Section 3 – Terms and Definitions.
- Section 4 – Abbreviations.
- Section 5 – Background information including a description of the provisioning reference architecture, components and interfaces.
- Section 6 – Provisioning overview including logical state transition diagram.
- Section 7 – Provisioning flows for initial power-on, post-power-on, scenarios involving updating services on an MTA endpoint, and limited failure scenarios.
- Section 8 – PacketCable requirements for DHCP [1].
- Section 9 – MTA Provisionable attributes (configuration file)
- Section 10 – List of MTA device capabilities
- Appendix I – Provisioning Events

1.4 Requirements Syntax

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

"MUST"	This word or the adjective "REQUIRED" means that the item is an absolute requirement of this specification.
"MUST NOT"	This phrase means that the item is an absolute prohibition of this specification.
"SHOULD"	This word or the adjective "RECOMMENDED" means that there may exist valid reasons in particular circumstances to ignore this item, but the full implications should be understood and the case carefully weighed before choosing a different course.
"SHOULD NOT"	This phrase means that there may exist valid reasons in particular circumstances when the listed behavior is acceptable or even useful, but the full implications should be

understood and the case carefully weighed before implementing any behavior described with this label.

"MAY" This word or the adjective "OPTIONAL" means that this item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because it enhances the product, for example; another vendor may omit the same item.

Other text is descriptive or explanatory.

2 REFERENCES

2.1 Normative References

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

[1] IETF RFC 2131, DHCP: Dynamic Host Configuration Protocol, March 1997.

[2] PacketCable MTA MIB, PKT-SP-MIB-MTA-I09-040402, April 2, 2004, Cable Television Laboratories, Inc., http://www.packetcable.com./

[3] PacketCable Signaling MIB, PKT-SP-MIB-SIG-I08-040113, January 13, 2004, Cable Television Laboratories, Inc., http://www.packetcable.com./

[4] PacketCable Network-Based Call Signaling Protocol Specification, PKT-SP-EC-MGCP-I10-040402, April 2, 2004, Cable Television Laboratories, Inc., http://www.packetcable.com./

[5] PacketCable Security Specification, PKT-SP-SEC-I10-040113, January 13, 2004, Cable Television Laboratories, Inc., http://www.packetcable.com./

[6] Data-Over-Cable Service Interface Specification, Radio Frequency Interface Specification. SP-RFIv1.1-I10-030730, July 30, 2003, Cable Television Laboratories, Inc. http://www.cablemodem.com/

[7] IETF RFC 3413, Simple Network Management Protocol (SNMP) Applications, December 2002.

[8] IETF RFC 3414, User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3), December 2002.

[9] IETF RFC 3415, View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP), December 2002.

[10] IETF RFC 3396, Encoding Long Options in the Dynamic Host Configuration Protocol (DHCPv4), November 2002.

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

2.2 Informative References

[12] IETF RFC 3442, The Classless Static Route Option for Dynamic Host Configuration Protocol (DHCP) version 4, December 2002.

[13] IETF RFC 1340, ASSIGNED NUMBERS, (contains ARP/DHCP parameters), July 1992.

[14] IETF RFC 1350, The TFTP Protocol (Revision 2), STD 33, MIT, July 1992.

[15] IETF RFC 1034, Domain Names—Concepts and Facilities, STD 13, November 1987.

[16] IETF RFC 1035, Domain Names—Implementation and Specifications, November 1987.

[17] IETF RFC 1591, Domain Name System Structure and Delegation, March 1994.

[18] IETF RFC 3495, DHCP Option for CableLabs Client Configuration, March 2003.

[19] PacketCable Architecture Framework Technical Report, PKT-TR-ARCH-I01-991201, Cable Television Laboratories, Inc., December 1, 1999, http://www.packetcable.com./

[20] Data-Over-Cable Service Interface Specifications, Cable Modem to Customer Premise Equipment Interface Specification, CMCI, DOCSIS SP-CMCI-I10-030730, July 30, 2003, Cable Television Laboratories, Inc., http://www.cablemodem.com/

[21] IETF RFC 3417, Transport Mappings for the Simple Network Management Protocol (SNMP), December 2002.

[22] IETF RFC 2579, Textual Conventions for SMIv2, April 1999.

[23] IETF RFC 3410, Introduction and Applicability Statements for Internet-Standard Management Framework, December 2002.

[24] IETF RFC 3411, An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks, December 2002.

[25] IETF RFC 3412, Message Processing and Dispatching for the Simple Network Management Protocol (SNMP), December 2002.

[26] Data-Over-Cable Service Interface Specifications, Operations Support System Interface Specification Radio Frequency Interface SP-OSSIv1.1-I07-030730, July 30, 2003, Cable Television Laboratories, Inc., http://www.cablemodem.com/

[27] IETF RFC 2821, Simple Mail Transfer Protocol, April 2001.

[28] IETF RFC 1123, Braden, R., Requirements for Internet Hosts -- Application and Support, October 1989.

[29] IETF RFC 2349, TFTP Timeout Interval and Transfer Size Options, May 1998.

[30] IETF RFC 1945, IETF RFC-2068, HTTP 1.0 and 1.1, May 1996.

[31] IETF RFC 2475, An Architecture for Differentiated Services, December 1998.

[32] PacketCable Audio/Video Codecs Specification, PKT-SP-CODEC-I04-021018, Cable Television Laboratories, Inc., October 18, 2002. http://www.packetcable.com./

[33] PacketCable Dynamic Quality of Service Specification, PKT-SP-DQOS-I08-030815, August 15, 2003, Cable Television Laboratories, Inc., http://www.packetcable.com./

[34] IETF RFC 3594, PacketCable Security Ticket Control Sub-Option for the DHCP CableLabs Client Configuration (CCC) Option, September 2003.

[35] IETF RFC 2782, A DNS RR for specifying the location of services (DNS SRV), February 2000.

[36] PacketCable MIBs Framework, PKT-SP-MIBS-I08-040113, January 13, 2004, Cable Television Laboratories, Inc., http://www.packetcable.com/

[37] IETF RFC 2616, Hypertext Transfer Protocol -- HTTP/1.1

[38] IETF RFC 3617, Uniform Resource Identifier (URI) Scheme and Applicability Statement for the Trivial File Transfer Protocol (TFTP)

3 TERMS AND DEFINITIONS

PacketCable specifications use the following terms:

Access Control	Limiting the flow of information from the resources of a system only to authorized persons, programs, processes, or other system resources on a network.
Active	A service flow is said to be "active" when it is permitted to forward data packets. A service flow must first be admitted before it is active.
Admitted	A service flow is said to be "admitted" when the CMTS has reserved resources (e.g., bandwidth) for it on the DOCSIS™ network.
A-link	A-Links are SS7 links that interconnect STPs and either SSPs or SCPs. 'A' stands for "Access."
Asymmetric Key	An encryption key or a decryption key used in public key cryptography, where encryption and decryption keys are always distinct.
Audio Server	An Audio Server plays informational announcements in PacketCable network. Media announcements are needed for communications that do not complete and to provide enhanced information services to the user. The component parts of Audio Server services are Media Players and Media Player Controllers.
Authentication	The process of verifying the claimed identity of an entity to another entity.
Authenticity	The ability to ensure that the given information is without modification or forgery and was in fact produced by the entity that claims to have given the information.
Authorization	The act of giving access to a service or device if one has permission to have the access.
Cipher	An algorithm that transforms data between plaintext and ciphertext.
Ciphersuite	A set which must contain both an encryption algorithm and a message authentication algorithm (e.g., a MAC or an HMAC). In general, it may also contain a key-management algorithm, which does not apply in the context of PacketCable.
Ciphertext	The (encrypted) message output from a cryptographic algorithm that is in a format that is unintelligible.
Cleartext	The original (unencrypted) state of a message or data. Also called plaintext.
Confidentiality	A way to ensure that information is not disclosed to anyone other then the intended parties. Information is encrypted to provide confidentiality. Also known as privacy.
Cryptanalysis	The process of recovering the plaintext of a message or the encryption key without access to the key.
Cryptographic algorithm	An algorithm used to transfer text between plaintext and ciphertext.
Decipherment	A procedure applied to ciphertext to translate it into plaintext.
Decryption	A procedure applied to ciphertext to translate it into plaintext.
Decryption key	The key in the cryptographic algorithm to translate the ciphertext to plaintext.

Digital certificate	A binding between an entity's public key and one or more attributes relating to its identity, also known as a public key certificate.
Digital signature	A data value generated by a public-key algorithm based on the contents of a block of data and a private key, yielding an individualized cryptographic checksum.
Downstream	The direction from the headend toward the subscriber location.
Encipherment	A method used to translate plaintext into ciphertext.
Encryption	A method used to translate plaintext into ciphertext.
Encryption Key	The key used in a cryptographic algorithm to translate the plaintext to ciphertext.
Endpoint	A Terminal, Gateway or Multipoint Conference Unit (MCU).
Errored Second	Any 1-second interval containing at least one bit error.
Event Message	A message capturing a single portion of a connection.
F-link	F-Links are SS7 links that directly connect two SS7 end points, such as two SSPs. 'F' stands for "Fully Associated."
Flow [DOCSIS Flow]	(a.k.a. DOCSIS-QoS "service flow") A unidirectional sequence of packets associated with a Service ID (SID) and a QoS. Multiple multimedia streams may be carried in a single DOCSIS Flow.
Flow [IP Flow]	A unidirectional sequence of packets identified by OSI Layer 3 and Layer 4 header information. This information includes source/destination IP addresses, source/destination port numbers, protocol ID. Multiple multimedia streams may be carried in a single IP Flow.
Gateway	Devices bridging between the PacketCable IP Voice Communication world and the PSTN. Examples are the Media Gateway, which provides the bearer circuit interfaces to the PSTN and transcodes the media stream, and the Signaling Gateway, which sends and receives circuit switched network signaling to the edge of the PacketCable network.
H.323	An ITU-T recommendation for transmitting and controlling audio and video information. The H.323 recommendation requires the use of the ITU-T H.225 and ITU-T H.245 protocol for communication control between a "gateway" audio/video endpoint and a "gatekeeper" function.
Header	Protocol control information located at the beginning of a protocol data unit.
Integrity	A way to ensure that information is not modified except by those who are authorized to do so.
IntraLATA	Within a Local Access Transport Area.
Jitter	Variability in the delay of a stream of incoming packets making up a flow such as a voice communication.
Kerberos	A secret-key network authentication protocol that uses a choice of cryptographic algorithms for encryption and a centralized key database for authentication.
Key	A mathematical value input into the selected cryptographic algorithm.
Key Exchange	The swapping of public keys between entities to be used to encrypt communication between the entities.
Key Management	The process of distributing shared symmetric keys needed to run a security protocol.

Key Pair	An associated public and private key where the correspondence between the two are mathematically related, but it is computationally infeasible to derive the private key from the public key.
Keying Material	A set of cryptographic keys and their associated parameters, normally associated with a particular run of a security protocol.
Keyspace	The range of all possible values of the key for a particular cryptographic algorithm.
Latency	The time, expressed in quantity of symbols, taken for a signal element to pass through a device.
Link Encryption	Cryptography applied to data as it travels on data links between the network devices.
Network Layer	Layer 3 in the Open System Interconnection (OSI) architecture that provides network information that is independent from the lower layers.
Network Management	The functions related to the management of data across the network.
Network Management OSS	The functions related to the management of data link layer and physical layer resources and their stations across the data network supported by the hybrid fiber/coax system.
Nonce	A random value used only once that is sent in a communications protocol exchange to prevent replay attacks.
Non-Repudiation	The ability to prevent a sender from denying later that he or she sent a message or performed an action.
Off-Net Call	A communication connecting a PacketCable subscriber out to a user on the PSTN.
On-Net Call	A communication placed by one customer to another customer entirely on the PacketCable Network.
One-way Hash	A hash function that has an insignificant number of collisions upon output.
Plaintext	The original (unencrypted) state of a message or data. Also called cleartext.
Pre-shared Key	A shared secret key passed to both parties in a communication flow, using an unspecified manual or out-of-band mechanism.
Privacy	A way to ensure that information is not disclosed to any one other then the intended parties. Information is usually encrypted to provide confidentiality. Also known as confidentiality.
Private Key	The key used in public key cryptography that belongs to an individual entity and must be kept secret.
Proxy	A facility that indirectly provides some service or acts as a representative in delivering information, thereby eliminating the need for a host to support the service.
Public Key	The key used in public key cryptography that belongs to an individual entity and is distributed publicly. Other entities use this key to encrypt data to be sent to the owner of the key.
Public Key Certificate	A binding between an entity's public key and one or more attributes relating to its identity, also known as a digital certificate.

Public Key Cryptography	A procedure that uses a pair of keys, a public key and a private key, for encryption and decryption, also known as an asymmetric algorithm. A user's public key is publicly available for others to use to send a message to the owner of the key. A user's private key is kept secret and is the only key that can decrypt messages sent encrypted by the user's public key.
Root Private Key	The private signing key of the highest-level Certification Authority. It is normally used to sign public key certificates for lower-level Certification Authorities or other entities.
Root Public Key	The public key of the highest level Certification Authority, normally used to verify digital signatures generated with the corresponding root private key.
Secret Key	The cryptographic key used in a symmetric key algorithm, which results in the secrecy of the encrypted data depending solely upon keeping the key a secret, also known as a symmetric key.
Session Key	A cryptographic key intended to encrypt data for a limited period of time, typically between a pair of entities.
Signed and Sealed	An "envelope" of information which has been signed with a digital signature and sealed using encryption.
Subflow	A unidirectional flow of IP packets characterized by a single source and destination IP address and single source and destination UDP/TCP port.
Symmetric Key	The cryptographic key used in a symmetric key algorithm, which results in the secrecy of the encrypted data depending solely upon keeping the key a secret, also known as a secret key.
Systems Management	Functions in the application layer related to the management of various Open Systems Interconnection (OSI) resources and their status across all layers of the OSI architecture.
Transit Delays	The time difference between the instant at which the first bit of a Protocol Data Unit (PDU) crosses one designated boundary, and the instant at which the last bit of the same PDU crosses a second designated boundary.
Trunk	An analog or digital connection from a circuit switch that carries user media content and may carry voice signaling (M_F, R_2, etc.).
Tunnel Mode	An IPsec (ESP or AH) mode that is applied to an IP tunnel, where an outer IP packet header (of an intermediate destination) is added on top of the original, inner IP header. In this case, the ESP or AH transform treats the inner IP header as if it were part of the packet payload. When the packet reaches the intermediate destination, the tunnel terminates and both the outer IP packet header and the IPsec ESP or AH transform are taken out.
Upstream	The direction from the subscriber location toward the headend.
X.509 certificate	A public key certificate specification developed as part of the ITU-T X.500 standards directory.

4 ABBREVIATIONS

PacketCable specifications use the following abbreviations.

AAA	Authentication, Authorization and Accounting.
AES	Advanced Encryption Standard. A block cipher, used to encrypt the media traffic in PacketCable.
AF	Assured Forwarding. This is a DiffServ Per Hop Behavior.
AH	Authentication header. An IPsec security protocol that provides message integrity for complete IP packets, including the IP header.
AMA	Automated Message Accounting. A standard form of call detail records (CDRs) developed and administered by Bellcore (now Telcordia Technologies).
ASD	Application-Specific Data. A field in some Kerberos key management messages that carries information specific to the security protocol for which the keys are being negotiated.
AT	Access Tandem.
ATM	Asynchronous Transfer Mode. A protocol for the transmission of a variety of digital signals using uniform 53-byte cells.
BAF	Bellcore AMA Format, also known as AMA.
BCID	Billing Correlation ID.
BPI+	Baseline Privacy Plus Interface Specification. The security portion of the DOCSIS 1.1 standard that runs on the MAC layer.
CA	Certification Authority. A trusted organization that accepts certificate applications from entities, authenticates applications, issues certificates and maintains status information about certificates.
CA	Call Agent. The part of the CMS that maintains the communication state, and controls the line side of the communication.
CBC	Cipher Block Chaining mode. An option in block ciphers that combine (XOR) the previous block of ciphertext with the current block of plaintext before encrypting that block of the message.
CBR	Constant Bit Rate.
CDR	Call Detail Record. A single CDR is generated at the end of each billable activity. A single billable activity may also generate multiple CDRs.
CIC	Circuit Identification Code. In ANSI SS7, a two-octet number that uniquely identifies a DSO circuit within the scope of a single SS7 Point Code.
CID	Circuit ID (Pronounced “kid”). This uniquely identifies an ISUP DS0 circuit on a Media Gateway. It is a combination of the circuit’s SS7 gateway point code and Circuit Identification Code (CIC). The SS7 DPC is associated with the Signaling Gateway that has domain over the circuit in question.
CIF	Common Intermediate Format.
CIR	Committed Information Rate.
CM	DOCSIS Cable Modem.
CMS	Cryptographic Message Syntax.

CMS	Call Management Server. Controls the audio connections. Also called a Call Agent in MGCP/SGCP terminology. This is one example of an Application Server.
CMTS	Cable Modem Termination System. The device at a cable headend which implements the DOCSIS RFI MAC protocol and connects to CMs over an HFC network.
CMSS	Call Management Server Signaling.
Codec	COder-DECoder.
COPS	Common Open Policy Service protocol. Currently an internet draft, which describes a client/server model for supporting policy control over QoS Signaling Protocols and provisioned QoS resource management.
CoS	Class of Service. The type 4 tuple of a DOCSIS configuration file.
CRCX	Create Connection.
CSR	Customer Service Representative.
DA	Directory Assistance.
DE	Default. This is a DiffServ Per Hop Behavior.
DES	Data Encryption Standard.
DF	Delivery Function.
DHCP	Dynamic Host Configuration Protocol.
DHCP-D	DHCP Default. Network Provider DHCP Server.
DNS	Domain Name Service.
DOCSIS™	Data-Over-Cable Service Interface Specifications.
DPC	Destination Point Code. In ANSI SS7, a 3-octet number which uniquely identifies an SS7 Signaling Point, either an SSP, STP, or SCP.
DQoS	Dynamic Quality-of-Service. Assigned on the fly for each communication depending on the QoS requested.
DSA	Dynamic Service Add.
DSC	Dynamic Service Change.
DSCP	DiffServ Code Point. A field in every IP packet that identifies the DiffServ Per Hop Behavior. In IP version 4, the TOS byte is redefined to be the DSCP. In IP version 6, the Traffic Class octet is used as the DSCP.
DTMF	Dual-tone Multi Frequency (tones).
EF	Expedited Forwarding. A DiffServ Per Hop Behavior.
E-MTA	Embedded MTA. A single node that contains both an MTA and a cable modem.
EO	End Office.
ESP	IPsec Encapsulating Security Payload. Protocol that provides both IP packet encryption and optional message integrity, not covering the IP packet header.
ETSI	European Telecommunications Standards Institute.
F-link	F-Links are SS7 links that directly connect two SS7 end points, such as two SSPs. 'F' stands for "Fully Associated."
FEID	Financial Entity ID.
FGD	Feature Group D signaling.
FQDN	Fully Qualified Domain Name. Refer to IETF RFC 2821 for details.
GC	Gate Controller.

GTT	Global Title Translation.
HFC	Hybrid Fiber/Coaxial. An HFC system is a broadband bi-directional shared media transmission system using fiber trunks between the headend and the fiber nodes, and coaxial distribution from the fiber nodes to the customer locations.
HMAC	Hashed Message Authentication Code. A message authentication algorithm, based on either SHA-1 or MD5 hash and defined in IETF RFC 2104.
HTTP	Hypertext Transfer Protocol. Refer to IETF RFC 1945 and RFC 2068.
IANA	Internet Assigned Numbered Authority. See www.ietf.org for details.
IC	Inter-exchange Carrier.
IETF	Internet Engineering Task Force. A body responsible, among other things, for developing standards used on the Internet. See www.ietf.org for details.
IKE	Internet Key Exchange. A key-management mechanism used to negotiate and derive keys for SAs in IPsec.
IKE–	A notation defined to refer to the use of IKE with pre-shared keys for authentication.
IKE+	A notation defined to refer to the use of IKE with X.509 certificates for authentication.
IP	Internet Protocol. An Internet network-layer protocol.
IPsec	Internet Protocol Security. A collection of Internet standards for protecting IP packets with encryption and authentication.
ISDN	Integrated Services Digital Network.
ISTP	Internet Signaling Transport Protocol.
ISUP	ISDN User Part. A protocol within the SS7 suite of protocols that is used for call signaling within an SS7 network.
ITU	International Telecommunication Union.
ITU-T	International Telecommunication Union–Telecommunication Standardization Sector.
IVR	Interactive Voice Response system.
KDC	Key Distribution Center.
LATA	Local Access and Transport Area.
LD	Long Distance.
LIDB	Line Information Database. Contains customer information required for real-time access such as calling card personal identification numbers (PINs) for real-time validation.
LLC	Logical Link Control. The Ethernet Packet header and optional 802.1P tag which may encapsulate an IP packet. A sublayer of the Data Link Layer.
LNP	Local Number Portability. Allows a customer to retain the same number when switching from one local service provider to another.
LSSGR	LATA Switching Systems Generic Requirements.
MAC	Message Authentication Code. A fixed-length data item that is sent together with a message to ensure integrity, also known as a MIC.
MAC	Media Access Control. It is a sublayer of the Data Link Layer. It normally runs directly over the physical layer.
MC	Multipoint Controller.
MCU	Multipoint Conferencing Unit.

MD5	Message Digest 5. A one-way hash algorithm that maps variable length plaintext into fixed-length (16 byte) ciphertext.
MDCP	Media Device Control Protocol. A media gateway control specification submitted to IETF by Lucent. Now called SCTP.
MDCX	Modify Connection.
MDU	Multi-Dwelling Unit. Multiple units within the same physical building. The term is usually associated with high-rise buildings.
MEGACO	Media Gateway Control IETF working group. See www.ietf.org for details.
MF	Multi-Frequency.
MG	Media Gateway. Provides the bearer circuit interfaces to the PSTN and transcodes the media stream.
MGC	Media Gateway Controller. The overall controller function of the PSTN gateway. Receives, controls and mediates call-signaling information between the PacketCable and PSTN.
MGCP	Media Gateway Control Protocol. Protocol follow-on to SGCP. Refer to IETF 2705.
MIB	Management Information Base.
MIC	Message Integrity Code. A fixed-length data item that is sent together with a message to ensure integrity, also known as a Message Authentication Code (MAC).
MMC	Multi-Point Mixing Controller. A conferencing device for mixing media streams of multiple connections.
MSB	Most Significant Bit.
MSO	Multi-System Operator. A cable company that operates many headend locations in several cities.
MSU	Message Signal Unit.
MTA	Multimedia Terminal Adapter. Contains the interface to a physical voice device, a network interface, CODECs, and all signaling and encapsulation functions required for VoIP transport, class features signaling, and QoS signaling.
MTP	The Message Transfer Part. A set of two protocols (MTP 2, MTP 3) within the SS7 suite of protocols that are used to implement physical, data link, and network-level transport facilities within an SS7 network.
MWD	Maximum Waiting Delay.
NANP	North American Numbering Plan.
NANPNAT	North American Numbering Plan Network Address Translation.
NAT Network Layer	Network Address Translation. Layer 3 in the Open System Interconnection (OSI) architecture. This layer provides services to establish a path between open systems.
NCS	Network Call Signaling.
NPA-NXX	Numbering Plan Area (more commonly known as area code) NXX (sometimes called exchange) represents the next three numbers of a traditional phone number. The N can be any number from 2-9 and the Xs can be any number. The combination of a phone number's NPA-NXX will usually indicate the physical location of the call device. The exceptions include toll-free numbers and ported number (see LNP).
NTP	Network Time Protocol. An internet standard used for synchronizing clocks of elements distributed on an IP network.

NTSC	National Television Standards Committee. Defines the analog color television broadcast standard used today in North America.
OID	Object Identification.
OSP	Operator Service Provider.
OSS	Operations Systems Support. The back-office software used for configuration, performance, fault, accounting, and security management.
OSS-D	OSS Default. Network Provider Provisioning Server.
PAL	Phase Alternate Line. The European color television format that evolved from the American NTSC standard.
PCES	PacketCable Electronic Surveillance.
PCM	Pulse Code Modulation. A commonly employed algorithm to digitize an analog signal (such as a human voice) into a digital bit stream using simple analog-to-digital conversion techniques.
PDU	Protocol Data Unit.
PHS	Payload Header Suppression. A DOCSIS technique for compressing the Ethernet, IP, and UDP headers of RTP packets.
PKCROSS	Public-Key Cryptography for Cross-Realm Authentication. Utilizes PKINIT for establishing the inter-realm keys and associated inter-realm policies to be applied in issuing cross-realm service tickets between realms and domains in support of Intradomain and Interdomain CMS-to-CMS signaling (CMSS).
PKCS	Public-Key Cryptography Standards. Published by RSA Data Security Inc. These Standards describe how to use public key cryptography in a reliable, secure and interoperable way.
PKI	Public-Key Infrastructure. A process for issuing public key certificates, which includes standards, Certification Authorities, communication between authorities and protocols for managing certification processes.
PKINIT	Public-Key Cryptography for Initial Authentication. The extension to the Kerberos protocol that provides a method for using public-key cryptography during initial authentication.
PSC	Payload Service Class Table, a MIB table that maps RTP payload Type to a Service Class Name.
PSFR	Provisioned Service Flow Reference. An SFR that appears in the DOCSIS configuration file.
PSTN	Public Switched Telephone Network.
QCIF	Quarter Common Intermediate Format.
QoS	Quality of Service. Guarantees network bandwidth and availability for applications.
RADIUS	Remote Authentication Dial-In User Service. An internet protocol (IETF RFC 2865 and RFC 2866) originally designed for allowing users dial-in access to the internet through remote servers. Its flexible design has allowed it to be extended well beyond its original intended use.
RAS	Registration, Admissions and Status. RAS Channel is an unreliable channel used to convey the RAS messages and bandwidth changes between two H.323 entities.
RC4	Rivest Cipher 4. A variable length stream cipher. Optionally used to encrypt the media traffic in PacketCable.
RFC	Request for Comments. Technical policy documents approved by the IETF which are available on the World Wide Web at http://www.ietf.cnri.reston.va.us/rfc.html.

RFI	The DOCSIS Radio Frequency Interface specification.
RJ-11	Registered Jack-11. A standard 4-pin modular connector commonly used in the United States for connecting a phone unit into a wall jack.
RKS	Record Keeping Server. The device, which collects and correlates the various Event Messages.
RSA	A public-key, or asymmetric, cryptographic algorithm used to provide authentication and encryption services. RSA stands for the three inventors of the algorithm; Rivest, Shamir, Adleman.
RSA Key Pair	A public/private key pair created for use with the RSA cryptographic algorithm.
RSVP	Resource Reservation Protocol.
RTCP	Real-Time Control Protocol.
RTO	Retransmission Timeout.
RTP	Real-time Transport Protocol. A protocol for encapsulating encoded voice and video streams. Refer to IETF RFC 1889.
SA	Security Association. A one-way relationship between sender and receiver offering security services on the communication flow.
SAID	Security Association Identifier. Uniquely identifies SAs in the DOCSIS Baseline Privacy Plus Interface (BPI+) security protocol.
SCCP	Signaling Connection Control Part. A protocol within the SS7 suite of protocols that provides two functions in addition to those provided within MTP. The first function is the ability to address applications within a signaling point. The second function is Global Title Translation.
SCP	Service Control Point. A Signaling Point within the SS7 network, identifiable by a Destination Point Code that provides database services to the network.
SCTP	Stream Control Transmission Protocol.
SDP	Session Description Protocol.
SDU	Service Data Unit. Information delivered as a unit between peer service access points.
SF	Service Flow. A unidirectional flow of packets on the RF interface of a DOCSIS system.
SFID	Service Flow ID. A 32-bit integer assigned by the CMTS to each DOCSIS Service Flow defined within a DOCSIS RF MAC domain. SFIDs are considered to be in either the upstream direction (USFID) or downstream direction (DSFID). Upstream Service Flow IDs and Downstream Service Flow IDs are allocated from the same SFID number space.
SFR	Service Flow Reference. A 16-bit message element used within the DOCSIS TLV parameters of Configuration Files and Dynamic Service messages to temporarily identify a defined Service Flow. The CMTS assigns a permanent SFID to each SFR of a message.
SG	Signaling Gateway. An SG is a signaling agent that receives/sends SCN native signaling at the edge of the IP network. In particular, the SS7 SG function translates variants ISUP and TCAP in an SS7-Internet Gateway to a common version of ISUP and TCAP.
SGCP	Simple Gateway Control Protocol. Earlier draft of MGCP.
SHA – 1	Secure Hash Algorithm 1. A one-way hash algorithm.

SID	Service ID. A 14-bit number assigned by a CMTS to identify an upstream virtual circuit. Each SID separately requests and is granted the right to use upstream bandwidth.
SIP	Session Initiation Protocol. An application-layer control (signaling) protocol for creating, modifying, and terminating sessions with one or more participants.
SIP+	Session Initiation Protocol Plus. An extension to SIP.
S-MTA	Standalone MTA. A single node that contains an MTA and a non-DOCSIS MAC (e.g., ethernet).
SNMP	Simple Network Management Protocol.
SOHO	Small Office/Home Office.
SS7	Signaling System number 7. An architecture and set of protocols for performing out-of-band call signaling with a telephone network.
SSP	Service Switching Point. SSPs are points within the SS7 network that terminate SS7 signaling links and also originate, terminate, or tandem switch calls.
STP	Signal Transfer Point. A node within an SS7 network that routes signaling messages based on their destination address. This is essentially a packet switch for SS7. It may also perform additional routing services such as Global Title Translation.
TCAP	Transaction Capabilities Application Protocol. A protocol within the SS7 stack that is used for performing remote database transactions with a Signaling Control Point.
TCP	Transmission Control Protocol.
TD	Timeout for Disconnect.
TFTP	Trivial File Transfer Protocol.
TFTP-D	Default – Trivial File Transfer Protocol.
TGS	Ticket Granting Server. A sub-system of the KDC used to grant Kerberos tickets.
TGW	Telephony Gateway.
TIPHON	Telecommunications and Internet Protocol Harmonization Over Network.
TLV	Type-Length-Value. A tuple within a DOCSIS configuration file.
TN	Telephone Number.
ToD	Time-of-Day Server.
TOS	Type of Service. An 8-bit field of every IP version 4 packet. In a DiffServ domain, the TOS byte is treated as the DiffServ Code Point, or DSCP.
TSG	Trunk Subgroup.
UDP	User Datagram Protocol. A connectionless protocol built upon Internet Protocol (IP).
VAD	Voice Activity Detection.
VBR	Variable Bit Rate.
VoIP	Voice-over-IP.

5 BACKGROUND

5.1 Service Goals

Cable operators are interested in deploying high-speed data communications systems on cable television systems. Cable operators and Cable Television Laboratories, Inc. (on behalf of the CableLabs® member companies), have prepared a series of interface specifications that will permit the early definition, design, development, and deployment of packet data over cable systems on an uniform, consistent, open, non-proprietary, multi-vendor interoperable basis. The intended service enables voice communications, video, and data services based on bi-directional transfer of Internet protocol (IP) traffic, between the cable system headend and customer locations, over an all-coaxial or hybrid-fiber/coax (HFC) cable network, defined by the data over cable service interface specification (DOCSIS) standard [6]. This is shown in simplified form in Figure 1.

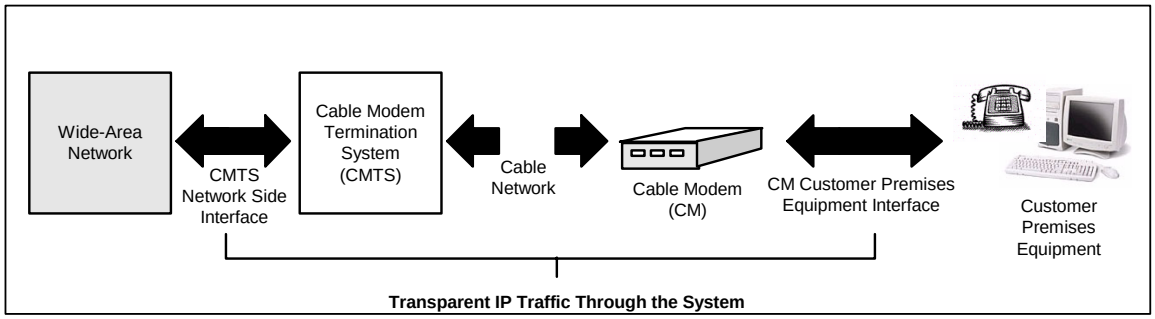


Figure 1. Transparent IP Traffic Through the Data-Over-Cable System

The transmission path over the cable system is realized at the headend by a cable modem termination system (CMTS), and at each customer location by a cable modem (CM). The intent is for operators to transfer IP traffic transparently between these interfaces.

The legal/regulatory classification of IP-based voice communications provided over cable networks and otherwise, and the legal/regulatory obligations, if any, borne by providers of such voice communications, are not yet fully defined by appropriate legal and regulatory authorities. Nothing in this specification is addressed to, or intended to affect, those issues. In particular, while this document uses standard terms such as “call,” “call signaling,” “telephony,” etc., it will be evident from this document that while a PacketCable network performs activities analogous to these PSTN functions, the manner by which it does so differs considerably from the manner in which they are performed in the PSTN by telecommunications carriers. These differences may be significant for legal/regulatory purposes.

5.2 Specification Goals

The goal of this specification document is to meet and to satisfy cable member companies (a.k.a. MSO), PacketCable, and CableLabs business and technical requirements.

Requirements relevant to device provisioning are:

- A single physical device (e.g., embedded-MTA) will be completely provisioned and managed by a single business entity. This provider may establish business relationships with additional providers for services such as data, voice communications, and other services.

- An embedded-MTA is a PacketCable 1.0 MTA combined with a DOCSIS 1.1 or DOCSIS 2.0 Cable Modem. Both DOCSIS 1.1 or DOCSIS 2.0 and PacketCable 1.0 device provisioning steps MUST be performed for this embedded-MTA device to be provisioned. The embedded-MTA MUST have two IP addresses; an IP address for the CM component, and a different IP address for the MTA component. The embedded-MTA MUST have two MAC addresses, one MAC address for the CM component, and a different MAC address for the MTA-component. Furthermore, the MTA MUST work in two environments where the MTA IP address in the same or different subnet as the CM.
- PacketCable requires a unique FQDN for the MTA-component in the embedded-MTA. This FQDN MUST be included in the DHCP offer to the MTA-component. PacketCable makes no additional FQDN requirements on the CM component in the embedded-MTA beyond those required by DOCSIS 1.1. Mapping of the FQDN to IP address MUST be configured in the network DNS server and be available to the rest of the network.
- PacketCable 1.0 embedded-MTA provisioning MUST use DHCP Option-12 and Option-15 to deliver the MTA FQDN to the E-MTA.
- PacketCable 1.0 embedded-MTA provisioning MUST support two separate configuration files, a DOCSIS-specified configuration file for the CM component, and a PacketCable-specified configuration file for the MTA component.
- The embedded-MTA is outside the PacketCable network trust boundary as defined in the PacketCable architecture document [19].
- PacketCable 1.0 MUST support DOCSIS 1.1 or DOCSIS 2.0 software download as defined in [6]. The DOCSIS 1.1 or DOCSIS 2.0 software download process supports the downloading of a single file to the cable modem or embedded MTA. A single DOCSIS 1.1 or DOCSIS 2.0 software download MUST be used to upgrade code for both DOCSIS and PacketCable software functions.
- PacketCable 1.0 MUST support use of SNMPv2c co-existence for network management operations for devices provisioned under the Basic Flow or the Hybrid Flow and SNMPv3/v2 co-existence for network management operations when the device is provisioned under the Secure Flow.
- PacketCable 1.0 embedded-MTA provisioning minimizes the impact to DOCSIS 1.1 or DOCSIS 2.0 devices (CM and CMTS) in the network.
- Standard server solutions (TFTP, SNMP, DNS, etc.) are preferable. It is understood that an application layer may be required on top of these protocols to coordinate PacketCable 1.0 embedded-MTA provisioning.
- Where appropriate, the DOCSIS 1.1 or DOCSIS 2.0 management protocols are supported (SNMP, DHCP, TFTP).

For the remainder of this specification, wherever we use the word DOCSIS or DOCSIS 1.1 it must be read as DOCSIS 1.1 or DOCSIS 2.0.

5.3 PacketCable Reference Architecture

Figure 2 shows the reference architecture for the PacketCable 1.0 Network. Refer to the PacketCable Architecture Document [19] for more detailed information on this reference architecture.

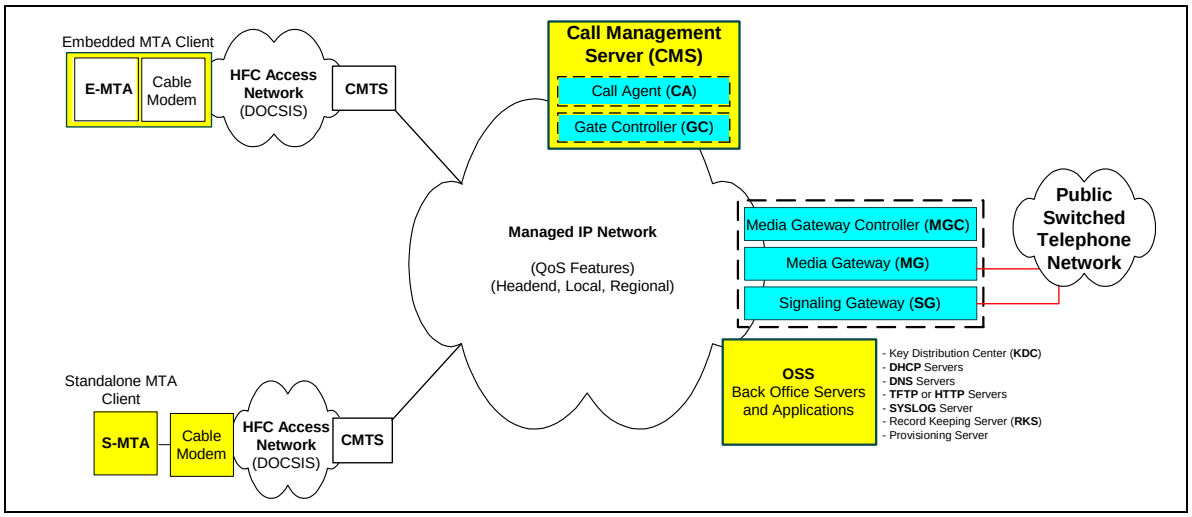


Figure 2. PacketCable 1.0 Network Component Reference Model

5.4 Components and Interfaces

This interface identifies specific requirements in the DHCP server and the client for IP assignment during the MTA initialization process. This figure represents the components and interfaces discussed in this document. All the PacketCable specifications have a similar diagram indicating which interfaces of the PacketCable Architecture are affected by a particular specification.

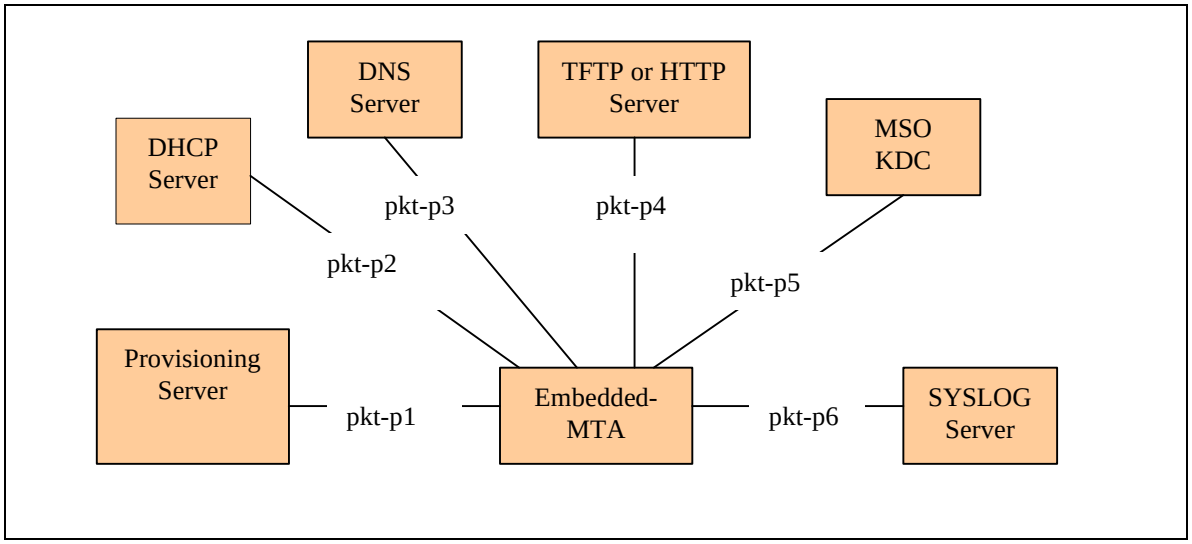


Figure 3. PacketCable Provisioning Interfaces

5.4.1 MTA

The MTA MUST conform to the following requirements during the provisioning sequence.

5.4.1.1 MTA Security Requirements

The MTA MUST conform to the following security requirements during the Secure Flow provisioning sequence:

- The MTA device MIB is structured to represent the assignment of an MTA endpoint to a CMS. However, the security association between an MTA and a CMS is on a per-endpoint basis, unless all endpoints are configured to same CMS.
- CMS Kerberos Principal Name is not explicitly configured in the MTA endpoints. The MTA MUST be able to determine the CMS Kerberos Principal Name based on the CMS FQDN, as specified in [5].
- For each unique pair of CMS Kerberos principal Name / Kerberos Realm assigned to an endpoint, the MTA MUST obtain a single Kerberos ticket [5]. If the MTA already has a valid Kerberos ticket for that CMS, the MTA MUST NOT request an additional Kerberos ticket for that CMS. (Unless the expiration time of the current Kerberos ticket <= current time + PKINIT Grace Period, in which case the MTA MUST obtain a fresh ticket for the same CMS.)
- In the case that a CMS FQDN maps to multiple IP addresses, the MTA MUST initially establish a pair of IPSEC Security Associations with one of the IP addresses returned by the DNS server. The MTA MAY also initially establish IPSEC Security Associations with the additional CMS IP addresses. Please refer to [5] for more information.
- If the MTA already has a pair of active Security Associations (inbound and outbound) with a particular CMS IP address, the MTA MUST NOT attempt to establish additional Security Associations with the same IP address.

During the provisioning sequence, there are no specific security requirements for the Basic Flow or the Hybrid Flow.

5.4.1.2 MTA SNMP Requirements

The MTA MUST conform to the following SNMPv3 requirements during the Secure Flow provisioning sequence:

- MTA SNMPv3 security is separate and distinct from DOCSIS SNMPv3 security. USM security information (authentication and privacy keys, and other USM table entries) is setup separately.
- SNMPv3 initialization MUST be completed prior to the provisioning enrollment inform.
- In Secure Flow, the MTA MUST support SNMPv3 based device management as defined in [30].

The MTA MUST conform to the following SNMPv2c requirements during the Hybrid Flow or Basic Flow provisioning sequence:

- The MTA SNMPv2c agent MUST be separate and distinct from the DOCSIS SNMPv2c agent.
- SNMPv2c initialization MUST be completed immediately after the DHCP phase.

In the Hybrid and Basic Flows, the MTA MUST support SNMPv2c based device management as defined in [30].

5.4.2 Provisioning Server

The Provisioning Server is made up of the following components:

- Provisioning Application - The Provisioning Application is responsible for coordinating the embedded-MTA provisioning process. This application has an associated SNMP Entity.

- Provisioning SNMP Entity – The provisioning SNMP entity MUST include a trap/inform handler for provisioning enrollment and the provisioning status traps/informs as well as a SNMP engine for retrieving device capabilities and setting the TFTP filename and access method. Refer to the PacketCable MTA MIB [2] for a description of the MIB accessible MTA attributes.

The interface between the Provisioning Application and the associated SNMP Entity is not specified in PacketCable 1.0 and is left to vendor implementation. The interface between the Provisioning Server and the TFTP Server is not specified in PacketCable 1.0 and is left to vendor implementation.

5.4.3 MTA to Telephony Syslog Server

E-MTA MUST receive its Telephony Syslog Server IP address in the DHCP OFFER, option 7 (RFC-2132). The length of the option MUST be 4 octets. The value of the option MUST be one of the following:.

- 0.0.0.0 or FF.FF.FF.FF - means that Syslog logging for MTA is turned off
- Valid IP address of the Telephony Syslog Server.

The MTA's SYSLOG message (when used) MUST be sent in the following format:

<level>MTA[vendor]:<eventId>text

Where:

level – ASCII presentation of the event priority, enclosed in angle brackets, which is constructed as a logical 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 PACKETCABLE for the standard PACKETCABLE messages.

eventId – ASCII presentation of the INTEGER number in HEX format, enclosed in angle brackets, which uniquely identifies the type of event.

Example: Syslog event for AC power failure in the MTA

<132>MTA[CableLabs]:<65535>AC Power Fail

In case of failure, an MTA MUST report the result of each Provisioning Flow as a Provisioning Event unless Syslog is set to 0.0.0.0or FF.FF.FF.FF. An MTA MUST use the information in Appendix I when formatting the Provisioning Failure Events and reporting them to a Syslog Server.

5.4.4 MTA to DHCP Server

This interface identifies specific requirements in the DHCP server and the client for IP assignment during the MTA initialization process.

- Both the DHCP server and the embedded-MTA MUST support DHCP option code 6, 7, 12, 15, 43, 60 and DHCP option code 122 (defined in [11]). Option code 12 (Host Name) and 15 (Domain Name) MUST form a Fully Qualified Domain Name and MUST be resolvable by the DNS server.
- The DHCP server MUST accept and support broadcast and unicast messages per RFC 3396 [10] from the MTA DHCP client.
- The DHCP server MUST include the MTA's assigned FQDN in the DHCP offer message to the MTA-component of the embedded-MTA. Refer to RFC 2131 [1] for details describing the DHCP offer message.

5.4.5 MTA to Provisioning Application

This interface identifies specific requirements for the Provisioning Application to satisfy MTA initialization and registration. The Provisioning Application requirements are:

- The MTA MUST generate a correlation ID — an arbitrary value that will be exchanged as part of the device capability data to the Provisioning Application. This value is used as an identifier to correlate related events in the MTA provisioning sequence.
- The Provisioning Application MUST provide the MTA with its MTA configuration data file. The MTA configuration file is specific to the MTA-component of the embedded-MTA and separate from the CM-component's configuration data file.
- The configuration data file format is TLV binary data suitable for transport over the specified TFTP or HTTP access method.
- The Provisioning Application MUST have the capability to configure the MTA with different data and voice service providers.
- The Provisioning Application MUST use only SNMPv3 to provision devices in the Secure Flow. The support of the Basic and Hybid Flows is optional for the Provisioning Application. If the Basic and Hybrid Flows are supported, the Provisioning Application MUST use only SNMPv2c to provision devices in the Hybrid or Basic Flow.
- The Provisioning Application MUST provide SNMPv3 and SNMPv2 for device management.
- The Provisioning Application MUST support online incremental device/subscriber provisioning using SNMP with security enabled for devices provisioned with the Secure Flow. If supported, the Provisioning Application MUST support online incremental device/subscriber provisioning using SNMP with security disabled for devices provisioned with the Basic or Hybrid Flow.
- MTA MUST Specify all of its Capabilities in DHCP Option-60 in accordance with section 10.
- Provisioning Application MUST NOT assume any Capabilities, which do not have default values. In case if Capabilities supplied by the MTA are not consistent in format and/or in number and/or in values, the Provisioning Application MUST use the other means to identify the MTA's capabilities (e.g. SNMPv3 if possible).

5.4.6 MTA to CMS

Signaling is the main interface between the MTA and the CMS. Refer to the PacketCable signaling document [4] for a detailed description of the interface.

The CMS MUST accept signaling and bearer channel requests from a MTA that has an active security association.

The CMS MUST NOT accept signaling and bearer channel requests from a MTA that does not have an active security association unless provisioned to do so with information corresponding to the "pktcMtaDevCmslpsecCtrl" MIB Object.

5.4.7 MTA to Security Server (KDC)

The interface between the MTA and the Key Distribution Center (KDC) MUST conform to the PacketCable security specification [5].

AP-REQ/REP exchange back off and retry mechanism of the Kerberized SNMPv3 key negotiation defined in [5] is controlled by the values delivered by DHCP Option 122 sub-option 5 (see section 8.1.4).

AS-REQ/REP exchange backoff and retry mechanism of the Kerberized SNMPv3 key negotiation defined in [5] is controlled by the values delivered by DHCP Option 122 sub-option 4 (see section 8.1.3) or by the default values of the corresponding MIB objects in the Realm Table if sub-option 4 is not present in the DHCP Option 122.

5.4.8 MTA and Configuration Data File Access

This specification allows for more than one access method to download the configuration data file to the MTA.

- The MTA MUST support the TFTP access method for downloading the MTA configuration data file.
- The MTA MAY support HTTP access method for downloading the MTA configuration data file.
- The Provisioning Server MUST provide the MTA with the URL-encoded TFTP/HTTP server address and configuration filename via a SNMPv3 SET for the Secure Flow. The Provisioning Server MUST provide the MTA with the URL-encoded TFTP/HTTP server address via an SNMPv2c SET if it supports the Hybrid Flow provisioning mode. The Basic Flow does not require an SNMP SET to get the configuration file; the Provisioning Server MUST provide the MTA with the TFTP/HTTP server address in the DHCP "file" and "siaddr" fields if it supports the Basic Flow provisioning mode. For additional information refer to section 7.3.

5.4.9 DOCSIS extensions for MTA Provisioning

This specification requires that the following additions to DOCSIS flows for MTA auto-provisioning be supported:

- A new DHCP option code 122 and the associated procedures MUST be implemented in DOCSIS.

6 PROVISIONING OVERVIEW

Provisioning is a subset of configuration management control. The provisioning aspects include, but are not limited to, defining configurable data attributes, managing defined attribute values, resource initialization and registration, managing resource software, and configuration data reporting. The resource (also referred to as the managed resource) always refers to the MTA device. Further, the associated subscriber is also referred to as a managed resource.

6.1 Device Provisioning

Device provisioning is the process by which an embedded-MTA device is configured to support voice communications service.

In either case, device provisioning involves the MTA obtaining its IP configuration required for basic network connectivity, announcing itself to the network, and downloading of its configuration data from its provisioning server.

When the device is provisioned using the "Secure Flow", the MTA device MUST be able to verify the authenticity of the configuration file it downloads from the server. The “Secure Flow” generated configuration file is “signed” and may be “sealed”. Please refer to [5] for further information.

Please refer to section 5.4.1 for provisioning rules related to security associations.

When the device is provisioned using the Basic Flow or the Hybrid Flow, a content integrity verification check MUST be conducted on the configuration file by the MTA. For details refer to section 9.1.

6.2 Endpoint Provisioning

Endpoint provisioning is when a provisioned MTA authenticates itself to the CMS, and establishes a security association with that server. This allows subsequent call signaling to be protected under the established security association.

The MTA MUST follow the requirements defined in the PacketCable Security Specification [5] for NCS Kerberized Key Management, independently of the provisioning flow (Secure, Hybrid or Basic Flow) the MTA was provisioned with.

6.3 Secure Flow Provisioning State Transitions

Figure 4 represents logical device states and the possible transitions across these logical states. This representation is for illustrative purposes only, and is not meant to imply a specific implementation. The following MTA state transitions do not specify the number of retry attempts or retry time out values.

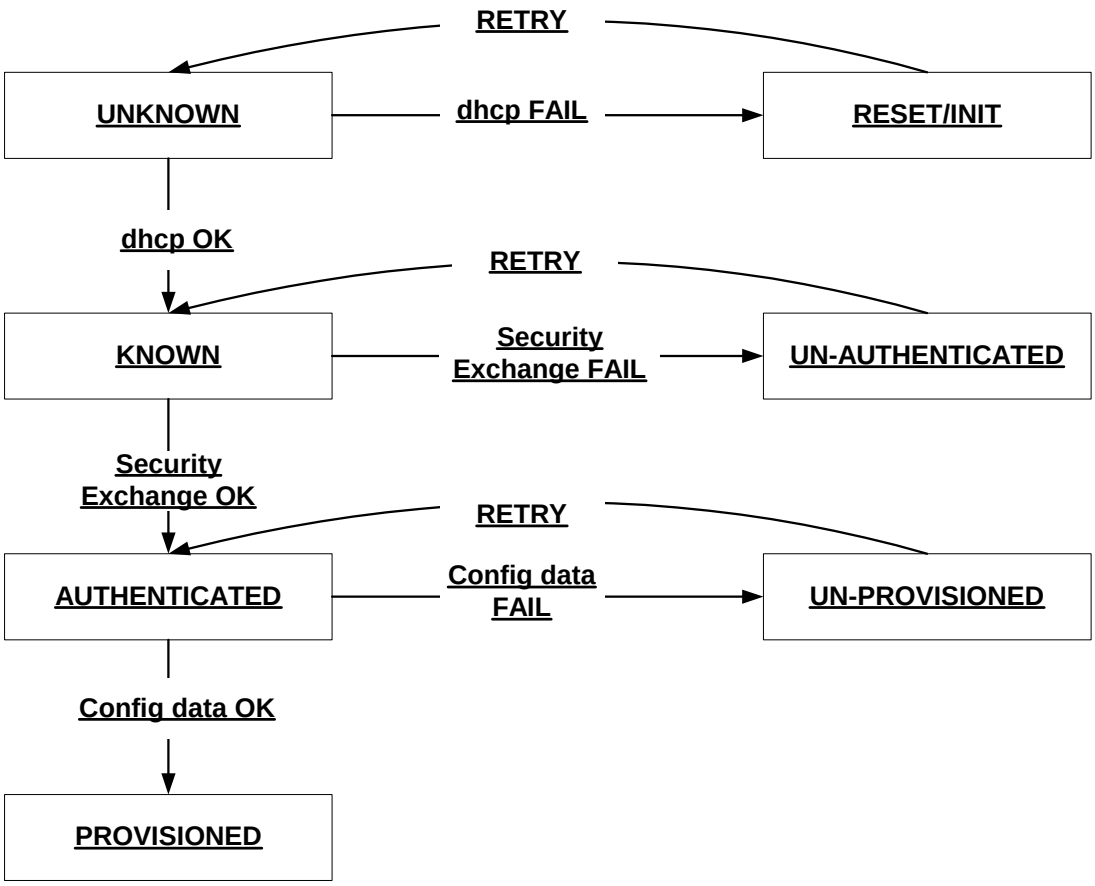


Figure 4. Device States and State Transitions for Secure Flow Provisioning

6.4 Basic and Hybrid Flow Provisioning State Transitions

Figure 5 represents logical device states and the possible transitions across these logical states. This representation is for illustrative purposes only, and is not meant to imply a specific implementation. The following MTA state transitions do not specify the number of retry attempts or retry time out values.

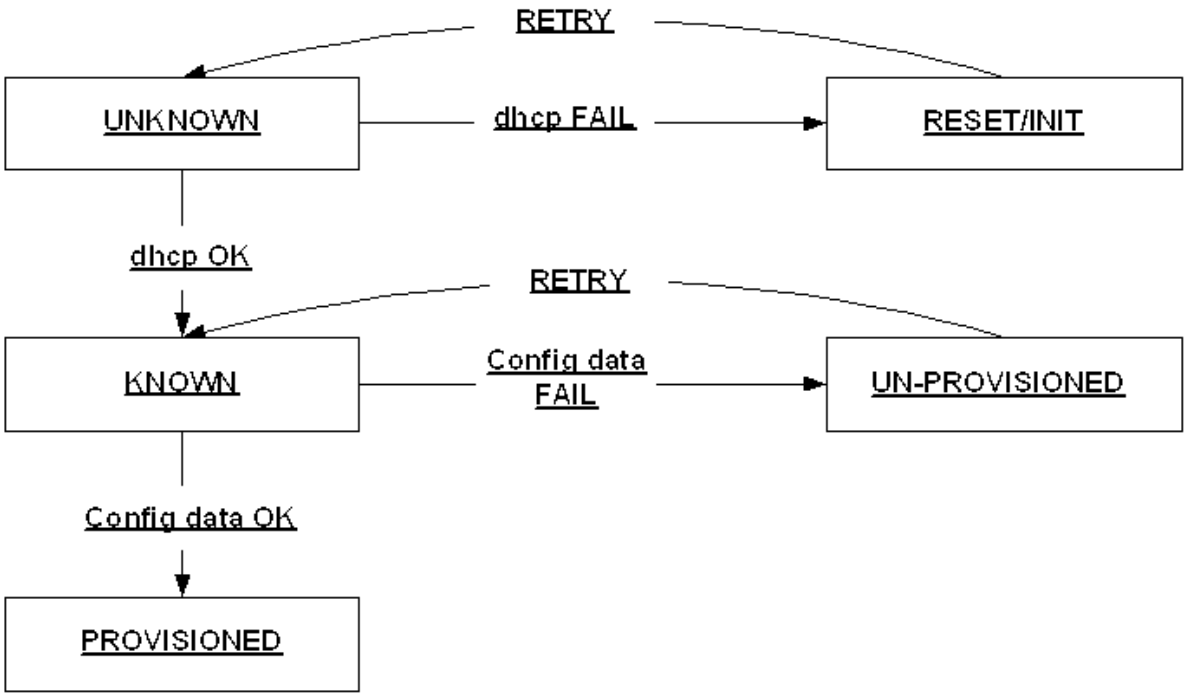


Figure 5. Device States and State Transitions for Basic and Hybrid Flow Provisioning

7 PROVISIONING FLOWS

A PacketCable MTA is provisioned via one of three provisioning flows.

- The Secure Flow supports Kerberos mutual authentication between the MTA and the provisioning system, as well as Kerberized SNMPv3 messaging. The Secure Flow MUST be supported by PacketCable MTAs and the Provisioning Applications.
- The Basic Flows are a simplified DOCSIS-like provisioning flows with no Kerberos or SNMPv3 security and no SNMP enrollment via SNMP INFORM. The Basic Flows SHOULD be supported by PacketCable MTAs and Provisioning Applications.
- The Hybrid Flows are essentially the Secure flow with the Kerberos message exchanges removed, and SNMPv2c substituted for SNMPv3. The Hybrid Flows SHOULD be supported by PacketCable MTAs and Provisioning Applications.

Any mention of SNMP in this specification without a specific reference to the SNMP protocol version must be interpreted as follows:

- For the Secure Flow, the MTA MUST support 'SNMPv3 only' for Provisioning and SNMPv3/v2c co-existence for Network Management and/or Monitoring operations. The SNMPv3/v2c co-existence MUST be supported and is configured using the TLV-38, TLV-11 and TLV64 in the MTA configuration file.
- For the Hybrid or Basic Flows, the MTA MUST support SNMPv2c for Provisioning, Network Management and/or Monitoring operations. The level of SNMPv2c access MUST be supported according to the values of the TLV38 TLV-11s and TLV64 in the MTA configuration file.

An MTA can also be configured with additional SNMPv2c targets via its configuration file by using TLV38 or TLV11 and TLV64.

An MTA is commanded to execute a specific flow via the contents of DHCP option 122 sub-option 6, as described in section 8.1.5. Each of these flows begin with a common set of flow steps.

7.1 Backoff, Retries, and Timeouts

Backoff mechanisms help the network to throttle device registration during a typical or mass registration condition when the MTA client requests are not serviced within the protocol specified timeout values. The details of provisioning behavior under mass-registration is beyond the scope of PacketCable 1.0, however this section provides the following recommendations and requirements.

- The recommendation for the throttling of registration MAY be based on DOCSIS 1.1 CM registration.
- The MTA MUST follow DHCP [1], and HTTP specifications for the timeout and retry mechanisms. It is recommended to follow IETF RFC 3413 [7] for SNMP timeout and retry mechanisms.
- The MTA MUST use an adaptive timeout for TFTP as specified in the DOCSIS 1.1 specification.
- The MTA MUST follow backoff and retry recommendations that are defined in the security specification [5] for the security message flows.

7.2 Embedded-MTA Power-On Initialization Flow (Secure Flow)

Following is the mandatory message flow that the embedded-MTA device MUST follow during power-on initialization (unless stated explicitly otherwise). It is understood that these flows do not imply implementation or limit functionality.

Although these flows show the MTA configuration file download from a TFTP Server, the descriptive text details the requirements to support the MTA configuration file download from a HTTP Server.

Note in the flow details below that certain steps may appear to be a loop in the event of a failure. In other words, the step to proceed to if a given step fails, is to retry that step again. However, it is recommended that if the desired number of backoff and retry attempts does not allow the step to successfully complete, the device detecting the failure should generate a failure event notification.

In the flow details below, the calculation of the Hash and the Encryption/Decryption of the MTA's Configuration File MUST follow requirements in [5].

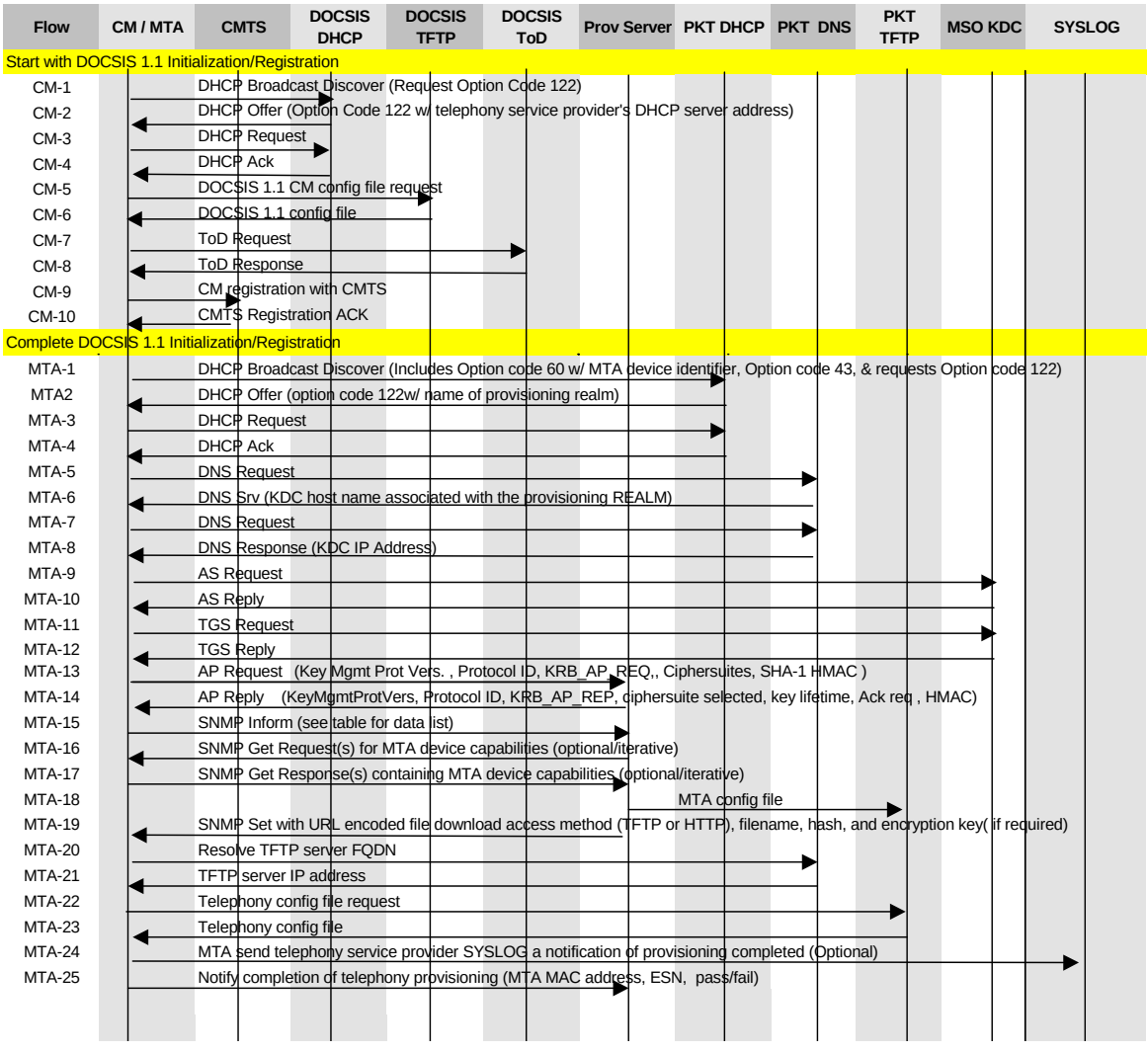


Figure 6. Embedded-MTA Secure Power-on Initialization Flow

Flow	Embedded-MTA Power-On Initialization Flow Description	Normal Flow Sequencing	MUST Proceed to here if this step fails
NOTE: Refer to the [6] for a complete description of flows CM1- CM10.			
CM1	As defined in the DOCSIS 1.1 specified registration sequence, the client device begins device registration by having the cable modem component send a broadcast DHCP discover message. This message includes Option code 60 (Vendor Specific Option) in the format "docsis1.1:xxxxxxx". This message MUST request Option 122 in Option 55, the request parameter list. The remainder of this message MUST conform to the DHCP discover data as defined in the DOCSIS 1.1 specification.	Initial MUST Step in Sequence	Per DOCSIS
CM2	The DOCSIS DHCP Server, if it has been configured to support MTA devices, MUST include Option Code 122 with sub-option 1 and, possibly, sub-option 2 as per section 8.1. If it is configured to prevent the MTA portion of the device from provisioning, then sub-option 1 in Option Code 122 MUST be set to 0.0.0.0. DOCSIS DHCP Servers without any prior knowledge of MTA devices MAY respond with DHCP OFFERS without including option 122.	CM2 MUST occur after CM1 Completion	Per DOCSIS
CM3	Upon receiving a DHCP OFFER, the CM MUST check for the requested option 122. If it is not present then it MUST retry the DHCP DISCOVER process (CM1) exponentially for 3 attempts (e.g. 2, 4, 8 second intervals). Upon failing to receive any DHCP OFFER with option 122 after the exponential retry mechanism it MUST consider OFFERs without option code 122 and accept one of them as per the DHCP specification [1]. The client device (CM) MUST then send a DHCP REQUEST broadcast message to the DHCP server whose OFFER is accepted as specified in the DHCP specification [1].	CM3 MUST occur after CM2 Completion	Per DOCSIS
CM4	The DHCP server sends the client device cable modem component a DHCP ACK message to confirm acceptance of the offered data. Upon receiving the DHCP ACK, the CM MUST check again for option 122. The absence of option 122 in the DHCP ACK message, that was accepted by the CM, implies that it MUST NOT initialize the embedded MTA. The presence of option 122 implies that it MUST initialize the MTA and pass suboption 1 and, possibly, suboption 2. If the option content of this DHCP ACK differs with the preceding DHCP OFFER, the option content of this DHCP ACK MUST be treated as authoritative (per RFC 2131).	CM4 MUST occur after CM3 Completion	Per DOCSIS

Flow	Embedded-MTA Power-On Initialization Flow Description	Normal Flow Sequencing	MUST Proceed to here if this step fails
CM5-CM10	The client device's cable modem component completes the remainder of the DOCSIS 1.1 specified registration sequence. This includes downloading the DOCSIS configuration file, requesting time of day registration, and registering with the CMTS.	CM5 – CM10 MUST occur after CM4 completion	Per DOCSIS
MTA1	DHCP Broadcast Discover The MTA MUST send a broadcast DHCP DISCOVER message. This message MUST include option code 60 (Vendor Specific Option) in the format "pktc1.0:xxxxxx". The MTA MUST include the DHCP option code 43 in the DHCP DISCOVER message as defined in section 8.5. The MTA MUST request in DHCP option 55 the following: 1, 3, 6, 7, 12, 15, and 122 options. If the CM DHCP option code 122 sub-option 1 (passed by the CM to the MTA) contains a DHCP server of value of 0.0.0.0, then the MTA MUST not attempt to provision and MUST remain dormant until it is reinitialized by the CM.	MTA1 MUST NOT occur before completion of CM4	If failure per DHCP protocol repeat MTA1
MTA2	DHCP OFFER The MTA may receive multiple DHCP OFFERs (during its wait period as per RFC 2131 [1].3. The following requirements apply to the MTA and/or the Provisioning Applications. 1. The MTA MUST only accept a valid DHCP OFFER message. A valid DHCP OFFER MUST be sent by the primary or secondary DHCP servers returned in DHCP option code 122 sub-options 1 and 2 as obtained by the E-MTA via the CM provisioning step CM-4. A valid DHCP OFFER MUST also include the following options: 1, 3, 6, 7, 12, 15, 122 with DHCP option 122 sub-options 3 and 6. DHCP option 122 MAY contain the additional sub-options 4, 5, 7, 8, and 9. 2. If the DHCP option 122 sub-option 6 returned by a valid DHCP server indicates that the Basic or Hybrid flow must be performed, the MTA MUST ignore the DHCP option 122 sub-options 4, 5, 7 and 9 if they are present. 3. If the DHCP option 122 sub-option 6 returned by a valid DHCP server indicates that the Basic Flow must be performed, the Provisioning Server MUST include the configuration file location in the 'siaddr' and 'file' fields in the DHCP responses. 4. If the DHCP option 122 sub-option 6 returned by a valid DHCP server indicates the Secure flow must be performed, the MTA MUST process the DHCP option 122 sub-options 4, 5, 7, and 9. The MTA next applies the following rules to the set of valid DHCP OFFERs:	MTA2 MUST occur after MTA1 completion	If failure per DHCP protocol return to MTA1

Flow	Embedded-MTA Power-On Initialization Flow Description	Normal Flow Sequencing	MUST Proceed to here if this step fails
	a. The MTA MUST check the value of the DHCP option 122 sub-option 3. If all valid OFFERs contain 0.0.0.0 in DHCP option 122 sub-option 3, then the MTA MUST not further the DHCP process and it MUST shutdown until it is reinitialized. Otherwise, the MTA MUST further restrict its set of valid OFFERs to those with a non-zero value in the DHCP option 122 sub-option 3. b. The MTA MUST check the value of the DHCP option 122 sub-option 6 for indication of the Secure Flow. If no valid DHCP OFFER message directs the MTA to the Secure flow, the MTA MUST retry the DHCP DISCOVER process (MTA-1) exponentially for 3 attempts (e.g. 2, 4, 8 second intervals). Upon failing to receive any valid DHCP OFFER indicating the Secure flow, the MTA MUST select, a valid Hybrid Flow DHCP OFFER, or a valid Basic Flow OFFER in that order. If no valid DHCP OFFER is received, the MTA MUST fail the corresponding provisioning flow step. NOTE: In the case of Secure Flow, if an MTA supports TGTs and receives the DHCP option 122 sub-option 7 set to a FALSE value, it MUST NOT request TGTs. If an MTA supports TGTs and receives the DHCP option 122 sub-option 7 set to a TRUE value, it MUST request TGTs. MTAs that do not support TGTs MUST ignore the DHCP option 122 sub-option 7.		
MTA3	DHCP Broadcast REQUEST Once the MTA has selected a valid DHCP OFFER, the MTA MUST send a DHCP REQUEST broadcast message to accept the DHCP OFFER per [1].	MTA3 MUST occur after MTA2 completion	If failure per DHCP protocol return to MTA1
MTA4	DHCP ACK The DHCP server sends a DHCP ACK message to the MTA. The DHCP ACK message MUST include all options and sub-options which had been sent in MTA 2 (DHCP OFFER). If the option and sub-option values of this DHCP ACK differ with the preceding DHCP OFFER (MTA-2), the option and sub-option values of this DHCP ACK MUST be treated as authoritative (per RFC 2131 [1]). If the DHCP ACK is not valid as per the criteria established in MTA 2, the MTA MUST fail this step.	MTA4 MUST occur after MTA3 completion	If failure per DHCP protocol return to MTA1

Flow	Embedded-MTA Power-On Initialization Flow Description	Normal Flow Sequencing	MUST Proceed to here if this step fails
NOTE: The provisioning flow forks into one of three directions as follows: If the MTA4 DHCP ACK indicates the Basic Flow, the MTA MUST proceed to flow step BMTA-22 described in section 7.3. If the MTA4 DHCP ACK indicates the Hybrid Flow, the MTA MUST proceed to flow step HMTA-15 described in section 7.4. Otherwise, the Secure Flow is indicated and the MTA MUST proceed to step MTA5 below.			
MTA5	DNS Srv Request The MTA requests the MSO KDC host name for the Kerberos realm.	MTA5 MUST occur after MTA4 completion	MTA1
MTA6	DNS Srv Reply Returns the MSO KDC host name associated with the provisioning REALM.	MTA6 MUST occur after MTA5 completion	MTA1
MTA7	DNS Request The MTA now requests the IP Address of the MSO KDC.	MTA7 MUST occur after MTA6 completion	MTA1
MTA8	DNS Reply The DNS Server returns the IP Address of the MSO KDC.	MTA8 MUST occur after MTA7 completion	MTA1
MTA9	AS Request The AS Request message is sent to the MSO KDC to request a Kerberos ticket.	If MTA9 occurs, it MUST occur after MTA8 completion.	MTA1
MTA10	AS Reply The AS Reply Message is received from the MSO KDC containing the Kerberos ticket. NOTE: The KDC must map the MTA MAC address to the FQDN before send the AS Reply.	MTA10 MUST occur after MTA9 completion	MTA1
NOTES: (1) Flows MTA11– MTA12 are optional in some cases, please reference the Security Specification [5]. (2) SNMPv3 entity (FQDN) MUST be resolved to an IP address anywhere during flows MTA-5 to MTA12. (3) If an IP address is provided in the Additional information field of the DNS-SRV response (MTA6), MTA MAY use the same and skip the flows MTA7 and MTA8. (4) If the MTA has valid provisioning application server ticket saved in NVRAM, then it MUST skip the flows MTA5 to MTA12 in successive MTA resets (flows MTA1 to MTA25).			
MTA11	TGS Request If MTA obtained TGT in MTA10, the TGS Request message is sent to the MSO KDC.	MTA11 occurs, it MUST occur after MTA10 completion	MTA1

Flow	Embedded-MTA Power-On Initialization Flow Description	Normal Flow Sequencing	MUST Proceed to here if this step fails
MTA12	TGS Reply The TGS Reply message is received from the MSO KDC.	MTA12 MUST occur after MTA11 completion	MTA1
MTA13	AP Request The AP Request message is sent to the Provisioning Server to request the keying information for SNMPv3.	MTA13 MUST occur after MTA12 or MTA 10 completion	MTA1
MTA14	AP Reply The AP Reply message is received from the Provisioning Server containing the keying information for SNMPv3. NOTE: The SNMPv3 keys must be established before the next step using the information in the AP Reply.	MTA14 MUST occur after MTA13 completion	MTA1
MTA15	SNMP Enrollment INFORM The MTA MUST send an SNMPv3 Enrollment INFORM to the PROV_SNMP_ENTITY (specified in the DHCP option 122 sub-option 3). The SNMP INFORM MUST contain a "PktcMtaDevProvisioningEnrollment object as defined in [2]. The PROV_SNMP_ENTITY notifies the Provisioning Application that the MTA has entered the management domain.	MTA15 MUST occur after MTA14 completion	If failure per SNMP protocol return to MTA1. SNMP server MUST send response to SNMP-INFORM.
NOTE: The provisioning server can reset the MTA at this point in the flows. The MTA is part of the security domain and MUST respond to management requests, the SNMP INFORM of MTA15 is the indicator, see section 5.4.1.2.			
MTA16	SNMPv3 GET Request (Optional) If any additional MTA device capabilities are needed by the PROV_APP, the PROV_APP requests these from the MTA via SNMPv3 Get Requests. This is done by having the PROV_APP send the PROV_SNMP_ENTITY a "get request" Iterative: The PROV_SNMP_ENTITY sends the MTA one or more SNMPv3 GET requests to obtain any needed MTA capability information. The Provisioning Application may use a GETBulk request to obtain several pieces of information in a single message.	MTA16 is optional, can occur after MTA15 completion	N/A

Flow	Embedded-MTA Power-On Initialization Flow Description	Normal Flow Sequencing	MUST Proceed to here if this step fails
MTA17	SNMPv3 GET Response Iterative: MTA sends the PROV_SNMP_ENTITY a response for each GET Request. After all the Gets, or the GetBulk, finish, the PROV_SNMP_ENTITY sends the requested data to the PROV_APP.	MTA17 MUST occur after MTA16 completion if MTA16 is performed	N/A
MTA18	This Protocol is not defined by PacketCable. The PROV_APP MAY use the information from MTA16 and MTA17 to determine the contents of the MTA Configuration Data file. Mechanisms for sending, storing and, possibly, creating the configuration file are outlined in MTA19.	MTA18 SHOULD occur after MTA15 completion unless MTA16 is performed, then it SHOULD be after MTA17 has completed	N/A
MTA19	SNMPv3 SET The PROV_APP MAY create the configuration file at this point, or send a predefined one. A hash MUST be run on the contents of the configuration file. The configuration file MAY be encrypted The hash and the encryption key (if the configuration file is encrypted) MUST be sent to the MTA. The PROV_APP MUST store the configuration file on the appropriate TFTP server. The PROV_APP then instructs the PROV_SNMP_ENTITY to send an SNMP SET message to the MTA containing the URL-encoded file access method and filename, the hash of the configuration file, and the encryption key (if the configuration file is encrypted). NOTES: In the case of file download using the HTTP access method, the filename MUST be URL-encoded with a URL format compliant with RFC 2616 [37]. An example of a valid URL-encoded HTTP filename is http://[IPv4] or FQDN of access server/mta-config-filename. In the case of file download using the TFTP access method, the filename MUST be URL-encodedencoded with a URL format compliant with RFC 3617 [38]. An example of a valid URL-encoded TFTP filename is: tftp://[IPv4] or FQDN of access server/mta-config-filename.	MTA19 MUST occur after MTA18 completion	If failure per SNMP protocol return to MTA1

Flow	Embedded-MTA Power-On Initialization Flow Description	Normal Flow Sequencing	MUST Proceed to here if this step fails
MTA20	DNS Request If the URL-encoded access method contains a FQDN instead of an IPv4 address, the MTA MUST use the service provider network's DNS server to resolve the FQDN into an IPv4 address of either the TFTP Server or the HTTP Server.	MTA20 MUST occur after MTA19 completion if FQDN is used	If failure per DNS protocol return to MTA1
MTA21	DNS Reply DNS Response: DNS server returns the IP address against MTA20 DNS request.	MTA21 MUST occur after MTA20 completion if FQDN is used	If failure per DNS protocol return to MTA1
MTA22	TFTP/HTTP Configuration file Request The MTA MUST perform either the TFTP or HTTP protocol exchange, as specified in step S-MTA-19, to download its configuration file. For specific details of each protocol, see [9], [25]	MTA22 MUST occur after MTA19 unless FQDN is specified then MUST be after MTA20 – MTA21	If failure per TFTP or HTTP protocols, return to MTA1
MTA23	TFTP/HTTP Configuration file Request The TFTP/HTTP Server MUST send the requested configuration file to the MTA. Specific details of each protocol are found in [9] and [25]. The hash of the downloaded configuration file is calculated by the MTA and compared to the value received in step MTA-19. If the hashes do not match, the MTA MUST fail this step. If encrypted, the configuration file MUST be decrypted. If the MTA does not complete this step in the time specified by the MIB object 'pktcMtaDevProvisioningTimer' the device MUST return to MTA1. Refer to section 9.1 for MTA configuration file contents.	MTA23 MUST occur after MTA22 completion	If the configuration file download failed per TFTP or HTTP protocols, return to MTA1. Otherwise, proceed to MTA24 or MTA25, and send the failed response if the MTA configuration file itself is in error.
MTA24	SYSLOG Notification The MTA SHOULD send the voice service provider's SYSLOG (specified in DHCP option 7) a "provisioning complete" notification. This notification will include the pass-fail result of the provisioning operation. The general format of this notification is as defined in section 5.4.3.	MTA24 is optional, can occur after MTA23 completion if SYSLOG used	A vendor MAY consider returning to MTA15, repeating until it is determined to be a hard failure and then MUST continue to MTA25.

Flow	Embedded-MTA Power-On Initialization Flow Description	Normal Flow Sequencing	MUST Proceed to here if this step fails
MTA25	SNMP INFORM The MTA MUST send the PROV_SNMP_ENTITY(specified in DHCP option 122 sub-option 3) an SNMP INFORM containing a "provisioning complete" notification. The receipt of the inform is acknowledged by the response message as defined in RFC 3414 [8]. The SNMP INFORM MUST contain a "PktcMtaDevProvisioningStatus" object as defined in [2]. NOTE: 1. At this stage, the MTA device provisioning data is sufficient to provide any minimal services as determined by the service provider (e.g. 611). 2. Depending on the TLV38 configuration, there might be multiple SNMP INFORMs sent to the configured SNMP Management stations.	MTA25 MUST occur after MTA24 if SYSLOG is used, otherwise MUST occur after MTA23 completion	MTA MAY generate a Provisioning Failure event notification to the Service Provider's Fault Management server. Provisioning process stops; Manual interaction required.SNMP server MUST send response to SNMP-INFORM.

7.3 Embedded-MTA Power-On Initialization Flow (Basic Flow).

The Basic MTA provisioning flow is very similar to the DOCSIS CM provisioning flow.

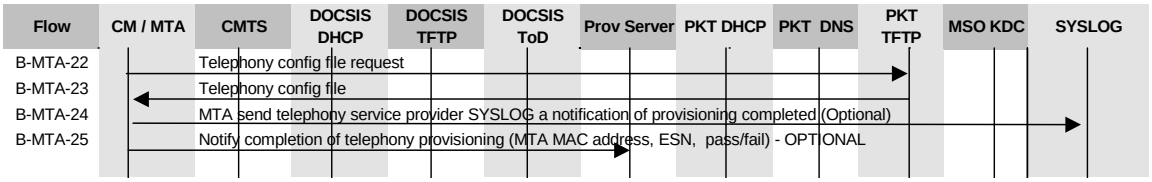


Figure 7. Embedded-MTA Basic Power-on Initialization Flow

Flow	Embedded-MTA Power-On Initialization Flow Description	Normal Flow Sequencing	MUST proceed here if this step fails
	NOTE: the FQDN provided in the DHCP ACK in DHCP option 122 sub-option 3 (Provisioning Entity Address) MUST be resolved to an IP address before step B-MTA-22.		
B-MTA-22	TFTP Configuration File Request The MTA MUST perform a TFTP protocol exchange to download its configuration file. The 'siaddr' and 'file' fields of the DHCP ACK are used to locate the configuration file. Specific details of the TFTP protocol can be found in [9].	B-MTA-22 MUST occur after MTA-4.	If failure per TFTP protocol, return to MTA-1.

Flow	Embedded-MTA Power-On Initialization Flow Description	Normal Flow Sequencing	MUST proceed here if this step fails
B-MTA-23	TFTP Configuration File Response The TFTP server MUST send the requested configuration file to the MTA. Specific details of the TFTP protocol can be found in [9]. The downloaded configuration file MUST contain the MIB object 'pktcMtaDevConfigHash'. The MTA MUST calculate the hash of the downloaded configuration file per section 9.1 and compare this value to the value contained in the 'pktcMtaDevConfigHash' object. If these values do not match, this step MUST fail. If the MTA does not complete this step in the time specified by the MIB object 'pktcMtaDevProvisioningTimer', the MTA device MUST return to MTA-1. Refer to section 9.1 for MTA configuration file contents	B-MTA-23 MUST occur after B-MTA-22	If the configuration file download failed per TFTP protocols, return to MTA1. Otherwise, proceed to B- MTA24 and send the failed response if the MTA configuration file itself is in error
B-MTA-24	SYSLOG Notification (optional) The MTA SHOULD send the voice service provider's SYSLOG (specified in DHCP option 7) a "provisioning complete" notification. This notification will include the pass-fail result of the provisioning operation. The general format of this notification is as defined in section 5.4.3	B-MTA-24 is optional, MAY occur after B-MTA-23 completion if SYSLOG used	Return to MTA-1.
B-MTA-25	SNMPv2C Provisioning Status INFORM (optional) If commanded by DHCP option 122 sub-option 6, the MTA MUST send the PROV_SNMP_ENTITY (specified in DHCP option 122 sub-option 3) an SNMP INFORM containing a "provisioning complete" notification. The receipt of the SNMP INFORM is acknowledged. The SNMP INFORM MUST contain a "PktcMtaDevProvisioningStatus" object as defined in [2] The SNMPv2c community name used in the status SNMP INFORM MUST have a value "public"(taken without the quotation mark). NOTES: 1. At this stage, the MTA device provisioning data is sufficient to provide any minimal services as determined by the service provider (e.g. 611). 2. Depending on the TLV38 configuration value pairs, there might be multiple SNMP INFORMs	B-MTA-25 is optional, MAY occur after B-MTA-24 if SYSLOG is used, otherwise MAY occur after B-MTA-23 completion	Provisioning process stops; Manual interaction required. SNMP server MUST send response to SNMP-INFORM

Flow	Embedded-MTA Power-On Initialization Flow Description	Normal Flow Sequencing	MUST proceed here if this step fails
	sent to the configured SNMP Management stations.		

7.4 Embedded-MTA Power-On Initialization Flow (Hybrid Flow).

The Hybrid Provisioning Flow (Hybrid Flow) is essentially the Secure Flow with Kerberos exchanges removed and SNMPv2c substituted for SNMPv3. The SNMPv2c community name, used in the SNMP INFORM messages sent by the MTA in steps H-MTA15 and H-MTA25 below, MUST have a value "public"(taken without the quotation mark).

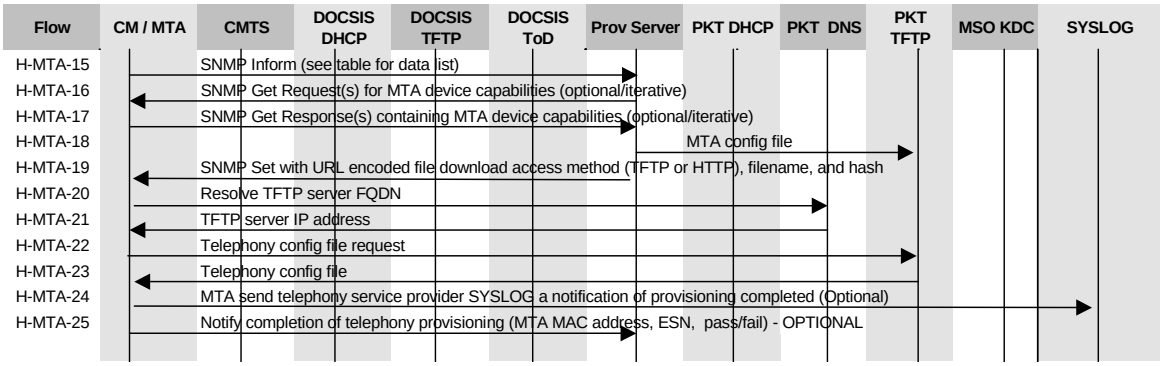


Figure 8. Embedded-MTA Hybrid Power-on Initialization Flow

Flow	Embedded-MTA Power-On Initialization Flow Description	Normal Flow Sequencing	MUST proceed here if this step fails
	Note: the FQDN provided in the DHCP ACK in DHCP option 122 sub-option 3 (Provisioning Entity Address) MUST be resolved to an IP address before step H-MTA-15.		
H-MTA-15	SNMPv2c Enrollment INFORM The MTA MUST send a SNMPv2c Enrollment INFORM to PROV_SNMP_ENTITY (specified in the DHCP option 122 sub-option 3). The SNMP INFORM MUST contain a 'PktcMtaDevProvisioningEnrollment' object as defined in [2]. The PROV_SNMP_ENTITY notifies the PROV_APP that the MTA has entered the management domain.	H-MTA-15 MUST occur after MTA-4 completion	If failure per SNMP protocol return to MTA-1. SNMP server MUST send response to SNMP-INFORM.
H-MTA-16	SNMPv2c GET Request (optional)	H-MTA-16 is optional, can occur after H-	N/A

Flow	Embedded-MTA Power-On Initialization Flow Description	Normal Flow Sequencing	MUST proceed here if this step fails
	The Provisioning Application may request additional MTA device capabilities from the MTA via SNMPv2c GET requests. This is done by having the Provisioning Application send the PROV_SNMP_ENTITY an SNMP GET request. Iterative: The PROV_SNMP_ENTITY sends the MTA one or more SNMPv2c GET requests to obtain any needed MTA capability information. The Provisioning Application may use a GETBulk request to obtain several pieces of information in a single message.	MTA-15 completion	
H-MTA-17	SNMPv2c GET Response (optional) Iterative: MTA sends the PROV_SNMP_ENTITY a Get Response for each Get Request. After all the Gets, or the GetBulk, finish, the PROV_SNMP_ENTITY sends the requested data to the Provisioning Application.	H-MTA-17 MUST occur after H-MTA-16 completion if H-MTA-16 is performed	N/A
H-MTA-18	This protocol is not defined by PacketCable. The Provisioning Application MAY use the information from H-MTA-15, -16, and -17 to determine the contents of the MTA configuration data file. Mechanisms for sending, storing and, possibly, creating the configuration file are outlined in H-MTA-19.	H-MTA-18 SHOULD occur after H-MTA-15 completion unless H_MTA-16 is performed, then it SHOULD be after H-MTA-17 has completed	N/A
H-MTA-19	SNMPv2c Configuration File Set The Provisioning Application MAY create the configuration file at this point, or send a predefined one. The Provisioning Application MUST calculate SHA-1 hash on the contents of the configuration file. The Provisioning Application MUST store the configuration file on the appropriate TFTP server. The Provisioning Application then instructs the PROV_SNMP_ENTITY to send an SNMPv2c SET message to the MTA, containing the following varbindings (defined in [2]): pktcMtaDevConfigFile	H-MTA-19 MUST occur after H-MTA-18 completion	If failure per SNMP protocol return to MTA-1

Flow	Embedded-MTA Power-On Initialization Flow Description	Normal Flow Sequencing	MUST proceed here if this step fails
	pktcMtaDevProvConfigHash Unlike the Secure Flow, the pktcMtaDevProvConfigKey MIB object MUST NOT be included. If the pktcMtaDevProvConfigKey MIB object is included, the MTA MUST ignore the value. In the case of file download using the HTTP access method, the filename MUST be URL-encoded with a URL format compliant with RFC 2616 [37].. An example of a valid URL-encoded HTTP filename is http://[IPv4] or FQDN of access server/mta-config-filename In the case of file download using the TFTP access method, the filename MUST be URL-encoded with a URL format compliant with RFC 3617 [38]. An example of a valid URL-encoded TFTP filename is tftp://[IPv4] or FQDN of access server/mta-config-filename		
H-MTA-20	DNS Request (optional) If the URL-encoded access method contains a FQDN instead of an IPv4 address, the MTA MUST use the service provider network's DNS server to resolve the FQDN into an IPv4 address of either the TFTP Server or the HTTP Server.	H-MTA-20 MUST occur after H-MTA-19 completion if FQDN is used	If failure per DNS protocol return to MTA-1
H-MTA-21	DNS Reply (optional) DNS Response: DNS server returns the IP address against H-MTA-20 DNS request.	H-MTA-21 MUST occur after H-MTA-20 completion if FQDN is used	If failure per DNS protocol return to MTA-1
H-MTA-22	TFTP/HTTP Configuration file Request The MTA MUST perform either the TFTP or HTTP protocol exchange, as specified in step H-MTA-19, to download its' configuration file. Specific details of each protocol see [9], [25]	H-MTA-22 MUST occur after H-MTA-19 unless FQDN is specified then MUST be after H-MTA-21.	If failure per TFTP or HTTP protocols, return to MTA-1.
H-MTA 23	TFTP/HTTP Configuration file Response TFTP/HTTP server MUST send the requested configuration file to the MTA. Specific details of	H-MTA-23 MUST occur after H-MTA-22	If the configuration file download failed per

Flow	Embedded-MTA Power-On Initialization Flow Description	Normal Flow Sequencing	MUST proceed here if this step fails
	each protocol are found in [9], [25] The hash of the downloaded configuration file is calculated by the MTA and compared to the value received in step H-MTA-19. If the hashes do not match, this step MUST fail. If the MTA does not complete this step in the time specified by the MIB object 'pktcMtaDevProvisioningTimer' the device MUST return to MTA-1. Refer to section 9.1 for MTA configuration file contents		TFTP or HTTP protocols, return to MTA1. Otherwise, proceed to MTA24 or MTA25, and send the failed response if the MTA configuration file itself is in error
H-MTA-24	SYSLOG Notification (optional) The MTA SHOULD send the voice service provider's SYSLOG (specified in DHCP option 7) a "provisioning complete" notification. This notification will include the pass-fail result of the provisioning operation. The general format of this notification is as defined in section 5.4.3.	H-MTA-24 is optional, can occur after H-MTA-22 completion if SYSLOG used	A vendor MAY consider returning to H-MTA-15, repeating until it is determined to be a hard failure and then MUST continue to H-MTA-25.

Flow	Embedded-MTA Power-On Initialization Flow Description	Normal Flow Sequencing	MUST proceed here if this step fails
H-MTA-25	SNMPv2C Provisioning Status Inform (optional) If commanded by DHCP 122 sub-option 6, the MTA MUST send the PROV_SNMP_ENTITY (specified in DHCP option 122 sub-option 3) a SNMPv2C Provisioning Status INFORM containing a "provisioning complete" notification. The receipt of the inform is acknowledged. The inform MUST contain a 'PktcMtaDevProvisioningStatus' object as defined in [2]. NOTES: 1. At this stage, the MTA device provisioning data is sufficient to provide any minimal services as determined by the service provider (e.g. 611). 2. Depending on the TLV38 configuration there might be multiple SNMPv2C INFORM sent to the configured SNMP Management stations.	H-MTA-25 is optional. It MAY occur after H-MTA-24 if SYSLOG is used, otherwise it MAY occur after H-MTA-23 completion	Provisioning process stops; Manual interaction required. SNMP server MUST send response to SNMP-INFORM

7.5 Endpoint Provisioning Completion Notifications

After the MTA has been provisioned successfully regardless of the selected provisioning flow, the MTA will set up the necessary security association for the related CMS configured realms (KDCs). The MTA NCS signaling software will initiate the establishment of the IPSec security association to the configured CMS clusters. Event notifications are triggered if security associations cannot be established (based on [5]).

With the selected Basic, Hybrid, or Secure flow complete, and after any required security associations are established, the MTA NCS signaling software determines whether a signaling path can be setup with an RSIP message and the associated ACK. Coming from a link down situation, the MTA will send an SNMP Link Up Trap when the RSIP has been properly acknowledged. This indicates that the endpoint is provisioned. If the same CMS is used for multiple endpoints, a SNMP link up message will be sent for each associated endpoint. If not all endpoints use the same CMS, the same process needs to be repeated for each endpoint needing a different configured CMS.

7.6 Post Initialization Incremental Provisioning

This section describes the flows allowing the Provisioning Application to perform incremental provisioning of individual voice communications endpoints after the MTA has been initialized. Post-Initialization incremental provisioning MAY involve communication with a Customer Service Representative (CSR).

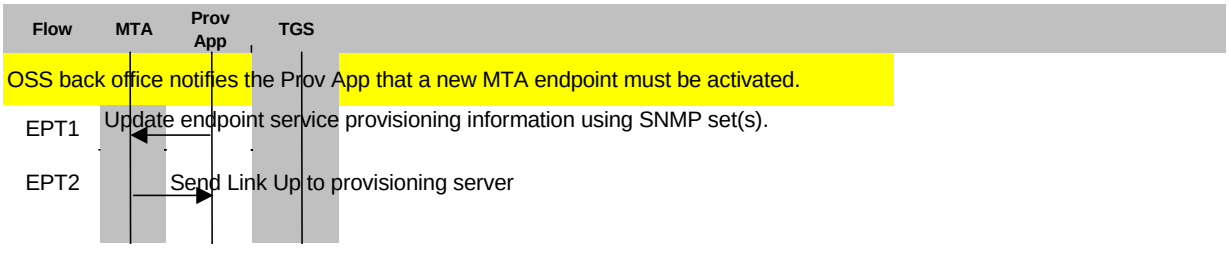
7.6.1 Synchronization of Provisioning Attributes with Configuration File

Incremental provisioning includes adding, deleting and modifying subscriber services on one or more endpoints of the embedded-MTA. Services on an MTA endpoint MUST be modified using SNMP via the MTA MIB [2]. The back office applications MUST support a “flow-through” provisioning mechanism that synchronizes all device provisioning information on the embedded-MTA with the appropriate back office databases and servers. Synchronization is required in the event that provisioning information needs to be recovered in order to re-initialize the device. Although the details of the back office synchronization are beyond the scope of this document, it is expected that, at a minimum, the following information is updated: customer records, and the MTA configuration file on the TFTP or HTTP server.

7.6.2 Enabling Services on an MTA Endpoint

Services may be provisioned on a per-endpoint basis whenever it is desired to add or modify service to a previously unprovisioned endpoint. This would be the case if a customer was already subscribing to service on one or more lines (endpoints), and now wanted to add additional service on another line (endpoint).

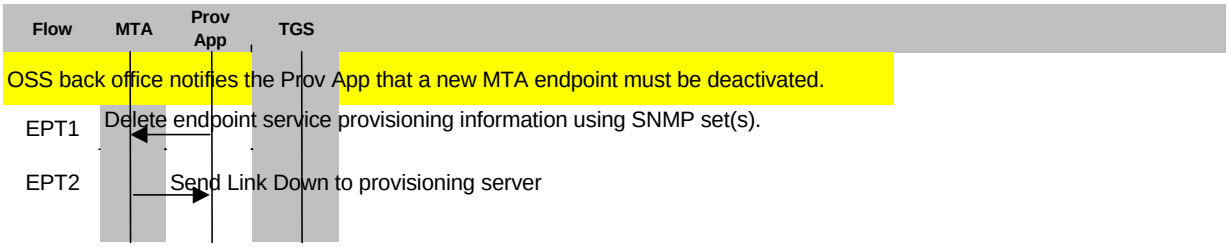
MTA Endpoint services are enabled using SNMP via the MTA MIB [2]. In this example, a subscriber is requesting that additional service be added. This example assumes the service provider’s account creation process has been completed, and shows only the applications critical for the flows. For instance, account creation and billing database creation are assumed to be available and integrated in the back office application suite.



Flow	Enabling Services on an MTA Endpoint Flow Description	Normal Flow Sequencing
EPT1	The Provisioning Application will now use SNMP Sets to update provisioning attributes on the device for which the device port is being enabled. These SET operations MUST include the device port CMS ID (associate the device port to the CMS ID from which the features will be supported) and the device port to enable. See section 5.4.1 for details of provisioning rules.	MUST occur after successful power-on initialization flow
EPT2	When an additional endpoint is configured it follows the same procedure as described in section 7.3 with the exception that the process is only executed for the single endpoint configured. If the corresponding security association for new endpoint is already configured and the MTA NSC Signaling Software is currently not in a disconnected state (defined in [4]), the SNMP Link Up Trap will occur immediately after the endpoint is configured. Otherwise, it occurs after the process described in section 7.3 has completed. NOTE: The SNMP Link Up trap is not optional but may be masked using ifLinkUpDownTrapEnable.	EPT2 is Optional if Event Notification is used

7.6.3 Disabling Services on an MTA Endpoint

MTA Endpoint services are disabled using SNMP Sets to the MTA. In this scenario, subscriber's voice communications service is disabled from one of the MTA endpoints. This example assumes the service provider's account update process has been completed and shows only the applications critical to MTA operation.



Flow	Disabling Services on an MTA Endpoint Flow Description	Normal Flow Sequencing
EPT1	The Provisioning Application will now use SNMP Sets to delete provisioning attributes from the device endpoint for which the service is being disabled. This MUST include setting the associated security parameters to a NULL value.	MUST occur after successful power-on initialization flow
EPT2	An SNMP link down trap will occur immediately after the endpoint is unconfigured (i.e., the configuration data for the endpoint is deleted) unless the MTA is currently in a disconnected state with the associated CMS. When an endpoint is unconfigured, the MTA is not required to release any security associations unless explicitly told to do so.	EPT2 is optional if Event Notification is used

7.6.4 Modifying Services on an MTA Endpoint

MTA Endpoint services are modified using SNMPv3Sets to the MTA MIB [2]. In this scenario subscriber's voice communications service features are being modified on one of the MTA endpoints. Once again, the accounting management aspects of the back office application are assumed to be correct.

The following are possible service modifications and none of these modifications cause the device to request a new Kerberos ticket from the KDC.

- Modification of call service features (add, delete call features). Changes to services require modifications in the CMS, not in the MTA.
- Modification of service level (change the subscriber service levels with respect to the QoS definition). This is part of the DOCSIS 1.1 provisioning and requires changes to the CM component in the MTA which requires rebooting the embedded-MTA. This updates the MTA (CM) as the initialization sequence is executed as part of the bootup process.

If the modification to the endpoint changes pktcNcsEndPntConfigCallAgentId and/or pktcNcsEndPntConfigCallAgentUdpPort, the endpoint is taken out of service (SNMP Link Down Trap is sent) followed by the placing the port in service (SNMP Link Up Trap is sent upon completion) with the new parameters. The SNMP Link Up Trap occurs after the sequence in section 7.3 has completed. For all other modifications, no indication is given to the Provisioning Server.

7.7 Behavior During A Disconnected State

Whenever the MTA-CMS association goes from a connected state to a disconnected state (CMS is not responding to MTA CMS Signaling messages), the MTA will send the provisioning server an SNMP Link Down Trap on all the affected endpoints. If a security association between the CMS and the MTA expires while the MTA is in a connected state, the MTA/CMS link will be placed in a disconnected state and the MTA will send an SNMP Link Down Trap for all affected endpoints. Whenever the MTA recovers the security association and the RSIP/Acknowledge sequence occurs, the MTA will send an SNMP Link Up for all affected endpoints.

Whenever the MTA-CMS association recovers from a disconnected state, the MTA will send a SNMP Link Up Trap on all of the affected endpoints.

7.8 Provisioning of the Signaling Communication Path Between the MTA and CMS

The Service Flow(s) for NCS (NCS SF) is(are) not required on the MTA; however, if the NCS SF(s) is(are) implemented and provisioned then the NCS SF(s) MUST be used to provide a signaling communication path between an E-MTA and CMS. One SF MUST be set for all of the MTA's endpoints in each direction.

If NOT provisioned, the signaling communication path will use the primary SF to pass NCS packets.

The following types of DOCSIS Scheduling Services should be used for NCS SF: "Non-Real time Polling Service" (nRTP) or "Best Effort Service" (BE). The other types of services should not be used because the NCS flow characteristics don't match the scheduling characteristics. With either BE or nRTP, the CMTS will give scheduling preference to the NCS flow over the primary flow. With nRTP, the CMTS should also give unicast request opportunities whenever it can so as to allow those flows to avoid contention request collisions.

The creation of the NCS SF MUST occur before Voice Communication Service is activated unless the creation of the SF fails, then Voice Communication Service may be activated and NCS messages will flow over the primary SF.

If the value "pktcSigServiceClassNameMask" MIB Object is not zero and "pktcSigServiceClassNameUS" and "pktcSigServiceClassNameDS " MIB Objects are not empty, then NCS SF MUST be created.

If the value of "pktcSigServiceClassNameUS" or "pktcSigServiceClassNameDS" is set to empty string, then corresponding NCS SF MUST be deleted if it currently exists. After the NCS SF is deleted, and if Voice Communication Service is Enabled, then primary SF for NCS packets will be used instead. If the SNMP Manager sets the object to a value which is not empty string, the NCS SF MUST be created using the corresponding Service Class name.

If Voice Communication Services become Disabled for all end-points of the MTA, then the NCS SF SHOULD be deleted.

If an MTA becomes disabled (pktcMtaDevEnabled is set to FALSE), then the NCS SF SHOULD be deleted (if it exists).

The "pktcSigNcsServiceFlowState" MIB Object MUST indicate the state of the NCS SF according to the Object's description. The E-MTA MUST create the Call Signaling Service Flow using the following steps:

- The E-MTA MUST issue a DSA_Request message to the CMTS for the upstream direction. The DSA-REQ message MUST include the SCN defined in the "pktcSigServiceClassNameUS" MIB Object. If the Service Class Name for the upstream direction is an empty string than the MTA MUST NOT request creation of a service flow.
- The E-MTA MUST issue a DSA_Request message to the CMTS for the Downstream Direction. The DSA-REQ message MUST include the SCN defined in the "pktcSigServiceClassNameDS" MIB Object. [1]
- If any of the above steps result in an error then "pktcSigNcsServiceFlowState" MIB Object MUST be set to the "error" state. An Event Log SHOULD be sent to the SYSLOG Server.

Each DSA message MUST include the resolved IP address for the CMS(s) in the 'IP Destination Address' and 'IP Source Address' for the upstream and downstream service flow creation respectively. A Call Signaling Service flow MAY contain more than one classifier.

[1] DSA-Request for upstream and downstream may be combined as in [18].

7.9 MTA Replacement

PacketCable 1.0 has no requirement to specify MTA replacement procedures. However, the provisioning sequence flows detailed within this document provide sufficient coverage and flexibility to support replacement. In fact, the initialization sequence for a replacement MTA could be the same as the original MTA's first time initialization. Back office procedures related to migration of subscriber profiles from one MTA to another are specific to individual service provider's network operations. As a result of this wide variance, discussion of these back office procedures are beyond the scope of PacketCable 1.0.

7.10 Temporary Signal Loss

If the CM or DOCSIS reset for any reason, the MTA MUST reset and reinitialize (this will impact calls in progress).

8 DHCP OPTIONS

DHCP is used to obtain IPv4 addresses for both the CM and the MTA. The CM and MTA requirements for DHCP Option Codes 122 and 60 are detailed in section 8.1 and 8.2.

8.1 DHCP Option 122: CableLabs Client Configuration Option

DHCP option code 122 is the RFCed replacement for the former option 177 (which was intended as a temporary code).

DHCP option code 122 is used in both the CM and MTA DHCP OFFER/ACK messages to provide the addresses of valid PacketCable network servers and various device configuration data.

Full details of DHCP option 122 encoding can be found in [18] and [34].

The following sections provide additional semantic details of each suboption in DHCP option 122.

Option	Sub-option	Description and Comments	Sub-option Required or Optional	Default Value
122	1	Service Provider's Primary DHCP Server Address Required by CM only.	Required	N/A
	2	Service Provider's Secondary DHCP Server Address Optional requirement for CM.	Optional	Empty String
	3	Service Provider's Provisioning Entity Address	Required	N/A
	4	AS-REQ/REP Exchange Backoff and Retry for SNMPv3 Key Management	Optional	As per the following MIB Objects: "pktcMtaDevRealmUnsolicitedKeyNomTimeout", "pktcMtaDevRealmUnsolicitedKeyMaxTimeout", "pktcMtaDevRealmUnsolicitedKeyMaxRetries"
	5	AP-REQ/REP Kerberized Provisioning Backoff and Retry	Optional	As per the following MIB Objects: "pktcMtaDevProvUnsolicitedKeyNomTimeout" "pktcMtaDevProvUnsolicitedKeyMaxTimeout" "pktcMtaDevProvUnsolicitedKeyMaxRetries"
	6	Kerberos Realm of SNMP Entity	Required	N/A
	7	Ticket Granting Server Usage	Optional	N/A – if MTA does not implement TGT. 0 – otherwise.

Option	Sub-option	Description and Comments	Sub-option Required or Optional	Default Value
	8	Provisioning Timer	Optional	As per "pktcMtaDevProvisioningTimer" MIB Object (10 minutes)
	9	Security Ticket Invalidation	Optional	0 – apply normal ticket invalidation rules per [5]

MTA MUST be able to retrieve and process the data from all sub-options in the above table. Provisioning Server MUST supply to the MTA all "required" sub-options and MAY supply all "optional" sub-options.

If an "optional" sub-option is not supplied by the Provisioning Server, the MTA MUST use the default value of the sub-option.

If the "required" sub-option is not supplied by the Provisioning Server, the MTA MUST reject the corresponding DHCP OFFER/ACK.

If the sub-option contains wrong (invalid) value, the MTA MUST:

- reject the corresponding DHCP OFFER/ACK in case of "required" sub-option
- use the default value in case of "optional" sub-option

An MTA MUST ignore any other sub-option in Option-122 except those listed in the above table.

8.1.1 Service Provider's DHCP Address (sub-option 1 and 2)

The Service Provider's DHCP Server Addresses identify the DHCP servers that a DHCP OFFER will be accepted from in order to obtain an MTA-unique IP address for a given service provider's network administrative domain.

The encoding of these sub-options is defined in [18].

Sub-option 1 MUST be included in the DHCP OFFER/ACK to the CM and it indicates the Primary DHCP server's IP address. The value contained in sub-option 1 MUST be a valid IP address, a value of 255.255.255.255 or a value of 0.0.0.0. The value contained in sub-option 2 MUST be a valid IP address.

The MTA MUST follow the logic in the table below when defining its DHCP strategy:

Suboption-1	Suboption-2	
	Valid IP – Available	Valid IP – Unavailable
Valid IP – Available	MTA MUST accept DHCP OFFERs coming only from the IP Address in the suboption-1.	MTA MUST accept DHCP OFFERs coming only from the IP Address in the suboption-1.
Valid IP – Unavailable	MTA MUST try exponentially at least three times before accepting the DHCP OFFER coming from the DHCP Server pointed out by Sub-option-2.	MTA MUST return to MTA-1 step.
255.255.255.255	MUST select the OFFERs according to the logic of RFC2131. Value in the sub-option-2 MUST be ignored	MTA MUST select the OFFERs according to the logic of RFC2131. Value in the sub-option-2 MUST be ignored.
0.0.0.0	MTA MUST stop all provisioning attempts as well as all other activities.	MTA MUST stop all provisioning attempts as well as all other activities.

8.1.2 Service Provider's Provisioning Entity Address (sub-option 3)

The Service Provider's Provisioning Entity Address is the network address of the provisioning server for a given voice service provider's network administrative domain.

The encoding of this sub-option is defined in [18]. This address MUST be configured as an FQDN only.

An FQDN value of 0.0.0.0 in suboption 3 of a valid MTA DHCP OFFER/ACK specifies that the MTA MUST shutdown and not try to provision unless it is reinitialized by the CM. This is explained in step MTA2 of the provisioning flow process of Section 7.2.

The Service Provider's Provisioning Entity Address component MUST be capable of accepting SNMP traps.

Sub-option 3 MUST be included in the DHCP offer to the MTA.

8.1.3 AS-REQ/REP Exchange Backoff and Retry for SNMPv3 Key Management (sub-option 4)

The MTA MUST use the DHCP option 122 sub-option 4, if supplied in Secure Flow only. AS-REQ/REP exchange backoff and retry mechanism of the Kerberized SNMPv3 key negotiation defined in [5] is controlled by the values delivered in this sub-option or by the default values of the corresponding MIB objects in the Realm Table if this sub-option is not present in the DHCP Option 122.

The encoding of this sub-option is defined in [18].

The sub-option's nominal timeout value corresponds to the pktcMtaDevRealmUnsolicitedKeyNomTimeout MIB object in the pktcMtaDevRealmTable.

The sub-option's maximum timeout value corresponds to the pktcMtaDevRealmUnsolicitedKeyMaxTimeout MIB object in the pktcMtaDevRealmTable.

The sub-option's max retry count corresponds to the pktcMtaDevRealmUnsolicitedKeyMaxRetries MIB object in the pktcMtaDevRealmTable.

An MTA MUST be able to retrieve the above parameters from this sub-option, if they are supplied by the Provisioning Server.

Provisioning Server MAY provision an MTA with the above parameters using this sub-option.

If any of the values defined in this suboption are "FFFFFFFF" (hexadecimal) then the default value of the corresponding column from the Realm Table MUST be used.

8.1.4 AP-REQ/REP Kerberized Provisioning Backoff and Retry (sub-option 5)

The MTA MUST use the DHCP option 122 sub-option 5, if supplied in Secure Flow only. AP-REQ/REP backoff and retry mechanism of the Kerberized SNMPv3 key negotiation defined in security [5] is controlled by the values delivered by this sub-option.

The encoding of this sub-option is defined in [18].

The sub-option's nominal timeout value corresponds to the pktcMtaDevProvUnsolicitedKeyNomTimeout MIB object.

The sub-option's maximum timeout value corresponds to the pktcMtaDevProvUnsolicitedKeyMaxTimeout MIB object.

The sub-option's max retry count corresponds to the pktcMtaDevProvUnsolicitedKeyMaxRetries MIB object.

An MTA MUST be able to retrieve the above parameters from this sub-option, if they are supplied by the Provisioning Server.

Provisioning Server MAY provision an MTA with the above parameters using this sub-option.

If any of the values defined in this suboption are “FFFFFFFF” (hexadecimal) then the default value of the corresponding MIB Object MUST be used.

8.1.5 Kerberos Realm of SNMP Entity (sub-option 6)

In conjunction with the Provisioning Entity Address, the Kerberos Realm is used as a means of contacting a SNMP Entity in the provisioning realm. The realm name is used to perform a DNS SRV lookup for the realm's KDC.

The DHCP option 122 sub-option 6 MUST be included in the DHCP OFFER to the MTA. For the Secure Flow, the DHCP option 122 sub-option 6 MUST only contain the realm name in the format of FQDN (type=0 as per [18]).

The MTA MUST select the corresponding Provisioning Flow as per the following table (the DHCP option 122 sub-option 6 content comparison is case-sensitive and MUST be in all capital letters).

Table 1. MTA Device Provisioning Flow Selection

Content of the DHCP option 122 sub-option 6	MTA Device Provisioning Flow Selection
BASIC.1	If the DHCP option 122 sub-option 6 value is BASIC.1, the MTA MUST execute the Basic flow without the provisioning complete SNMP INFORM.
BASIC.2	If the DHCP option 122 sub-option 6 value is BASIC.2, the MTA MUST execute the Basic flow with the provisioning complete SNMP INFORM.
HYBRID.1	If the DHCP option 122 sub-option 6 value is HYBRID.1, the MTA MUST execute the Hybrid flow without the provisioning complete SNMP INFORM.
HYBRID.2	If the DHCP option 122 sub-option 6 value is HYBRID.2, the MTA MUST execute the Hybrid flow with the provisioning complete SNMP INFORM.

The MTA MUST use the Secure Flow if any other value is provided in the DHCP option 122 sub-option 6. For Secure Flow, the encoding of the DHCP option 122 sub-option 6 is defined in [18]].

8.1.5.1 SNMPv3 Key Establishment

The SNMPv3 Key Establishment is applicable for Secure Flow only. The AP Request/AP Reply described in Figure 6 the accompanying flow description, and the security specification are used by the MTA in the initial provisioning phase to establish keys with the SNMPv3 USM User “MTA-Prov-xx:xx:xx:xx:xx:xx”. Where xx:xx:xx:xx:xx:xx represents the MAC address of the MTA and MUST be uppercase. The MTA MUST instantiate this user in the USM MIB described in RFC 3414 [8], with the ability to be keyed using the PacketCable Kerberized key management method described in the security specification. SNMPv3 authentication is required and privacy is optional. For the list of allowed SNMPv3 authentication and privacy algorithms see [5].

Additionally, the usmUserSecurityName MUST be the set to the string “MTA-Prov-xx:xx:xx:xx:xx:xx” (quotation marks not included). Where xx:xx:xx:xx:xx:xx represents the MAC address of the MTA and MUST be uppercase. This ensures a unique usmUserSecurityName is created for each MTA.

The MTA must first obtain a service ticket for the provisioning realm as described in step MTA9. USM key management is performed over UDP, as specified in [5]. The SNMPv3 keys are established prior to any SNMPv3 communication and therefore SNMPv3 messages MUST be authenticated at all times (with privacy being optional). The MTA MUST use the USM user created above in the initial INFORM.

8.1.6 Ticket Granting Server Usage (sub-option 7)

The MTA MUST use the DHCP option 122 sub option 6 if supplied for the provisioning kerberized key management in Secure Flow only. This sub-option contains a Boolean, which when true, indicates that the MTA SHOULD get its TGT (ticket granting ticket).

Sub-option 7 MAY be included in the DHCP OFFER/ACK to the MTA.

The encoding of this sub-option is defined in [18].

8.1.7 Provisioning Timer (sub-option 8)

Sub-option 8 defines the value to be used for the provisioning timer. Sub-option 8 MAY be included in the DHCP OFFER/ACK to the MTA.

The encoding of this sub-option is defined in [18].

8.1.8 Security Ticket Invalidation (sub-option 9)

Sub-option 9 contains a bit mask that directs the MTA to invalidate specific application server security tickets. Sub-option 9 MAY be included in the DHCP OFFER/ACK to the MTA. The encoding of this sub-option is defined in [34].

8.2 DHCP Option 60: Vendor Client Identifier

Option code 60 contains a string identifying Capabilities of the MTA. The MTA- MUST send the following ASCII Coded String in DHCP Option code 60: “pktc1.0:xxxxxx”. Where xxxxxx MUST be an ASCII representation of the hexadecimal encoding of the MTA TLV Encoded Capabilities, as defined in Section 10.

8.3 DHCP Options 12 and 15

MTA FQDN MUST be sent to the E-MTA in Option-12 and Option-15. Option-12 MUST contain “Host Name” part of the FQDN, and the Option-15 MUST contain “Domain Name” part of the FQDN.

For example, if MTA FQDN is “mta1.pclab.com”, then Option-12 must contain “mta1” and Option-15 must contain “pclab.com”.

8.4 DHCP Option 6

DHCP Option 6 MUST be used to provide the MTA with its list of DNS server addresses. Option 6 MUST contain at least one DNS server address. Option 6 MAY contain a secondary DNS server address. If this option contains more than two DNS servers, the MTA MUST use the first two addresses.

8.5 DHCP Option 43

The MTA MUST send the DHCP Option 43 in the DHCP DISCOVER and DHCP REQUEST for the Secure, Hybrid and Basic Flows.

DHCP Option 43 contains the number of sub-options defined to provide the MTA device specific information to the back-office systems. The DHCP option 43 sub-options 1 through 10, 31 and 32 are specified by PacketCable, sub-options 11-30 are reserved for the CableLabs CableHome project, sub-options 33 through 50 are reserved for PacketCable, sub-options 51 through 127 are reserved for future CableLabs use, and sub-options 128 and above are reserved for vendor use. The PacketCable DHCP option 43 sub-options MUST be present in the format of "Encapsulated vendor-specific extensions" ([RFC2132]).

The following table contains the sub-options of the DHCP Option-43, which the MTA MUST use. The MTA MUST send all required sub-options listed in the table below unless explicitly stated otherwise. If the total number of octets in all DHCP option 43 sub-options exceeds 255 octets, the MTA MUST follow RFC 3396 [10] to split the option into multiple smaller options.

Table 2. DHCP Option 43 Syntax

MTA DHCP Option 43 Sub-options	Required / Not Used in OPTION-43	Value	Description
Sub-option 1	Not Used		The request sub-option vector is a list of sub-options (within option 43) to be returned to client by the server upon reply to the request. None defined. The DHCP option 43 sub-option 1 MUST NOT be used by the MTA, and if present, it MUST be ignored by the Provisioning Server
Sub-option 2	R	<DevType>	The sub-option 2 contains the device type of the component making the DHCP request. The MTA MUST send the DHCP option 43 sub-option 2. For PacketCable MTAs, the allowable device types are: - "EMTA" - for E-MTAs - "SMTA" - for S-MTAs
Sub-option 3	Not Used		The sub-option 3 contains a colon separated list of all components in the eDOCSIS device. It is used by the eDOCSIS eCM device. The DHCP option 43 sub-option 3 MUST NOT be sent by the MTA, and if present, it MUST be ignored by the Provisioning Server.
Sub-option 4	R	<device serial number>	The sub-option 4 contains the device serial number represented as an ASCII string. The MTA MUST send the DHCP option 43 sub-option 4. The DHCP option 43 sub-option 4 value MUST be identical to the value of the pktcMtaDevSerialNumber MIB Object.
Sub-option 5	R	<Hardware version>	The sub-option 5 contains the hardware version number represented as an ASCII string. The MTA MUST send the DHCP option 43 sub-option 5. The DHCP option 43 sub-option 5 MUST be identical to the value of the Hardware

			version number as in <Hardware version> field in the MIB II object sysDescr.
Sub-option 6	R	<Software version>	The sub-option 6 contains the software version number represented as an ASCII string. The MTA MUST send the DHCP option 43 sub-option 6. The DHCP option 43 sub-option 6 value MUST be identical to the value of the pktcMtaDevSwCurrentVers MIB object.
Sub-option 7	R	<Boot ROM Version>	The sub-option 7 contains the Boot ROM Version represented as an ASCII string. The MTA MUST send the DHCP option 43 sub-option 7. The DHCP option 43 sub-option 7 value MUST be identical to the <Boot ROM version> field in MIB II object sysDescr.
Sub-option 8	R	<OUI>	The sub-option 8 contains the Organizational Unique Identifier (OUI) represented as a hexadecimal-encoded 3-byte octet string. It MAY match the OUI in the MTA MAC address. The MTA MUST send the DHCP option 43 sub-option 8. If omitted, the Provisioning Server SHOULD use the MTA MAC address as the MTA OUI.
Sub-option 9	R	<Model Number>	The sub-option 9 contains the MTA Device Model Number represented as an ASCII string. The MTA MUST send the DHCP option 43 sub-option 9. The DHCP option 43 sub-option 9 value MUST be identical to <Model Number> field in the MIB-II object sysDescr.
Sub-option 10	R	<Vendor Name>	The sub-option 10 contains the Vendor Name represented as an ASCII string. The MTA MUST send the DHCP option 43 sub-option 10. The DHCP option 43 sub-option 10 value MUST be identical to <Vendor Name> field in the MIB-II object sysDescr.
Sub-options 11 –30			Reserved for CableHome
Sub-option 31	R	<MTA MAC Address>	The sub-option 31 contains the MTA MAC Address encoded as a 6 byte octet string. The MTA MUST send the DHCP option 43 sub-option 31. The DHCP option 43 sub-option 31 value MUST be identical to the content of the pktcMtaDevMacAddress MIB object.
Sub-option 32	R	<Correlation ID>	The sub-option 32 contains the Correlation ID number encoded as 4-byte INTEGER in the network order. The MTA MUST send the DHCP option 43 sub-option 32.

			The DHCP option 43 sub-option 32 value MUST be identical to the content of the pktcMtaDevCorrelationId MIB object.
Sub-options 33-50			Reserved for PacketCable.
Sub-options 51 to 127			Reserved for CableLabs.
Sub-options 128 to 254			Reserved for vendors.

9 MTA PROVISIONABLE ATTRIBUTES

This section includes the list of attributes and their associated properties used in device provisioning. All of the provisionable attributes specified in this section MAY be updated via the MTA configuration data file, or on a per-attribute basis using SNMP.

PacketCable 1.0 requires that a MTA configuration data file MUST be provided to all embedded-MTAs during the registration sequence. Endpoint voice services do not have to be enabled at the time of initialization. MTA device level configuration data MUST be provisioned during initialization. These items are contained in section 9.1.1.

The MTA configuration data URL generated by the Provisioning Application MUST be less than 255 bytes in length and cannot be NULL. Since this filename is provided to the MTA by the Provisioning Application during the registration sequence, it is not necessary to specify a file naming convention.

9.1 MTA Configuration File

The following is a list of attributes and their syntax for objects included in the MTA configuration file. This file contains a series of "type length and value" (TLV) parameters. Each TLV parameter in the configuration file describes an MTA or endpoint attribute. The configuration data file includes TLVs that have read-write, read only, and no MIB access. Unless specifically indicated, all MIB-accessible configuration file parameters MUST be defined using the DOCSIS TLV type 11, the PacketCable type 64, or PacketCable TLV type 38. TLV 64 is a PacketCable defined TLV where the length value is 2 bytes long instead of the 1 byte for DOCSIS TLV type 11. The TLV type 64 MUST be used when the length is greater than 254 bytes. If desired, vendor-specific information may be added to the configuration file using the vendor-specific TLV43. This TLV has been specified by the DOCSIS specification [6]. Vendors MUST NOT provision vendor-specific information using TLV type 11 or 64. TLV 38 is a PacketCable defined TLV, analogous to TLV-38 used by DOCSIS and CableHome. The MTA MUST be able to process the TLVs given in the following table:

Type	Length	Value
11	n, where n is 1 byte	variable binding
64	m, where m is 2 bytes	variable binding
38	n, where n is 1 byte	Composite (Contains sub TLVs)
254	1 byte	0xFE for beginning of the file and0xFF for the end of the file
NOTE: The use of TLV type 11 rather than TLV 64 is recommended wherever possible.		

In the future, new TLVs introduced in PacketCable must have a "length field" size 2 bytes.

The VarBind is encoded in ASN.1 Basic Encoding Rules, just as it would be if part of an SNMP Set request.

The MTA configuration file MUST start with the "telephony configuration file start" tag and MUST end with the "telephony configuration file end" tag. These tags enable the MTA TLV parameters to be distinguished from DOCSIS TLV parameters. These tags also provide deterministic indications for start and stop of the MTA configuration file.

The MTA configuration file MUST contain the attributes identified as "required" in the Device Level Configuration Data table, which appears in section 9.1.1; failing which, the MTA MUST reject the configuration file and take the necessary steps as defined in section 7.2 (failure of step MTA 23 due to 'Configuration file error'). The MTA configuration file MAY contain any of the non-required attributes which appear in the Device Level Configuration Data table. If the configuration file does not contain required attributes, it MUST be rejected. The MTA configuration file MUST be sent to the embedded-MTA every time this device is powered on.

The Device Level Service Data MAY be sent to the MTA as part of the MTA configuration file or it MAY be sent to the MTA using SNMP. If included in the configuration file it MUST contain all of attributes identified as 'required' in the Device Level service data, if any. The MTA configuration file MAY additionally contain any of the non-required attributes that appear in the Device Level Service Data table.

If voice services are required on the MTA on any endpoint, the following MUST be done:

1. pktcMtaDevEnabled MUST be set to TRUE,
2. Per endpoint configuration data MUST be supplied either through the MTA configuration file (during provisioning) or through end-point provisioning (using SNMP) in the post-provisioning phase.

The End point details, when included MUST contain the attributes identified as "required" in the Per-Endpoint Configuration Data table, which appears in section 9.1.3. The MTA configuration file MAY contain any of the non-required attributes which appear in the Per-Endpoint Configuration Data table in section 9.1.3. The Per-Endpoint Configuration Data MUST be sent to the MTA when voice communications service is activated.

It is to be noted that the Device Level Service Data and Per-Endpoint Configuration Data MAY also be sent to the MTA via incremental provisioning, using SNMP. The MTA MUST support incremental provisioning.

The MTA MUST be able to process all TLV11 and TLV64 values with variable bindings containing all MIB objects defined in [2].

The Device Level Configuration data parameter 'pktcMtaDevEnabled' is used to actually enable or disable voice services on an MTA.

Refer to section 7.6.1 for a discussion concerning synchronization of provisioning attributes with back office systems.

For the Secure and Hybrid Provisioning Flows, the MTA MUST authenticate the configuration file according to PacketCable Security Specification [5] ; the MTA MUST reject the configuration file if the configuration file authentication fails and take the necessary steps as defined in section 7.2 for the Secure Flow and Section 7.4 for the Hybrid Flow. If the configuration file contains the MIB object 'pktcMtaDevProvConfigHash' in the Secure Flow or the Hybrid Flow, the MTA MUST ignore the value of this MIB object and proceed with further processing of the configuration file and report passWithWarnings and populate the Error OID table (pktcMtaDevErrorOidsTable).

For the Basic Flow, the Provisioning Server and the MTA MUST support the configuration file data verification process as described below:

1. When the Provisioning Server creates a new MTA Configuration File or modifies an existing one, to be served for an MTA intended to go through the Basic Flow, it MUST calculate a SHA-1 hash value of the contents of the entire MTA Configuration File including start and end markers, taken as a byte string.
2. The Provisioning Server MUST add the hash value, calculated in Step 1 to the MTA Configuration File as a TLV-11 triplet corresponding to the 'pktcMtaDevProvConfigHash' MIB Object. The Provisioning Server MUST insert the TLV11 triplet before the Configuration file end-marker. The Provisioning Server MUST NOT change the order of the TLVs in the configuration file after the hash has been calculated. The MTA Configuration File is then made available to the MTA through the appropriate TFTP/HTTP server.

3. Upon receiving the configuration file, the MTA MUST do the following:

   If the MIB object 'pktcMtaDevProvConfigHash' is absent, MTA MUST reject the configuration file and MUST report 'failOtherReason'.

   If the MIB object 'pktcMtaDevProvConfigHash' is present, then MTA MUST:

   a. Calculate SHA-1 over the contents of the file without TLV-11 triplet containing the pktcMtaDevProvConfigHash' and MUST populate the calculated value into pktcMtaDevProvConfigHash mib object. The MTA must maintain the order of the TLVs for the hash calculation to be correct.

   b If the computed hash and the value of the 'pktcMtaDevProvConfigHash' MIB object are the same, the MTA Configuration File integrity is verified and the MTA MUST accept the configuration file for further processing ; otherwise, the MTA MUST reject the Configuration File and the MTA MUST report 'failOtherReason'.

The MTA must also check for errors in the configuration file. As described above, errors in any of the mandatory parameters MUST be treated as an error in the configuration file and appropriate steps taken (failure of step MTA 23 due to 'Configuration file error').

If there are errors in the non-required OIDs then the MTA MUST accept the configuration file, but report the same in the status (MTA-25).

If the Configuration file contains per-cms data and per-endpoint parameters related to CMSs which are not associated to endpoints, an MTA MUST NOT establish SAs till and end-point gets associated with that particular CMS (either using SNMP or via NCS redirection).

The MTA MUST report the state of the configuration file it received in the 'Provisioning complete Inform' (step MTA25 in the provisioning process) as given below:

- If the configuration file could be parsed successfully and the MTA is able to reflect the same in its MIB, it must return: 'pass'.
- If the configuration file was in error due to incorrect values in the mandatory parameters, the MTA MUST reject the configuration file and return: 'failConfigFileError'.

  It MUST also populate '`pktcMtaDevErrorOidsTable`' with the parameter containing the incorrect value and MAY also populate it with other OID errors/warnings if it parsed the file completely.
- If the configuration file had proper values for all the mandatory parameters but has errors in any of the optional parameters (this includes any vendor specific OIDs which are incorrect or not known to the MTA) it must return: 'passWithWarnings'.

  It MUST also populate 'pktcMtaDevErrorOidsTable' with a list of all the parameters which were rejected and the reason for the same. The MTA MUST also use the default values for all such parameters, unless they were overridden by some other means like DHCP, in which case it must use the overridden values.
- If the configuration file is proper, but the MTA cannot reflect the same in its MIB (For ex: Too many entries leading to memory exhaustion), it MUST accept details related to the CMSs associated with the endpoints and return: 'passWithIncompleteParsing'.

  It MUST also populate 'pktcMtaDevErrorOidsTable' with a list of all the parameters which cannot be reflected in the MIB.
- If the configuration file cannot be parsed due to an internal error it must return 'failureInternalError'. It SHOULD try to populate 'pktcMtaDevErrorOidsTable' for parameters which lead to failure.
- If the MTA cannot accept the configuration file for any other reason than the ones stated above, it must return 'failureOtherReason'. It SHOULD try to populate 'pktcMtaDevErrorOidsTable' for parameters, which lead to the failure.

The MTA Configuration File MUST contain Per-Realm Configuration Data. Per-Realm Configuration Data MUST contain at least the data for the Provisioning Realm that is identified in DHCP Option-122, suboption-6.

After Receiving the MTA Configuration File, an MTA MUST validate the following:

- "pktcMtaDevRealmName" MIB Object of the Realm Table MUST be the same as the Realm Name supplied to the MTA in DHCP Option-122, suboption-6.
- "pktcMtaDevRealmOrgName" MIB Object of the Realm Table MUST be the same as the "Organization Name" attribute in the Service Provider Certificate.
- Encryption and Authentication of the MTA Configuration File as per [5].

An MTA MUST treat any of the above validation failures as failure of the MTA23 Provisioning Flow and the MTA MUST discard the Configuration File.

If the MTA encounters a vendor-specific TLV43 with a vendor ID that the MTA does not recognize as its own, the MTA must ignore the TLV 43 and the MTA MUST continue to process the configuration file. If the MTA detects the presence of an unrecognized TLV (TLV type other than TLV 11, TLV 43, TLV 64, TLV 38, or TLV254), the MTA MUST ignore the TLV assuming the length field of the unrecognized TLV is 2 bytes and proceed with further processing. The MTA MUST report a provisioning state of passWithWarnings and populate the error OID table (pktcMtaDevErrorOidsTable) if it detects the presence of an unrecognized TLV. If the MTA encounters an unrecognized variable binding in a TLV 11 or TLV 64, it MUST ignore this binding, MUST report a provisioning state of passWithWarnings and populate the error OID table (pktcMtaDevErrorOidsTable). If the MTA encounters unrecognized sub TLVs within TLV 43 or TLV38, it MUST ignore the sub TLVs, continue with further processing and MUST report a provisioning state of passWithWarnings.

The MTA MUST attempt to accept configuration file that contains valid set of per-realm and per-CMS configuration data identified in sections 9.1.4 and 9.1.5 even if the MTA endpoints are not associated with the CMS in the per-CMS configuration data.

The MIB objects of type RowStatus MUST NOT be included in the MTA configuration file. The MTA MUST ignore any MIB object of type RowStatus and include 'passWithWarnings' in the final SNMP inform message (MTA-25). The Error OID table (pktcMtaDevErrorOidsTable in the MTA MIB) MUST be populated with the RowStatus OID.

9.1.1 Device Level Configuration Data

Refer to the MTA MIB [2] for more detailed information concerning these attributes and their default values.

- The MTA Manufacturer Certificate validates the MTA Device Certificate.

Attribute	Syntax	Configuration Access	SNMP Access	MIB File	Object	Comments
Telephony Config File Start	Integer	W, required	None	N/A	N/A	Type length value 254 1 1 The MTA config file MUST start with this attribute.
Telephony Config File End	Integer	W, required	None	N/A	N/A	Type length value 254 1 255 This MUST be the last attribute in the MTA config file.
Telephony MTA Admin State	ENUM	W, required	R/W	MTA Device MIB	pktcMtaDevEnabled	Used to enable/disable all telephony ports on the MTA. Applies to the MTA side of the embedded-MTA or the entire stand-alone MTA. Allows blanket management of all telephony ports (external interfaces) on the device. The state of the MTA is controlled by this MIB Object. For more information about this object, refer to the MTA MIB [2]. .
Realm Organization Name	String	W, Required	R/W	MTA Device MIB	pktcMtaDevRealmOrgName	The value of the X.500 name organization name attribute in the subject name of the service provider certificate for MSO KDC.
Solicited Key Timeout	Integer	W, optional	R/W	N/A	pktcMtaDevProvSolicitedKeyTimeout	This timeout applies only when the Provisioning Server initiated key management (with a Wake Up message) for SNMPv3. It is the period during which the MTA will save a nonce (inside the sequence number field) from the sent out AP Request and wait for the matching AP Reply from the Provisioning Server. Since there is a default value, this is optional.
Reset Kerberos ticket information	Integer32	W, optional	R/W	MTA Device	pktcMtaDevResetKrbTickets	Security Specification [5] allows the Kerberos tickets associated with any of the application server (Provisioning

Attribute	Syntax	Configuration Access	SNMP Access	MIB File	Object	Comments
				MIB		Server or CMS) to be stored in the MTA NVRAM until ticket expiry. In order to control the invalidation of the tickets stored in NVRAM, this MIB attribute is used to communicate the required action to the MTA. Upon receiving this attribute in the config file, an MTA MUST take the specified action. Refer to [2] for more information.

9.1.2 Device Level Service Data

Refer to the MTA MIB [2], the SIGNALING MIB [3], the NCS Call Signaling specification [4], and RFC 2475 [31] for more detailed information concerning these attributes and their default values.

Attribute	Syntax	Configuration Access	SNMP Access	MIB File	Object	pktcDevEvSysloComments
NCS Default Call Signaling TOS	Integer	W, optional	R/W	MTA Signaling MIB	pktcSigDefCallSigTos	The default value used in the IP header for setting the TOS value for NCS call signaling.
NCS Default Media Stream TOS	Integer	W, optional	R/W	MTA Signaling MIB	pktcSigDefMediaStreamTos	The default value used in the IP header for setting the TOS value for NCS media stream packets.
MTA UDP receive port used for NCS	Integer (1025..65535)	W, optional	R/O	MTA Signaling MIB	pktcSigDefNcsReceiveUdpPort	This object contains the MTA User Datagram Protocol Receive Port that is used for NCS call signaling. This object should only be changed by the configuration file.
NCS TOS Format Selector	ENUM	W, optional	R/W	MTA Signaling MIB	pktcSigTosFormatSelector	The format of the default NCS signaling and media TOS values. Allowed values are “IPv4 TOS octet” or “DSCP codepoint”. Refer to IETF RFC 2475.

Attribute	Syntax	Configuration Access	SNMP Access	MIB File	Object	pktcDevEvSysloComments
R0 cadence	Bit-field	W, optional	R/W	MTA Signaling MIB	pktcSigDevR0Cadence	User defined field where each bit (least significant bit) represents a duration of 100 milliseconds (6 seconds total) 1= active ringing, 0= silence 64 bits are used for representation; MSB 60 bits for ring cadence. Bit 61 is used to represent repeatable (when set to ZERO) and non repeatable (when set to ONE). Other three bits are reserved for future use, and currently set to 000.
R6 cadence	Bit-field	W, optional	R/W	MTA Signaling MIB	pktcSigDevR6Cadence	User defined field where each bit (least significant bit) represents a duration of 100 milliseconds (6 seconds total) 1= active ringing, 0= silence 64 bits are used for representation; MSB 60 bits for ring cadence. Bit 61 is used to represent repeatable (when set to ZERO) and non repeatable (when set to ONE). Other three bits are reserved for future use, and currently set to 000.
R7 cadence	Bit-field	W, optional	R/W	MTA Signaling MIB	pktcSigDevR7Cadence	User defined field where each bit (least significant bit) represents a duration of 100 milliseconds (6 seconds total) 1= active ringing, 0= silence 64 bits are used for representation; MSB 60 bits for ring cadence. Bit 61 is used to represent repeatable (when set to ZERO) and non repeatable (when set to ONE). Other three bits are reserved for future use, and currently set to 000.
R1 cadence	Bit-field	W, optional	R/W	MTA Signaling MIB	pktcSigDevR1Cadence	User defined field where each bit (least significant bit) represents a duration of 100 milliseconds (6 seconds total)

Attribute	Syntax	Configuration Access	SNMP Access	MIB File	Object	pktcDevEvSysloComments
						1= active ringing, 0= silence 64 bits are used for representation; MSB 60 bits for ring cadence. Bit 61 is used to represent repeatable (when set to ZERO) and non repeatable (when set to ONE). Other three bits are reserved for future use, and currently set to 000.
R2 cadence	Bit-field	W, optional	R/W	MTA Signaling MIB	pktcSigDevR2Cadence	User defined field where each bit (least significant bit) represents a duration of 100 milliseconds (6 seconds total) 1= active ringing, 0= silence 64 bits are used for representation; MSB 60 bits for ring cadence. Bit 61 is used to represent repeatable (when set to ZERO) and non repeatable (when set to ONE). Other three bits are reserved for future use, and currently set to 000.
R3 cadence	Bit-field	W, optional	R/W	MTA Signaling MIB	pktcSigDevR3Cadence	User defined field where each bit (least significant bit) represents a duration of 100 milliseconds (6 seconds total) 1= active ringing, 0= silence 64 bits are used for representation; MSB 60 bits for ring cadence. Bit 61 is used to represent repeatable (when set to ZERO) and non repeatable (when set to ONE). Other three bits are reserved for future use, and currently set to 000.
R4 cadence	Bit-field	W, optional	R/W	MTA Signaling MIB	pktcSigDevR4Cadence	User defined field where each bit (least significant bit) represents a duration of 100 milliseconds (6 seconds total) 1= active ringing, 0= silence 64 bits are used for representation; MSB 60 bits for ring cadence. Bit 61 is used to represent repeatable (when set to ZERO) and non repeatable (when set to

Attribute	Syntax	Configuration Access	SNMP Access	MIB File	Object	pktcDevEvSysloComments
						ONE). Other three bits are reserved for future use, and currently set to 000
R5 cadence	Bit-field	W, optional	R/W	MTA Signaling MIB	pktcSigDevR5Cadence	User defined field where each bit (least significant bit) represents a duration of 100 1= active ringing, 0= silence 64 bits are used for representation; MSB 60 bits for ring cadence. Bit 61 is used to represent repeatable (when set to ZERO) and non repeatable (when set to ONE). Other three bits are reserved for future use, and currently set to 000.
Rg cadence	Bit-field	W, optional	R/W	MTA Signaling MIB	pktcSigDevRgCadence	User defined field where each bit (least significant bit) represents a duration of 100 milliseconds (6 seconds total) 1= active ringing, 0= silence 64 bits are used for representation; MSB 60 bits for ring cadence. Bit 61 is used to represent repeatable (when set to ZERO) and non repeatable (when set to ONE). Other three bits are reserved for future use, and currently set to 000.
Rt cadence	Bit-field	W, optional	R/W	MTA Signaling MIB	pktcSigDevRtCadence	User defined field where each bit (least significant bit) represents a duration of 100 milliseconds (6 seconds total) 1= active ringing, 0= silence 64 bits are used for representation; MSB 60 bits for ring cadence. Bit 61 is used to represent repeatable (when set to ZERO) and non repeatable (when set to ONE). Other three bits are reserved for future use, and currently set to 000.
Rs cadence	Bit-field	W, optional	R/W	MTA Signaling	pktcSigDevRsCadence	User defined field where each bit (least significant bit) represents a duration of 100 milliseconds

Attribute	Syntax	Configuration Access	SNMP Access	MIB File	Object	pktcDevEvSysloComments
				MIB		1= active ringing, 0= silence 64 bits are used for representation; MSB 60 bits for ring cadence. Bit 61 is used to represent repeatable (when set to ZERO) and non repeatable (when set to ONE). Other three bits are reserved for future use, and currently set to 000.
Call Signaling SCN Up	String	W	R/W	MTA Signaling MIB	pktcSigServiceClassNameUS	The string contains the Service Class Name that is to be used when the Service Flow is created for the upstream direction.
Call Signaling SCN Down	String	W	R/W	MTA Signaling MIB	pktcSigServiceClassNameDS	The string contains the Service Class Name that is to be used when the Service Flow is created for the downstream direction.
Call Signaling Network Mask	Integer32	W	R/W	MTA Signaling MIB	pktcSigServiceClassNameMask	The value is used as the NCS Call Signaling classifier mask.

9.1.3 Per-Endpoint Configuration Data

Refer to the SIGNALING MIB [3], the NCS spec [4], the security spec [5] and the MTA MIB [2] for more detailed information concerning these attributes and their default values.

MTA sends KDC the MTA/CMS certificate, MTA's FQDN, CMS-ID. The KDC returns the MTA a "Kerberos Ticket" that says "this MTA is assigned to this CMS".

The Telephony Service Provider Certificate validates the MTA Telephony Certificate

If two different endpoints share the same Kerberos Realm and same CMS FQDN, then these four attributes MUST be identical: PKINIT grace period, KDC name list, MTA telephony certificate, telephony service provider certificate.

Attribute	Syntax	Access	SNMP Access	MIB File	Object	Comments
Port Admin State	ENUM	W, optional	R/W	IF-MIB (RFC 2863)	ifAdminStatus	The administrative state of the port the operator can access to either enable or disable service to the port. The administrative state can be used to disable access to the user port without de-provisioning the subscriber. Allowed values for this attribute are: up(1) or down(2). For SNMP access ifAdminStatus is found in the ifTable of MIB-II.
Call Management Server Name	String	W, required	R/W	MTA Signaling MIB	pktcNcsEndPntConfigCallAgentId	This attribute is a string indicating the name of the CMS assigned to the endpoint. The call agent name after the character '@' MUST be a fully qualified domain name and MUST have a corresponding conceptual row in the pktcMtaDevCmsTable. DNS support is assumed to support multiple CMS's as described in the NCS spec.
Call Management Server UDP Port	Integer	W	R/W	MTA Signaling MIB	pktcNcsEndPntConfigCallAgentUdpPort	UDP port for the CMS.
Partial Dial Timeout	Integer	W	R/W	MTA Signaling MIB	pktcNcsEndPntConfigPartialDialTO	Timeout value in seconds for partial dial timeout.

Attribute	Syntax	Access	SNMP Access	MIB File	Object	Comments
Critical Dial Timeout	Integer	W	R/W	MTA Signaling MIB	pktcNcsEndPntConfigCriticalDialTO	Timeout value in seconds for critical dial timeout.
Busy Tone Timeout	Integer	W	R/W	MTA Signaling MIB	pktcNcsEndPntConfigBusyToneTO	Timeout value in seconds for busy tone.
Dial tone timeout	Integer	W	R/W	MTA Signaling MIB	pktcNcsEndPntConfigDialToneTO	Timeout value in seconds for dialtone.
Message Waiting timeout	Integer	W	R/W	MTA Signaling MIB	pktcNcsEndPntConfig MessageWaitingTO	Timeout value in seconds for message waiting.
Off Hook Warning timeout	Integer	W	R/W	MTA Signaling MIB	pktcNcsEndPntConfigOffHookWarnToneTO	Timeout value in seconds for off hook warning.
Ringing Timeout	Integer	W	R/W	MTA Signaling MIB	pktcNcsEndPntConfigRingingTO	Timeout value in seconds for ringing.
Ringback Timeout	Integer	W	R/W	MTA Signaling MIB	pktcNcsEndPntConfigRingBackTO	Timeout value in seconds for ringback.
Reorder Tone timeout	Integer	W	R/W	MTA Signaling MIB	pktcNcsEndPntConfigReorderToneTO	Timeout value in seconds for reorder tone.
Stutter dial timeout	Integer	W	R/W	MTA Signaling MIB	pktcNcsEndPntConfigStutterDialToneTO	Timeout value in seconds for stutter dial tone.
TS Max	Integer	W	R/W	MTA Signaling MIB	pktcNcsEndPntConfigTSMax	Contains the maximum time in seconds since the sending of the initial datagram.
Max1	Integer	W	R/W	MTA Signaling MIB	pktcNcsEndPntConfig Max1	The suspicious error threshold for each endpoint retransmission.

Attribute	Syntax	Access	SNMP Access	MIB File	Object	Comments
Max2	Integer	W	R/W	MTA Signaling MIB	pktcNcsEndPntConfig Max2	The disconnect error threshold per endpoint retransmission.
Max1 Queue Enable	Enum	W	R/W	MTA Signaling MIB	pktcNcsEndPntConfig Max1QEnable	Enables/disables the Max1 DNS query operation when Max1 expires.
Max2 Queue Enable	Enum	W	R/W	MTA Signaling MIB	pktcNcsEndPntConfig Max2QEnable	Enables/disables the Max2 DNS query operation when Max2 expires.
MWD	Integer	W	R/W	MTA Signaling MIB	pktcNcsEndPntConfig MWD	Number of seconds to wait to restart after a restart is received.
Tdinit	Integer	W	R/W	MTA Signaling MIB	pktcNcsEndPntConfigTdinit	Number of seconds to wait after a disconnect.
TDMin	Integer	W	R/W	MTA Signaling MIB	pktcNcsEndPntConfigTdmin	Minimum number of seconds to wait after a disconnect.
TDMax	Integer	W	R/W	MTA Signaling MIB	pktcNcsEndPntConfigTdmax	Maximum number of seconds to wait after a disconnect.
RTO Max	Integer	W	R/W	MTA Signaling MIB	pktcNcsEndPntConfigRtoMax	Maximum number of seconds for the retransmission timer.
RTO Init	Integer	W	R/W	MTA Signaling MIB	pktcNcsEndPntConfigRtoInit	Initial value for the retransmission timer.
Long Duration Keepalive	Integer	W	R/W	MTA Signaling MIB	pktcNcsEndPntConfigLongDurationKeepAlive	Timeout in minutes for sending long duration call notification messages.
Thist	Integer	W	R/W	MTA Signaling MIB	pktcNcsEndPntConfigThist	The timeout period in seconds before no response is declared.

Attribute	Syntax	Access	SNMP Access	MIB File	Object	Comments
Call Waiting Max Reps	Integer	W, optional	R/W	MTA Signaling MIB	pktcNcsEndPntConfigCallWaitingMaxRep	This object contains the maximum number of repetitions of the call waiting that the MTA will play from a single CMS request. A value of zero (0) will be used when the CMS invokes any play repetition
Call Waiting Delay	Integer	W, optional	R/W	IF-MIB (RFC 2863)	pktcNcsEndPntConfigCallWaitingDelay	This object contains the delay between repetitions of the call waiting that the MTA will play from a single CMS request.

9.1.4 Per-Realm Configuration Data

Refer to the MTA MIB [2] for more detailed information concerning these attributes and their default values. Refer to the security spec [5] for more information on the use of Kerberos realms. There MUST be at least one conceptual row in the pktcMtaDevRealmTable to establish service upon completion of configuration. These configuration parameters are optional in the config file, but if included the config file MUST contain at least one Realm name to permit proper instantiation of the table. There may be more than one set of entries if multiple realms are supported.

Attribute	Syntax	Access	SNMP Access	MIB File	Object	Comments
Pkinit Grace Period	Integer	W, optional	R/W	MTA Device MIB	pktcMtaDevRealmPkinitGracePeriod	For the purpose of IPSec key management with a CMS, the MTA MUST obtain a new Kerberos ticket (with a PKINIT exchange) this many minutes before the old ticket expires. The minimum allowable value is 15 mins. The default is 30 mins. This parameter MAY also be used with other Kerberized applications.

Attribute	Syntax	Access	SNMP Access	MIB File	Object	Comments
TGS Grace Period	Integer	W, optional	R/W	MTA Device MIB	pktcMtaDevRealmTgsGracePeriod	When the MTA implementation uses TGS Request/TGS Reply Kerberos messages for the purpose of IPSec key management with the CMS, the MTA MUST obtain a new service ticket for the CMS (with a TGS request) this many minutes before the old ticket expires. The minimum allowable value is 1 min. The default is 10 mins. This parameter MAY also be used with other Kerberized applications.
Realm Org Name	Integer	W, required	R/W	MTA Device MIB	pktcMtaDevRealmOrgName	The value of the X.500 organization name attribute in the subject name of the Service provider certificate.
Unsolicited Keying max Timeout	Integer	W, optional	R/W	MTA Device MIB	pktcMtaDevRealmUnsolicitedKeyMaxTimeout	This timeout applies only when the MTA initiated key management. The maximum timeout is the value which may not be exceeded in the exponential backoff algorithm.
Unsolicited Keying Nominal Timeout	Integer	W, optional	R/W	MTA Device MIB	pktcMtaDevRealmUnsolicitedKeyNomTimeout	This timeout applies only when the MTA initiated key management. Typically this is the average roundtrip time between the MTA and the KDC.
Unsolicited Keying Max Retries	Integer	W, optional	R/W	MTA Device MIB	pktcMtaDevRealmUnsolicitedKeyMaxRetries	This is the maximum number of retries before the MTA gives up attempting to establish a Security Association.

9.1.5 Per-CMS Configuration Data

Refer to the MTA MIB [2] for more detailed information concerning these attributes and their default values. There MUST be at least one conceptual row in the pktcDevCmsTable to establish service upon completion of configuration. These configuration parameters are optional in the config file, but if included the config file MUST identify at least one CMS and its corresponding Kerberos Realm Name. There may be more than one set of entries if multiple CMSs are supported.

As per Security Specification Requirement [5], the IPSEC signaling security must be controlled by the Operator depending on the deployment and operational conditions. As the IPSEC Security Association is established between the MTA and the CMS, the IPSEC enabling/disabling control should also be on per CMS basis. Enabling/Disabling of the IPSEC Signaling Security MUST be defined only by the information in the MTA's Configuration File when the file is being downloaded, and the enable/disable toggling MUST be done only as a result of the MTA Reset.

For more details on the MIB Object controlling the IPSEC enabling/disabling, refer to the MTA MIB [2].

Attribute	Syntax	Access	SNMP Access	MIB File	Object	Comments
Kerberos Realm Name	String	W, required*	R/W	MTA Device MIB	pktcMtaDevCmsKerbRealmName	The name for the associated Kerberos Realm. This is the corresponding Kerberos Realm Name in the Per Realm Configuration Data.
CMS Maximum Clock Skew	Integer	W, optional	R/W	MTA Device MIB	pktcMtaDevCmsMaxClockSkew	This is the maximum allowable clock skew between the MTA and CMS.
CMS Solicited Key Timeout	Integer	W, optional	R/W	MTA Device MIB	pktcMtaDevCmsSolicitedKeyTimeout	This timeout applies only when the CMS initiated key management (with a Wake Up or Rekey message). It is the period during which the MTA will save a nonce (inside the sequence number field) from the sent out AP Request and wait for the matching AP Reply from the CMS.
Unsolicited Key Max Timeout	Integer	W, optional	R/W	MTA Device MIB	pktcMtaDevCmsUnsolicitedKeyMaxTimeout	This timeout applies only when the MTA initiated key management. The maximum timeout is the value which may not be exceeded in the exponential backoff algorithm.
Unsolicited Key Nominal Timeout	Integer	W. optional	R/W	MTA Device MIB	pktcMtaDevCmsUnsolicitedKeyNomTimeout	This timeout applies only when the MTA initiated key management. Typically this is the average roundtrip time between the MTA and the CMS.
Unsolicited Key Max Retries	Integer	W, optional	R/W	MTA Device MIB	pktcMtaDevCmsUnsolicitedKeyMaxRetries	This is the maximum number of retries before the MTA gives up attempting to establish a security association.
IPSEC Control	Integer	W, optional	R/O	MTA Device MIB	pktcMtaDevCmsIpsecCtrl	IPSEC Control for each CMS: controls the IPSEC establishment and IPSEC related Key Management.

* If any data from the Per-CMS Data Table is included in the config file, this entry MUST be included.

10 MTA DEVICE CAPABILITIES

MTA Capabilities string is supplied to the Provisioning Server in Option code 60 (Vendor Class Identifier) — to allow the Back-Office to differentiate between MTAs during the Provisioning Process. Use of Capabilities information by the Provisioning Application is optional.

Capabilities string is encoded as an ASCII string containing the Capabilities information in Type/Length/Value (TLV) Format.

For example, the ASCII encoding of the first two TLVs (PacketCable Version 1.0 and Number of Telephony Endpoints = 2) of an MTA would be 05nn010100020102. Note that many more TLVs are required for PacketCable MTA, and the field "nn" will contain the length of all the TLVs. This example shows only two TLVs for simplicity.

The "value" field describes the capabilities of a particular modem, i.e. implementation dependent limits on the particular features or number of features, which the modem can support. It is composed from a number of encapsulated TLV fields. The encapsulated sub-types define the specific capabilities for the MTA. Note that the sub-type fields defined are only valid within the encapsulated capabilities configuration setting string.

Type	Length	Value
5	n	

The set of possible encapsulated fields is described below.

MTA MUST Send Capabilities String in option 60 of the DHCP DISCOVER request.

10.1 PacketCable Version

This TLV MUST be supplied in the Capabilities String.

Type	Length	Values	Comment	Default Value
5.1	1	0	PacketCable 1.0	NONE
		1	PacketCable 1.1	
		2	PacketCable 1.2	
		3	PacketCable 1.3 (S-MTA)	
		4-255	Reserved	

10.2 Number Of Telephony Endpoints

This TLV MUST be supplied in the Capabilities String.

Type	Length	Values	Comment	Default
5.2	1	n	Number of endpoints	NONE

10.3 TGT Support

Type	Length	Value	Comment	Default Value
5.3	1	0	0: No	0
		1	1: Yes	

10.4 HTTP Download File Access Method Support

Type	Length	Value	Comment	Default Value
5.4	1	0	0: No	0
		1	1: Yes	

10.5 MTA-24 Event SYSLOG Notification Support

Type	Length	Value	Comment	Default Value
5.5	1	0	0: No	0
		1	1: Yes	

10.6 NCS Service Flow Support

Type	Length	Value	Comment	Default Value
5.6	1	0	0: No	0
		1	1: Yes	

10.7 Primary Line Support

Type	Length	Value	Comment	Default Value
5.7	1	0	0: No	0
		1	1: Yes	

10.8 Vendor Specific TLV Type(s)

This TLV can be supplied in the Capabilities String if an MTA requires a specific processing of the Vendor Specific TLV Type(s).

Type	Length	Value	Comment	Default Value
5.8	n	{seq-of-bytes}:	One type per byte	43
			per byte	

10.9 NVRAM Ticket/Ticket Information Storage Support

Type	Length	Value	Comment	Default Value
5.9	1	0	0: No	1
		1	1: Yes	

10.10 Provisioning Event Reporting Support (para 5.4.3)

Type	Length	Value	Comment	Default Value
5.10	1	0	0: No	0
		1	1: Yes	

10.11 Supported CODEC(s)

This TLV MUST be supplied in the Capabilities String.

Type	Length	Value	Comment	Default Value
5.11	n	{seq-of-bytes}	one ID per byte	NONE

CODEC ID is the value represented by the Enumerated Type of "PktcCodecType" TEXTUAL CONVENTION in MTA MIB:

1: other,

2: unknown,

3: G.729,

4: reserved,

5: G.729E,

6: PCMU,

7: G.726-32

8: G.728,

9: PCMA,

10: G.726-16

11: G.726 24

12: G.726 40

10.12 Silence Suppression Support

Type	Length	Value	Comment	Default Value
5.12	1	0	0: No	0
		1	1: Yes	

10.13 Echo Cancellation Support

Type	Length	Value	Comment	Default Value
5.13	1	0	0: No	0
		1	1: Yes	

10.14 RSVP Support

Type	Length	Value	Comment	Default Value
5.14	1	0	0: No	0
		1	1: Yes	

10.15 UGS-AD Support

Type	Length	Value	Comment	Default Value
5.15	1	0	0: No	0
		1	1: Yes	

10.16 MTA's "ifIndex" starting number in "ifTable"

This TLV contains the value of the "ifIndex" for the first MTA Telephony Interface in "ifTable" MIB Table. The TLV MUST be supplied in the Capabilities String.

Type	Length	Value	Comment	Default Value
5.16	1	n	first MTA Interface	9

10.17 Provisioning Flow Logging Support

This capability is set to a corresponding value depending on the support of the logging capability of the Provisioning Flow (as per section 5.4.3).

Type	Length	Value	Comment	Default Value
5.17	1	0	0: No	0
		1	1: Yes	

10.18 Supported Provisioning Flows

An MTA MUST include this TLV in the Capabilities String. This TLV indicates the provisioning flows the MTA supports (Basic, Hybrid and Secure). It contains a bitmask indicating all the provisioning flows supported by the MTA.

Type	Length	Value	Comment	Default Value
5.18	2	{bit-mask}	See below	NONE

The Value field consists of the octet(s) in network byte order. Each bit represents a specific provisioning flow. If a bit set to 1, the MTA supports the corresponding flow. If a bit is set to 0 (zero), the MTA does not support the flow.

Bit assignments:

Bit 0 – Secure Flow (Full Secure Provisioning Flow)

Bit 1 – Hybrid Flow

Bit 2 – Basic Flow

The MTA MUST set all unused bits in the bitmask to 0. The MTA MUST set bit 0 in the TLV to 1 to indicate that it supports the Secure Flow. The MTA MUST set bits 1 and 2 in the TLV to indicate whether it supports the Basic and Hybrid Flows. To provide backward compatibility with the MTAs certified prior to the introduction of the Basic & Hybrid Flows, the absence of this TLV indicates that the MTA only supports the Secure Flow.

11 TLV-38 SNMP NOTIFICATION RECEIVER SPECIFICATION

This PacketCable TLV 38 specifies one or more Network Management Stations that must receive notifications from the MTA (MTA25 and post-provisioning, if required). If TLV38 contains wrong Subtypes or Length, the MTA MUST reject the configuration file and report a "failConfigFile" error. If TLV38 contains sub-types with wrong Values, then the MTA MUST follow the requirements specified below in each sub-TLV.

Type	Length	Value
38	N	Composite (contains sub-TLVs)

Unless specified or configured otherwise, the MTA MUST send the notifications to the default provisioning system (defined in DHCP option 122 sub-option 3).

11.1 Sub-TLVs of TLV-38

11.1.1 SNMP Notification Receiver IP Address

This sub-TLV specifies the IP address of the notification receiver.

Type	Length	Value
38.1	4	4 bytes of an IPv4 address in network byte order

If TLV 38 is present in the configuration file and the sub-TLV 38.1 is absent, the MTA MUST ignore TLV-38 and proceed with further processing of the configuration file and MUST report a provisioning state of passWithWarnings and populate the error OID table (pktcMtaDevErrorOidsTable).

11.1.2 SNMP Notification Receiver UDP Port Number

This sub-TLV specifies the Port number on the notification receiver to receive the notifications.

Type	Length	Value
38.2	2	UDP Port Number

If TLV 38 is present and the sub-TLV 38.2 is absent, then a default value of 162 MUST be used.

11.1.3 SNMP Notification Receiver Type

This sub-TLV specifies the SNMP Notification Receiver Type; it is the type of SNMP notifications the MTA MUST send to the associated SNMP Notification Receiver.

Type	Length	Value
38.3	2	1: SNMPv1 trap in an SNMPv1 packet 2: SNMP v2c trap in an SNMP v2c packet 3: SNMP INFORM in an SNMP v2c packet 4: SNMP trap in an SNMPv3 packet 5: SNMP INFORM in a SNMPv3 packet

If TLV 38 is present in the configuration file but sub-TLV 38.3 is absent, the MTA MUST ignore the entire TLV-38 and proceed with further processing of the configuration file and MUST report passWithWarnings and populate the Error OID table (pktcMtaDevErrorOidsTable). The MTA and Provisioning Server MUST support notification type values 2 and 3, and MAY support notification type values 1,4 or 5 from the above table. If an unsupported or invalid notification type value is received, the MTA MUST ignore the entire TLV38 that contains this entry and MUST report passWithWarnings and populate the error OID table (pktcMtaDevErrorOidsTable).

11.1.4 SNMP Notification Receiver Timeout

This sub-TLV specifies the wait time before a retry is attempted when the sender of an SNMP INFORM fails to receive an acknowledgement. Note that the number of retries is defined in sub-TLV 38.5.

Type	Length	Value
38.4	2	Time in milliseconds

If TLV 38 is present in the configuration file and the sub-TLV 38.4 is absent, the MTA MUST assume a default value of 15000 milliseconds. This corresponds to the default value of 1500 hundredths of a second defined for the snmpTargetAddrTimeout MIB object (see Appendix P of [26] and RFC 3413 [7]).

11.1.5 SNMP Notification Receiver Retries

This sub-TLV specifies the maximum number of times the MTA MUST retry sending an SNMP INFORM message if an acknowledgement is not received. Note that the wait time before each retry is defined by sub-TLV 38.4.

Type	Length	Value
38.5	2	Number of retries

If not present, the MTA MUST use a default value of 3. The maximum number of retries that can be specified is 255.

11.1.6 SNMP Notification Receiver Filtering Parameters

This sub-TLV specifies the filtering scheme for notifications and contains the root OID of the MIB sub-tree that defines the notifications to be sent to the Notification Receiver. The MTA MUST filter notifications being sent to the SNMP manager specified in sub-TLV 38.1 using the information provided. If this sub-TLV is not present, the MTA MUST use the default OID value for the 'iso' root.

Type	Length	Value
38.6	n	Filter OID (ASN.1 formatted Object Identifier)

The encoding of this TLV value field starts with the ASN.1 Universal type 6 (Object Identifier) followed by the ASN.1 length field and is terminated by the ASN.1 encoded object identifier component.

11.1.7 SNMPv3 Notification Receiver Security Name

This sub-TLV specifies the SNMPv3 Security Name to use when sending an SNMPv3 Notification. This sub-TLV is only being used if MTA supports TLV 38.3 (Notification Receiver Type) types 4 and 5. The MTA MUST ignore this sub-TLV 38.7 if a Notification Receiver Type (sub-TLV 38.3) other than 4 or 5 is received in the configuration file.

The following requirements apply to MTAs that supports Notification Receiver Type values of 4 or 5 in sub-TLV-38.3:

- If this sub-TLV38.7 is omitted, then the SNMPv3 Notifications MUST be sent in the noAuthNoPriv security level using the security name "@mtaconfig".
- If this sub-TLV is included, the MTA verifies that the value of the Security Name exists for the MTA local authoritative SNMP engine and creates an entry to further associate with the notification receiver authoritative engine (using the security levels and keys from the existing Security Name). If the Security Name of this sub-TLV does not exist for the local engine, the entire TLV 38 MUST be ignored and the MTA MUST reports passWithWarnings and populate the Error OID table (pktcMtaDevErrorOidsTable) for the entire TLV 38 and associated sub-TLVs that are ignored.

TYPE	LENGTH	VALUE
38.7	2-26	Security Name

11.2 Mapping of TLV fields into SNMP Tables

The following sections detail the MTA configuration file TLV-38 “PacketCable SNMP Notification Receiver” mapping onto SNMP functional tables.

Upon receiving each TLV 38 value, the MTA MUST make entries to the following tables in order to cause the desired SNMP TRAP or INFORM transmission: snmpNotifyTable, snmpTargetAddrTable, snmpTargetAddrExtTable, snmpTargetParamsTable, snmpNotifyFilterProfileTable, snmpNotifyFilterTable, snmpCommunityTable, usmUserTable, vacmSecurityToGroupTable, vacmAccessTable, and vacmViewTreeFamilyTable. An MTA MUST support a minimum of ten TLV-38 elements in a configuration file.

11.2.1 Mapping of TLV fields into created SNMP Table rows

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

The correspondence between the tags and the sub-TLVs themselves is as shown below:

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

The creation of rows with column values or indices containing the suffix "n" in the tables below indicates that these entries are created with the (n-1)th TLV 38 found in the MTA configuration file

11.2.1.1 snmpNotifyTable

If TLV38 elements are present and irrespective of the number of elements the MTA MUST create two rows with fixed values as described in the table below.

Table 3. snmpNotifyTable

snmpNotifyTable (RFC 3413, SNMP-NOTIFICATION-MIB)	First Row	Second Row
Column Name (* = Part of Index)	Column Value	Column Value
* snmpNotifyName	"@mtaconfig_inform"	"@mtaconfig_trap"
snmpNotifyTag	"@mtaconfig_inform"	"@mtaconfig_trap"
snmpNotifyType	inform (2)	trap (1)
snmpNotifyStorageType	Volatile	Volatile
snmpNotifyRowStatus	active (1)	active (1)

11.2.1.2 snmpTargetAddrTable

For each TLV38 element in the configuration file, the MTA MUST create one row according to the table below.

Table 4. snmpTargetAddrTable

snmpTargetAddrTable (RFC 3413, SNMP-TARGET-MIB)	New Row
Column Name (* = Part of Index)	Column Value
* snmpTargetAddrName	"@mtaconfig_n" Where n ranges from 0 to m-1 where m is the number of notification receiver TLV elements in the Configuration 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> == 2 "@mtaconfig_trap" Else If <Trap type> = 3 "@mtaconfig_inform"
snmpTargetAddrParams	"@mtaconfig_n" (Same as snmpTargetAddrName value)
snmpTargetAddrStorageType	Volatile
snmpTargetAddrRowStatus	active (1)

11.2.1.3 snmpTargetAddrExtTable

For each TLV38 element in the configuration file, the MTA MUST create one row according to the table below.

Table 5. snmpTargetAddrExtTable

snmpTargetAddrExtTable (RFC 3584, SNMP-COMMUNITY-MIB)	New Row
Column Name (* = Part of Index)	Column Value
* snmpTargetAddrName	"@mtaconfig_n" Where n ranges from 0 to m-1 where m is the number of notification receiver TLV elements in the Configuration file
snmpTargetAddrTMask	<Zero length octet string>
snmpTargetAddrMMS	0

11.2.1.4 snmpTargetParamsTable

For each TLV 38 element in the configuration file MTA MUST create one row according to the table below.

Table 6. snmpTargetParamsTable

snmpTargetParamsTable (RFC 3413, SNMP-TARGET-MIB)	New Row
Column Name (* = Part of Index)	Column Value
* snmpTargetParamsName	"@mtaconfig_n" Where n ranges from 0 to m-1 where m is the number of notification receiver TLV elements in the Configuration file
snmpTargetParamsMPModel SYNTAX: snmpMessageProcessingModel	SNMPv2c (1)
snmpTargetParamsSecurityModel SYNTAX: snmpSecurityModel	SNMPv2c (2) NOTE: The mapping of SNMP protocol types to value here are different from snmpTargetParamsMPModel
snmpTargetParamsSecurityName	"@mtaconfig"
snmpTargetParamsSecurityLevel	NoAuthNoPriv
snmpTargetParamsStorageType	Volatile
snmpTargetParamsRowStatus	active (1)

11.2.1.5 snmpNotifyFilterProfileTable

For each TLV 38 element in the configuration file with non zero value of TLV38 sub type 6 MTA MUST create one row according to the table below.

Table 7. snmpNotifyFilterProfileTable

snmpNotifyFilterProfileTable (RFC 3413, SNMP-NOTIFICATION-MIB)	New Row
Column Name (* = Part of Index)	Column Value
* snmpTargetParamsName	"@mtaconfig_n" Where n ranges from 0 to m-1 where m is the number of notification receiver TLV elements in the Configuration file
snmpNotifyFilterProfileName	"@mtaconfig_n" Where n ranges from 0 to m-1 where m is the number of notification receiver TLV elements in the Configuration file
snmpNotifyFilterProfileStorType	volatile
snmpNotifyFilterProfileRowStatus	active (1)

11.2.1.6 snmpNotifyFilterTable

For each TLV 38 element in the configuration file with non zero value of TLV38 sub type 6 MTA MUST create one row according to the table below.

Table 8. snmpNotifyFilterTable

snmpNotifyFilterTable (RFC 3413, SNMP-NOTIFICATION-MIB)	New Row
Column Name (* = Part of Index)	Column Value
* snmpNotifyFilterProfileName	"@mtaconfig_n" Where n ranges from 0 to m-1 where m is the number of notification receiver TLV elements in the Configuration file
* snmpNotifyFilterSubtree	<Filter OID> from the TLV
snmpNotifyFilterMask	<Zero Length Octet String>
snmpNotifyFilterType	included (1)
snmpNotifyFilterStorageType	Volatile
snmpNotifyFilterRowStatus	active (1)

11.2.1.7 snmpCommunityTable

If TLV38 elements are present and irrespective of the number of elements the MTA MUST create one row with fixed values as described in the table below.

Table 9. snmpCommunityTable

snmpCommunityTable (RFC 3584, SNMP-COMMUNITY-MIB)	First Row
Column Name (* = Part of Index)	Column Value
* snmpCommunityIndex	"@mtaconfig"
snmpCommunityName	"public"
snmpCommunitySecurityName	"@mtaconfig"
snmpCommunityContextEngineID	<The engineID of the MTA>
snmpCommunityContextName	<Zero length octet string>
snmpCommunityTransportTag	<Zero length octet string>
snmpCommunityStorageType	Volatile
snmpCommunityStatus	active (1)

11.2.1.8 usmUserTable

The usmUserTable is defined in RFC 3414 [8]. The entries in the table specify the user name on the remote notification receiver to which notification is to be sent. Rows in usmUserTable are created in two different ways when <Notification Receiver Type> (TLV-38.3) values 4 and 5 are supported by the MTA and is included in TLV38.

- If <Security Name> (TLV-38.7) is not included, irrespective of the number of TLV 38 elements in the configuration file, the MTA MUST create one entry row with fixed values as described by the first column ("Static row") in the table below.

- If <Security Name> (TLV38.7) is included then MTA MUST create additional entry rows as described by the second column ("Other Rows") in the table below. In this case, the creation of additional rows in usmUserTable occurs each time the engine ID of a notification receiver needs to be discovered (see RFC 3414 [8] for more details).

Table 10. usmUserTable

usmUserTable (RFC 3414, SNMP-USER-BASED-SM-MIB)	**Static Row** **Case 1**	**Other Rows** **Case 2**
Column Name (* = Part of Index)	Column Value	Column Value
* usmUserEngineID	0x00, create a new row on each time the EngineID of the Authoritative Notification Receiver is discovered.	0x00, create a new row on each time the EngineID of the Authoritative Notification Receiver is discovered.
usmUserName	"@mtaconfig".	When other rows are created, this is replaced with the <Security Name> field from the TLV element.
usmUserSecurityName	"@mtaconfig"	When other rows are created, this is replaced with the <Security Name> field from the TLV element
usmUserCloneFrom	<ignore> (zerodotZero) This row is not created by cloning mechanism	<ignore> (zerodotZero) This row is not created by cloning mechanism
usmUserAuthProtocol	None (usmNoAuthProtocol)	When other rows are created, this is replaced with none (usmNoAuthProtocol), or MD5 (usmHMACMD5AuthProtocol), or SHA (usmHMACSHAAuthProtocol) depending of the security level of the SNMPv3 user.
usmUserAuthKeyChange	Empty	Empty
usmUserOwnAuthKeyChange	Empty	Empty
usmUserPrivProtocol	Case 1: none (usmNoPrivProtocol)	When other rows are created this is replaced with none (usmNoPrivProtocol) or DES (usmDESPrivProtocol), depending of the security level of the SNMPv3 user.
usmUserPrivKeyChange	Empty	Empty
usmUserOwnPrivKeyChange	Empty	Empty
usmUserPublic	Empty	Empty
usmUserStorageType	volatile(2)	volatile(2)
usmUserStatus	active (1)	active (1)

11.2.1.9 vacmSecurityToGroupTable

If TLV38 elements are present and irrespective of the number of elements the MTA MUST create "Second Row" column and MAY create "First Row" or "Third Row" columns with fixed values as described in the Table below. MTA MUST populate "Second Row" and "Third Row" columns for Secure Provisioning Flow only.

Table 11. vacmSecurityToGroupTable

vacmSecurityToGroupTable (RFC 3415, SNMP-VIEW-BASED-ACM-MIB)	First Row	Second Row	Third Row
Column Name (* = Part of Index)	Column Value	Column Value	Column Value
* vacmSecurityModel	SNMPV1 (1)	SNMPV2c (2)	SNMPUSM (3)
* vacmSecurityName	"@mtaconfig"	"@mtaconfig"	"@mtaconfig"
vacmGroupName	"@mtaconfigV1"	"@mtaconfigV2"	"@mtaconfigUSM"
vacmSecurityToGroupStorageType	volatile(2)	volatile(2)	volatile(2)
vacmSecurityToGroupStatus	active (1)	active (1)	active (1)

11.2.1.10 VacmAccessTable

If TLV38 elements are present and irrespective of the number of elements the MTA MUST create "Second Row" column and MAY create "First Row" or "Third Row" columns with fixed values as described in the Table below. MTA MUST populate "Second Row" and "Third Row" columns for Secure Provisioning Flow only.

Table 12. vacmAccessTable

vacmAccessTable (RFC 3415, SNMP-VIEW-BASED-ACM-MIB)	First Row	Second Row	Third Row
Column Name (* = Part of Index)	Column Value	Column Value	Column Value
* vacmGroupName	"@mtaconfigV1"	"@mtaconfigV2"	"@mtaconfigUSM"
* vacmAccessContextPrefix	Empty	Empty	Empty
* vacmAccessSecurityModel	SNMPv1 (1)	SNMPv2c (2)	USM (3)
* vacmAccessSecurityLevel	noAuthNoPriv (1)	noAuthNoPriv (1)	noAuthNoPriv (1)
vacmAccessContextMatch	exact (1)	exact (1)	exact (1)
vacmAccessReadViewName	Empty	Empty	Empty
vacmAccessWriteViewName	Empty	Empty	Empty
vacmAccessNotifyViewName	"@mtaconfig"	"@mtaconfig"	"@mtaconfig"
vacmAccessStorageType	volatile(2)	volatile(2)	volatile(2)
vacmAccessStatus	active (1)	active (1)	active (1)

11.2.1.11 vacmViewTreeFamilyTable

If TLV38 elements are present and irrespective of the number of elements the entry below MUST be created. Note that this entry is already created at MTA initialization

Table 13. vacmViewTreeFamilyTable

vacmViewTreeFamilyTable (RFC 3415, SNMP-VIEW-BASED-ACM-MIB)	First Row
Column Name (* = Part of Index)	Column Value
* vacmViewTreeFamilyViewName	"@mtaconfig"
* vacmViewTreeFamilySubtree	1.3
vacmViewTreeFamilyMask	<Default from MIB>
vacmViewTreeFamilyType	included (1)
vacmViewTreeFamilyStorageType	volatile
vacmViewTreeFamilyStatus	active (1)

11.3 TLV38 and TLV11 Configuration Example

This section presents configuration examples for the generation of TLV-38 and TLV-11 for the purpose of SNMP framework configuration based on the framework model and message processing defined in RFC 3410 [23], RFC3411 [24], and RFC 3412 [25].

11.3.1 TLV-38 Example

This section is informational. The example below presents the usability of TLV-38. One of the objectives of this section is to illustrate the usage of @mtaConfig_n. The following assumptions are made

- MTA ignores entries with <trap type> 1 and supports <trap type> 2,3, 4 and 5
- MTA already via a configuration process has an entry with usmUserName and usmUserSecurityName which is 'mtaUser' and another entry set for 'superUser'. For simplification, no VACM entries associated to this profile are included

The table below contains the Configuration File elements. Empty cells means use default values when applicable.

Table 14. Example Configuration File elements

Sub-TLV					
TLV38 order in the Configuration file	**TLV 38 Number 1**	**TLV 38 Number 2**	**TLV38 Number 3**	**TLV38 Number 4**	**TLV 38 Number 5**
SNMP Notification Receiver IP Address	10.0.5.9	10.0.5.9	10.0.4.9	10.0.4.9	10.0.8.9
SNMPv2 Notification Receiver UDP Port Number		162		57000	
SNMPv2 Notification Receiver Trap Type	2	3	1	4	5
SNMPv2 Notification Receiver Timeout	1500		2000		
SNMPv2 Notification Receiver Retries	3	1	2		
Notification Receiver Filtering Parameters	org	pktcMtaProvisioningStatus	Mib-2	PktcMtaMib	pktcMtaProvisioningStatus
Notification Receiver Security Name		notused		SuperUser	mtaUser
@mta@config_n	0	1	2	3	4

11.3.2 Content of the SNMP framework tables after processing of the above example TLV38s

Based on the above assumptions and the contents of TLV38 specified in previous sections, this section illustrates the tables the MTA should create. The MTA ignores TLV38 number 1 (notification type = 1), therefore @mtaconfig_2 entries do not exist); the Security Name in TLV n=2 is ignored.

Table 15. snmpCommunityTable

Index	[@mtaconfig]
Name	"public"
SecurityName	@mtaconfig
ContextEngineID	<MTA ENGINEID>
ContextName	""
TransportTag	""
StorageType	volatile
Status	active

Table 16. snmpTargetAddrExtTable

index	@mtaconfig_0	@mtaconfig_1	[@mtaconfig_2]	[@mtaconfig_3]	[@mtaconfig_4]	[@mtaconfig_5]
TMask	""	""	""	""	""	""
MMS	0	0	0	0	0	0

Table 17. usmUserTable

Index	[0x00][@mtaconfig]	[<local-EngineID>][mtaUser]	[<local-EngineID>][superUser]	[0x00/<Notif-recv-EngineID>][mtaUser]	[0x00/<Notif-recv-EngineID>][superUser]
SecurityName	@mtaconfig	MtaUser	superUser	mtaUser	superUser
CloneFrom	ZeroDotZero	ZeroDotZero	zeroDotZero	zeroDotZero	zeroDotZero
AuthProtocol	usmNoAuthProtocol	usmNoAuthProtocol	usmHMACMD5AuthProtocol	usmNoAuthProtocol	usmHMACMD5AuthProtocol
AuthKeyChange	""	""	""	""	""
OwnAuthKeyChange	""	""	""	""	""
PrivProtocol	usmNoPrivProtocol	usmNoPrivProtocol	usmDESPrivProtocoll	usmNoPrivProtocol	usmDESPrivProtocol
PrivKeyChange	""	""	""	""	""
OwnPrivKeyChange	""	""	""	""	""
Public	""	""	""	""	""
StorageType	Volatile	Volatile	Volatile	Volatile	Volatile
Status	active	active	active	active	active

Table 18. vacmContextTable

index
VacmContextName

Table 19. vacmSecurityToGroupTable

index	[1][@mtaconfig]	[2][@mtaconfig]	[3][@mtaconfig]
GroupName	@mtaconfigV1	@mtaconfigV2	@mtaconfigUSM
SecurityToGroupStorageType	Volatile	Volatile	Volatile
SecurityToGroupStatus	active	active	active

Table 20. vacmAccessTable

index	[@mtaconfigV1][][1][noAuthNoPriv]	[@mtaconfigV2][][2][noAuthNoPriv]	[@mtaconfigUSM][][3][noAuthNoPriv]
ContextMatch	exact	exact	exact
ReadViewName	""	""	""
WriteViewName	""	""	""
NotifyViewName	@mtaconfig	@mtaconfig	@mtaconfig
StorageType	Volatile	Volatile	Volatile
Status	active	active	active

Table 21. vacmViewTreeFamilyTable

index	[@mtaconfig][org]
Mask	""
Type	included
StorageType	Volatile
Status	active

Table 22. snmpNotifyTable

index	[@mtaconfig_inform]	[@mtaconfig_trap]
Tag	@mtaconfig_inform	@mtaconfig_trap
Type	inform	Trap
StorageType	Volatile	Volatile
RowStatus	active	active

Table 23. snmpTargetAddrTable

index	[@mtaconfig_0]	[@mtaconfig_1]	[@mtaconfig_3]	[@mtaconfig_4]
TDomain	snmpUDPDomain	snmpUDPDomain	snmpUDPDomain	snmpUDPDomain
TAddress	"0A 00 05 09 00 82"	"0A 00 05 09 00 82"	"0A 00 04 09 DE A8"	"0A 00 08 09 00 82"
Timeout	1500	1500	1500	1500
RetryCount	3	1	3	3
TagList	@mtaconfig_trap	@mtaconfig_inform	@mtaconfig_trap	@mtaconfig_inform
Params	@mtaconfig_0	@mtaconfig_1	@mtaconfig_3	@mtaconfig_4
StorageType	Volatile	Volatile	Volatile	Volatile
RowStatus	active	active	active	active

Table 24. snmpTargetParamsTable

Index	[@mtaconfig_0]	[@mtaconfig_1]	[@mtaconfig_3]	[@mtaconfig_4]
MPModel	1	1	3	3
SecurityModel	2	2	3	3
SecurityName	'@mtaconfig	'@mtaconfig	'@mtaconfig	'@mtaconfig
SecurityLevel	noAuthNoPriv	noAuthNoPriv	noAuthNoPriv	NoAuthNoPriv
StorageType	Volatile	Volatile	Volatile	Volatile
RowStatus	active	active	active	active

Table 25. snmpNotifyFilterProfileTable

index	[@mtaconfig_0]	[@mtaconfig_1]	[@mtaconfig_3]	[@mtaconfig_4]
Name	[@mtaconfig_0]	[@mtaconfig_1]	[@mtaconfig_3]	[@mtaconfig_4]
StorType	Volatile	Volatile	Volatile	Volatile
RowStatus	active	active	active	active

Table 26. snmpNotifyFilterTable

index	[@mtaconfig_0] [org]	[@mtaconfig_1] [pktcMtaProvision-ingStatus]	[@mtaconfig_3] [PktcMtaMib]	[@mtaconfig_4] [pktcMtaProvision-ingStatus]
Mask	""	""	""	""
Type	included	included	included	included
StorageType	Volatile	Volatile	Volatile	Volatile
RowStatus	active	active	active	active

12 SNMPV2C MANAGEMENT REQUIREMENTS

The management of an MTA device using SNMPv2c can be configured if required by an operator by setting the proper co-existence tables (using TLV11) in the MTA configuration file or via post-provisioning management.

- In the Basic and Hybrid Flows, the MTA MUST configure the tables described in section 12.1 and 12.2 after MTA4 to provide SNMPv2c read/write access to the default management system (provisioning entity provided in DHCP option 122 sub-option 3).
- In the Secure Flow, the MTA MUST configure the tables in section 12.2 only if in the configuration file contains TLV11 related to snmpCommunityTable and snmpTargetAddrTable or snmpTargetAddrExtTable.

Appendix II provides an example template for operators to enable SNMPv2c management.

12.1 SNMPV2c Co-existence mode tables content created by MTA after MTA-4 for Hybrid and Basic Flows.

Table 27. snmpCommunityTable Content

snmpCommunityTable (RFC 3584, SNMP-COMMUNITY-MIB)	**Read write Access**
Column Name (* = Part of Index)	Column Value
* snmpCommunityIndex	"@mtaprov"
snmpCommunityName	"public"
snmpCommunitySecurityName	"@mtaprov"
snmpCommunityContextEngineID	Empty
snmpCommunityContextName	Empty
snmpCommunityTransportTag	"@mtaprovTag"
snmpCommunityStorageType	Volatile(2)
snmpCommunityStatus	active(1)

Table 28. snmpTargetAddrTable Content

snmpTargetAddrTable (RFC 3413, SNMP-TARGET-MIB)	First Row
Column Name (* = Part of Index)	Column Value
* snmpTargetAddrName	"@mtaprov"
snmpTargetAddrTDomain	snmpUDPDomain = snmpDomains.1
snmpTargetAddrTAddress (IP Address non-Authoritative SNMP entity)	OCTET STRING (6) Octets 1-4: <IP address of SNMP Entity derived from 122.3> Octets 5-6: <0x0000>
snmpTargetAddrTimeout	Ignore, <use default>
snmpTargetAddrRetryCount	ignore, <use default>
snmpTargetAddrTagList	"@mtaprovTag"
snmpTargetAddrParams	Empty
snmpTargetAddrStorageType	volatile (2)
snmpTargetAddrRowStatus	active(1)

Table 29. snmpTargetAddrExtTable Content

snmpTargetAddrExtTable (RFC 3584, SNMP-COMMUNITY-MIB)	First Row
Column Name (* = Part of Index)	Column Value
* snmpTargetAddrName	"@mtaprov"
snmpTargetAddrTMask	Empty
snmpTargetAddrMMS	0

12.2 SNMP Default entries for SNMPv2 Access

The following tables MUST be created by the MTA during the SNMP agent initialization to configure SNMPv2 access.

Table 30. vacmSecurityToGroupTable Default Entries

vacmSecurityToGroupTable (RFC 3415, SNMP-VIEW-BASED-ACM-MIB)	First Row	Second Row	Third Row
Column Name (* = Part of Index)	Column Value	Column Value	Column Value
* vacmSecurityModel	SNMPv2c (2)	SNMPv2c (2)	SNMPv2c (2)
* vacmSecurityName	"@mtaprov"	"admin"	"operator"
vacmGroupName	"@mtaprov"	"admin"	"operator"
vacmSecurityToGroupStorageType	permanent(4)	permanent(4)	permanent(4)
vacmSecurityToGroupStatus	active(1)	active(1)	active(1)

Table 31. vacmAccessTable Default Entries

vacmAccessTable (RFC 3415, SNMP-VIEW-BASED-ACM-MIB)	First Row	Second Row	Third Row
Column Name (* = Part of Index)	Column Value	Column Value	Column Value
* vacmGroupName	"@mtaprov"	"admin"	"operator"
* vacmAccessContextPrefix	Empty	Empty	Empty
* vacmAccessSecurityModel	SNMPv2 (2)	SNMPv2 (2)	SNMPv2 (2)
* vacmAccessSecurityLevel	noAuthNoPriv (1)	noAuthNoPriv (1)	noAuthNoPriv (1)
vacmAccessContextMatch	exact (1)	exact (1)	exact (1)
VacmAccessReadViewName	"@mtaconfig"	"@mtaconfig"	"@mtaconfig"
VacmAccessWriteViewName	"@mtaconfig"	"@mtaconfig"	Empty
vacmAccessNotifyViewName	"@mtaconfig"	Empty	Empty
vacmAccessStorageType	permanent(4)	permanent(4)	permanent(4)
vacmAccessStatus	active(1)	active(1)	active(1)

Table 32. vacmViewTreeFamilyTable Default Entry

vacmViewTreeFamilyTable (RFC 3415, SNMP-VIEW-BASED-ACM-MIB)	First Row
Column Name (* = Part of Index)	Column Value
* vacmViewTreeFamilyViewName	@mtaconfig
VacmViewTreeFamilySubtree	1.3
vacmViewTreeFamilyMask	Empty <default from MIB>
vacmViewTreeFamilyType	included (1)
vacmViewTreeFamilyStorageType	permanent(4)
VacmViewTreeFamilyStatus	active (1)

Note that this entry is also created by default for the purpose of TLV-38 processing, It means only one default entry is needed in the MTA to define SNMPv2 management and TLV-38 configuration

Table 33. snmpTargetParamsTable Default Entry

snmpTargetParamsTable (RFC 3413, SNMP-TARGET-MIB)	First Row
Column Name (* = Part of Index)	Column Value
* snmpTargetParamsName	"@mtaprov"
snmpTargetParamsMPModel	1
snmpTargetParamsSecurityModel	2
snmpTargetParamsSecurityName	"@mtaprov"
snmpTargetParamsSecurityLevel	noAuthNoPriv
snmpTargetParamsStorageType	permanent(4)
snmpTargetParamsRowStatus	active (1)

Table 34. snmpNotifyTable Default Entry

snmpNotifyTable (RFC 3413, SNMP-NOTIFICATION-MIB)	First Row
Column Name (* = Part of Index)	Column Value
* snmpNotifyName	"@mtaprov"
snmpNotifyTag	"@mtaprovTag"
snmpNotifyType	inform (2)
snmpNotifyStorageType	permanent(4)
snmpNotifyRowStatus	active (1)

Table 35. snmpNotifyFilterProfileTable Default Entry

snmpNotifyFilterProfileTable (RFC 3413, SNMP-NOTIFICATION-MIB)	First Row
Column Name (* = Part of Index)	Column Value
* snmpTargetParamsName	"@mtaprov"
snmpNotifyFilterProfileName	"@mtaprov"
snmpNotifyFilterProfileStorType	permanent(4)
snmpNotifyFilterProfileRowStatus	active(1)

Table 36. snmpNotifyFilterTable Default Entry

snmpNotifyFilterTable (RFC 3413, SNMP-NOTIFICATION-MIB)	First Row	Second Row
Column Name (* = Part of Index)	Column Value	Column Value
* snmpNotifyFilterProfileName	"@mtaprov"	"@mtaprov"
* snmpNotifyFilterSubtree	pktcMtaNotification	snmpTraps
snmpNotifyFilterMask	Empty	Empty
snmpNotifyFilterType	included(1)	included(1)
snmpNotifyFilterStorageType	permanent(4)	permanent(4)
snmpNotifyFilterRowStatus	active(1)	active(1)

Appendix I Provisioning Events

Event Name	Default Severity for Event	Default Display String	Packet-Cable EventID	Comments
PROV-EV-1	Critical	"Waiting For ProvRealmKdcName Response"	65,535	DNS Srv request has been transmitted and no reply has yet been received.
PROV-EV-2	Critical	"Waiting For ProvRealmKdcAddr Response"	65,534	DNS Request has been transmitted and no reply has yet been received.
PROV-EV-3	Critical	"Waiting For AS-Reply"	65,533	AS request has been sent, and no MSO KDC AS Kerberos ticket reply has yet been received.
PROV-EV-4	Major	"Waiting For TGS-Reply"	65,532	TGS request has been transmitted and no TGS ticket reply has yet been received.
PROV-EV-5	Critical	"Waiting For AP-Reply"	65,531	AP request has been transmitted and no SNMPv3 key info reply has yet been received.
PROV-EV-6	Critical	"Waiting For Snmp GetRequest"	65,530	INFORM message has been transmitted and the device is waiting on optional/iterative GET requests.
PROV-EV-7	Critical	"Waiting For Snmp SetInfo"	65,529	MTA is waiting On config file download access information.
PROV-EV-8	Major	"Waiting For TFTP AddrResponse"	65,528	DNS Request has been transmitted and no reply has yet been received.
PROV-EV-9	Critical	"Waiting For ConfigFile"	65,527	TFTP request has been Transmitted and no reply has yet been received or a download is in progress.
PROV-EV-10	Major	"Waiting For TelRealmKdc NameResponse"	65,526	DNS Srv request has been transmitted and no name reply has yet been received.
PROV-EV-11	Major	"Waiting For TelRealmKdc Addr Response"	65,525	DNS Request has been transmitted and no address reply has yet been received.
PROV-EV-12	Major	"Waiting For Pkinit AS-Reply"	65,524	AS request has been transmitted and no ticket reply has yet been received.
PROV-EV-13	Major	"Waiting For CmsKerbTick TGS-Reply"	65,523	TGS request has been transmitted and no ticket reply has yet been received.
PROV-EV-14	Major	"Waiting For CmsKerbTick AP-Reply"	65,522	AP request has been transmitted and no Ipsec parameters reply has yet been received.
PROV-EV-15	Critical	"Provisioning TimeOut"	65,521	The provisioning sequence took too long from MTA-15 to MTA-19 (specified in pktcMtaDevProvisioningTimer).

PROV-EV-16	Critical	"ConfigFile – BadAuthentication"	65,520	The config file authentication value did not agree with the value in pktcMtaDevProvConfigHash or the authentication parameters were invalid.
PROV-EV-17	Critical	"ConfigFile – BadPrivacy"	65,519	The privacy parameters were invalid.
PROV-EV-18	Critical	"ConfigFile – BadFormat"	65,518	The format of the configuration file was not as expected.
PROV-EV-19	Major	"ConfigFile – MissingParam"	65,517	Mandatory parameter of the configuration file is missing.
PROV-EV-20	Major	"ConfigFile – BadParam"	65,516	Parameter within the configuration file had a bad value.
PROV-EV-21	Major	"ConfigFile – BadLinkage"	65,515	Table linkages in the configuration file could not be resolved.

Appendix II SNMPv2c co-existence Configuration Example - Template for service providers

The operators can use the template defined in this section to enable SNMPv2c management (default entries defined in section 12.2 are reused in the example). Note that service providers are not restricted to use this template.

Table 37. snmpCommunityTable Template for Basic and Hybrid Flows Configuration file

snmpCommunityTable (RFC 3584, SNMP-COMMUNITY-MIB)	**Read write Access**	**Read only Access**
Column Name (* = Part of Index)	Column Value	Column Value
* snmpCommunityIndex	"admin"	"operator" or <any>
snmpCommunityName	<SNMP Community Name>	<SNMP Community Name>
snmpCommunitySecurityName	"admin"	"operator"
snmpCommunityContextEngineID	Empty	Empty
snmpCommunityContextName	Empty	Empty
snmpCommunityTransportTag	"adminTag"	"operatorTag"
snmpCommunityStorageType	volatile(2)	Volatile (2)
snmpCommunityStatus	createAndGo(4)	createAndGo(4)

Table 38. snmpTargetAddrTable Template for Basic and Hybrid Flows Configuration file

snmpTargetAddrTable (RFC-3413 - SNMP-TARGET-MIB	**First Row**	**Second Row**
Column Name (* = Part of Index)	Column Value	Column Value
* snmpTargetAddrName	"admin"	"operator"
snmpTargetAddrTDomain	snmpUDPDomain = snmpDomains.1	snmpUDPDomain = snmpDomains.1
snmpTargetAddrTAddress (IP Address non-Authoritative SNMP entity	OCTET STRING (6) Octets 1-4: <SNMP Mgmt Station IPv4 Address> Octets 5-6: <0x0000>	OCTET STRING (6) Octets 1-4: <SNMP Mgmt Station IPv4 Address> Octets 5-6: <0x0000>
snmpTargetAddrTimeout	ignore, <use default>	Ignore, <use default>
snmpTargetAddrRetryCount	Ignore, <use default>	Ignore, <use default>
snmpTargetAddrTagList	"adminTag"	"operatorTag"
snmpTargetAddrParams	Empty	Empty
snmpTargetAddrStorageType	volatile (2)	volatile (2)
snmpTargetAddrRowStatus	createAndGo(4)	createAndGo(4)

Table 39. snmpTargetAddrExtTable Template for Basic and Hybrid Flows Configuration file

snmpTargetAddrExtTable (RFC 3584, SNMP-COMMUNITY-MIB)	First Row	Second Row
Column Name (* = Part of Index)	Column Value	Column Value
* snmpTargetAddrName	"admin"	"operator"
snmpTargetAddrTMask	OCTET STRING (6) Octets 1-4: <SNMP Mgmt Station Sub Net Mask> Octets 5-6: <0x0000>	OCTET STRING (6) Octets 1-4: <SNMP Mgmt Station Sub Net Mask> Octets 5-6: <0x0000>
snmpTargetAddrMMS	0	0

Appendix III Acknowledgements

On behalf of CableLabs and its participating member companies, I would like to extend a heartfelt thanks to all those who contributed to the development of this specification. Certainly all the participants of the provisioning focus team have added value to this effort by participating in the review and weekly conference calls. Particular thanks are given to:

Sumanth Channabasappa (Alopa Networks)

Angela Lyda, Rick Morris, Rodney Osborne (Arris Interactive);

Steven Bellovin and Chris Melle (AT&T);

Eugene Nechamkin (Broadcom);

John Berg, Maria Stachelek (CableLabs);

Paul Duffy, Klaus Hermanns, Azita Kia, Michael Thomas, Rich Woundy (Cisco);

Deepak Patil (Com21);

Jeff Ollis, Rick Vetter (General Instrument/Motorola);

Roger Loots, David Walters (Lucent);

Burcak Besar (Pacific Broadband);

Peter Bates (Telcordia);

Patrick Meehan (Tellabs);

Roy Spitzer (Telogy);

Satish Kumar, Itay Sherman and Roy Spitzer (Texas Instrument),

Aviv Goren (Terayon);

Prithivraj Narayanan (Wipro)

Venkatesh Sunkad, CableLabs Team

Appendix IV Revision History

Engineering Change Numbers incorporated in PKT-SP-I02-010323.

ECN	ECN Date	Summary
prov-n-00008	4/20/00	MTA Device Signature
prov-n-99005-v2	4/28/00	MTA's two separate code images
prov-n-00023	6/2/00	Telephony Certificate
prov-n-00024-v2	6/2/00	Security Association
prov-n-00030-v2	6/9/00	DHCP options
sec-n-00022-v2	6/2/00	TGS Certificate
mib-n-00027	6/9/00	Configuration file entities
prov-n-00026	9/11/00	New TLV
prov-n-00043-v3	9/11/00	Provisioning flow sequencing
prov-n-00019-v3	9/25/00	DHCP option code 60
prov-n-00099	1/15/01	Examples for HTTP and TFTP transport protocols
prov-n-00101	1/15/01	Provisioning Correlation ID
prov-n-00100-v2	1/22/01	Clarification
prov-n-00122	1/22/01	TLV value is now specified
sec-n-00146-v214	1/22/01	Secure Provisioning ECN
sec-n-00079	3/12/01	Kerberos principal name without downloading new config file
prov-n-00098-v3	3/5/01	Behavior of MTA for changed DHCP values
prov-n-01006	2/26/01	X.509 Certificate
prov-n-01008-v3	3/5/01	Encryption keys for SNMPv3 Informs
prov-n-01012	3/12/01	DHCP information in the config file
prov-n-01013	3/12/01	Clarify previous ECNs impact
prov-n-01018	3/26/01	MIB changes impact on I02
prov-n-01017	3/12/01	MTA and SNMP set

Engineering Change incorporated in PKT-SP-I03-011221.

ECN ID	ECN Date	Summary
mib-n-01077	7/16/01	Clarify usage of pktcMtaDev Provisioning Timer, clean up some security related objects
prov-n-01033-v2	5/7/01	Error conditions that can occur between MTA15 and MTA25.
prov-n-01037	5/7/01	Several editorial changes in Security document.
prov-n-01038	5/7/01	It is not clear what event is reported if an endpoint is provisioned or an endpoint is no longer provisioned.
prov-n-01039	5/7/01	Config file MUST be rejected if the required info is not present.
prov-n-01059	6/4/01	Language clarification regarding the Provisioning Flow as a mandatory requirement.
prov-n-01060	6/4/01	Testing group cannot easily determine associated MIB object used in configuration file
prov-n-01061	6/4/01	The receive UDP port cannot be configured using the config file.
prov-n-01065	6/4/01	Revise wording in section 9.1 to specify the tables being referenced.
prov-n-01066	6/18/01	Correct typos in ECN 00184
prov-n-01076	7/16/01	Clarification regarding the presence of the "Telephony Service Provider SNMP Entity" attribute in the Device Level Configuration Data.
prov-n-01078	7/16/01	Clarify the intent of the e-MTA firmware download
prov-n-01079	7/16/01	The Provisioning Specification (I02) allows two ways of distributing of the MTA FQDN
prov-n-01106	8/20/01	Duplicate instances of requirement statements for Code 177 sub-option 1 thru 7 are deleted. Also corrects typo.
prov-n-01118	9/10/01	The description of the requirements for tables (and their corresponding MIB entries) is unclear.
prov-n-01119	9/10/01	Augment sec-n-01029 and clear up several.
prov-n-01123	9/10/01	MTA Provisioning Spec clarification on MTA FQDN supply to E-MTA during provisioning.
prov-n-01156	10/15/01	Add suboption 9 to DHCP option 177 to support provisioning CMS for E911/611 service.
prov-n-01157	10/15/01	Clarification in the usage of "pktcSigDefNcsReceiveUdpPort" MIB object.
prov-n-01176	11/19/01	Additions on the usage of the Log Event Mechanism in E-MTA
prov-n-01198	11/19/01	Add "MUST" statements to provisioning flows MTA20 – MTA22
prov-n-01182	12/10/01	Provisioning of the Signaling Communication Path between the MTA and CMS introduced, with new conf file TLV.
prov-n-01219	12/17/01	Correction of minor typographical errors and modifications to provisioning specification.

The following Engineering Change are incorporated in PKT-SP-PROV-I04-021018.

ECN ID	ECN Date	Summary
mibsig-n-02043	6/24/02	Changes representation of ring cadences to allow more granular ring cadences
prov-n-02014	6/24/02	While Provisioning Specification mandates Kereberized SNMPv3 key negotiation, it does not define the mechanism for initial delivery of the timeouts for the AS-REQ/REP backoff and retry mechanism.
prov-n-02020	6/24/02	Remove statements carried over from I01 version that are irrelevant now.
prov-n-02025	6/24/02	Allow and define how vendor-specific information should be entered in the MTA configuration file
prov-n-02026	6/24/02	Editorial Corrections and Clarifications
prov-n-02032	6/24/02	Insure that all DHCP ACK message content is treated as authoritative.
prov-n-02033	6/24/02	The behavior of the MTA during its provisioning MUST be dictated by the presence/ absence of option code 177 and if present, the value in its suboptions.
prov-n-02045	6/24/02	In Section 7.6, paragraph 1 it is not clear whether or not the NCS Service Flow MUST be implemented on the MTA.
prov-n-02050	6/24/02	Correction to the description of the Service Provider's SNMP Entity Address in section 8.1.2 to bring in line with the related pktcMtaDevSnmpEntity MIB definition in PKT-SP-MIB-MTA-I03-020116.
prov-n-02076	6/24/02	Specification requires clarification about MTA behavior under specific conditions for DHCP in regard to the use of option codes and PacketCable specific sub-options.
prov-n-02090	6/24/02	Per-Realm Configuration data should have “MUST” attribute to be able to provide the “OrgName” in the MTA Configuration File needed to verify the validity of the Organization Name attribute in the Service Provider Certificate.
prov-n-02100	6/24/02	The ECR defines the list and the representation of the MTA Capabilities in DHCP Option-60.
prov-n-02106	7/1/02	MTA18 provisioning step contains “SHOULD” terminology for normal flow sequencing which contradicts the “MUST” condition for configuration file hash.
prov-n-02147	7/29/02	The document will continue to use the DOCSIS TFTP backoff and retry.
prov-n-02145	7/29/02	There is a need for the Telephony Service Provider to shut of the MTA if and when required using DHCP.
prov-n-02146	7/29/02	Clarify the TLV to be used for lengths of values greater than 254 in the TLV configuration file.
prov-n-02155	8/22/02	Defines the approach, which would allow the Service Providers to control the enabling/disabling of the signaling security (IPSEC) and Key Management flows associated with it.

The following Engineering Change are incorporated in PKT-SP-PROV-I05-021127.

ECN ID	ECN Date	Summary
prov-n-02183	11/18/02	Correction to EventID not defined properly with unique numbers.
prov-n-02188	11/18/02	Updates to Sec 8.1, DHCP option code 177, sub-options 1 and 2; addresses the condition where an MTA stores valid Kerberos tickets and then bypasses the AS-REQ upon reboot because ticket hasn't expired.
prov-n-02204	11/20/02	Clarification for configuration file handling due to changes in the MTA-MIB and assoc reference changes.
prov-n-02205	11/18/02	Define config file size limit behavior during a PC MTA initialization.
prov-n-02206	11/18/02	Ensure MTA can work in a single and dual IP subnets environments.

The following Engineering Change are incorporated in PKT-SP-PROV-I06-030415.

ECN ID	ECN Date	Summary
prov-n-02219	1/13/02	Provisioning Specification failure condition is not defined properly
prov-n-03018	3/10/03	Eliminates the MIB data dependencies which E-MTA has with the eDOCSIS part of the modem.
prov-n-03025	3/10/03	The ECR proposes the new way to indicate the default values in sub-option-10 and -11 data.
prov-n-03033	3/10/03	Utilize the new IANA assigned DHCP option code in the PacketCable environment.
prov-n-03034	3/10/03	Clarification of CMS Kerberos Realm Name in Per-CMS Configuration Data Table
pkt-n-03006	2/12/02	Updates standard definition of SFID.

The following Engineering Change are incorporated in PKT-SP-PROV-I07-030728.

ECN ID	ECN Date	Summary
prov-n-03041	6/23/03	Clarifies the usage of pktcmtadevenabled MIB object in provisioning specification.
prov-n-03061	6/30/03	Require MTAs to store Kerberos Tickets in NVRAM.

The following Engineering Change are incorporated in PKT-SP-PROV-I08-040113.

ECN ID	ECN Date	Summary
prov-n-03073	9/22/03	The ECR introduces the MUST requirement to the Provisioning Flow in case of each individual flow step failure.
prov-n-03080	12/1/03	Clarifies the misleading MTA requirements in provisioning specification and also cleanup the spec.
prov-n-03092	11/10/03	Incorporate CableLabs Client Configuration sub-option 9 (RFC 3594) into the MTA Provisioning Specification; Clarify DHCP OFFER/ACK language in section 8.

The following Engineering Change are incorporated in PKT-SP-PROV-I09-040402.

ECN ID	ECN Date	Summary
prov-n-03108-4	3/11/04	Changes made due to the recent introduction of Simplified PacketCable Provisioning Flows for PacketCable upon the request of MSOs.
prov-n-03121-v5	3/11/04	As Provisioning System requires some additional information to support Simplified Provisioning Flows, there is a proposal to provide this additional info by means of DHCP Option-60 and Option-43.
prov-n-03123-v5	3/11/04	Describe non-secure provisioning flows.