PacketCable™ 1.5 Specifications

CMS Subscriber Provisioning

PKT-SP-CMSPROV1.5-I01-050128

ISSUED

Document Status Sheet

Document Control Number:	PKT-SP-CMSPROV1.5-I01-050128			
Document Title:	CMS Subscriber Provisioning			
Revision History:	D01 – Released September 30, 2004			
	D02 – Released December 10, 2004			
	I01 – Issued January 28, 2005			
Date:	January 28, 2005			
Status:	~~Work in Progress~~	~~Issued~~	Draft	~~Closed~~
Distribution Restrictions:	~~Author Only~~	~~CL/Member~~	~~CL/ Member/ 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 review 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 SCOPE

1.1 Purpose of document

PacketCable 1.5 service provisioning can be viewed as two distinct operations: MTA provisioning and CMS subscriber provisioning. MTA initialization and provisioning is outlined in the PacketCable MTA Device Provisioning Specification [2]. This document defines the interface used between the CMS and Provisioning Server for the exchange of service provisioning information. It is intended to facilitate interoperability of conforming hardware and software from multiple vendors.

The interface employs a Web Service model. Specified in Web Service Description Language (WSDL 1.1), the interface transports XML encoded objects within SOAP 1.1 encoded messages over an HTTP 1.1 transport. This interface is secured via IPSec.

The data model transported upon this interface is specifically designed to be extensible, allowing incorporation of as yet undefined PacketCable features and specific vendor extensions.

1.2 Document Scope

The scope of this document is limited to the provisioning of a PacketCable 1.5 CMS by a single service provider. Additionally:

- The CMS provisioning interface is limited to the exchange of service activation data between the CMS and the PS. The interface between the PS and the BackOffice Operations Support System (OSS) is out of scope.
- CMS element management and network element provisioning (dial plans, etc.) are out of scope.
- Customer record creation/billing is considered part of the back office OSS application and is currently out of scope.

1.3 Requirements

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

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

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

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] PacketCable 1.5 Architecture Framework Technical Report**,** PKT-TR-ARCH1.5-V01-050128, January 28, 2005, Cable Television Laboratories, Inc.

[2] PacketCable 1.5 MTA Device Provisioning Specification**,** PKT-SP-PROV1.5-I01-050128, January 28, 2005, Cable Television Laboratories, Inc.

[3] PacketCable 1.5 Security Specification**,** PKT-SP-SEC1.5-I01-050128, January 28, 2005, Cable Television Laboratories, Inc.

[4] IETF RFC 2822, Internet Message Format, April 2001.

[5] PacketCable 1.5 Audio/Video Codecs Specification, PKT-SP-CODEC1.5-I01-050128, January 28, 2005, Cable Television Laboratories, Inc.

[6] PacketCable 1.5 Network Based Call Signalling Protocol Specification, PKT-SP-NCS1.5-I01-050128, January 28, 2005, Cable Television Laboratories, Inc.

2.2 Informational References

[7] Web Services Description Language. http://www.w3.org/TR/wsdl

[8] Simple Object Access Protocol. http://www.w3.org/TR/SOAP

[9] XML Protocol. http://www.w3.org/2000/xp

[10] Data-Over-Cable Service Interface Specifications, Cable Modem to Customer Premise Equipment Interface Specification, CMCI, DOCSIS SP-CMCI-I09-030730, July 30, 2003, Cable Television Laboratories, Inc.

2.3 Reference Acquisition

- Cable Television Laboratories, Inc., 858 Coal Creek Circle, Louisville, CO 80027; Phone 303-661-9100; Fax 303-661-9199; Internet: www.packetcable.com./; www.cablemodem.com/
- International Organization for Standardization (ISO) 1, rue de Varembé, Case postale 56, CH-1211 Geneva 20, Switzerland, Phone 41-22-749-01-11; Fax 41-22-733-34-30, Internet: www.iso.org/
- Telcordia Technologies, 445 South Street, Morristown, NJ 07960-6438, Phone 973-829-2000, Internet: www.telcordia.com/
- Internet Engineering Task Force (IETF) Secretariat c/o Corporation for National Research Initiatives, 1895 Preston White Drive, Suite 100, Reston, VA 20191-5434, Phone 703-620-8990, Fax 703-620-9071, Internet: www.ietf.org/

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.
Adaptive Timeout	Retry with exponential timeout: 1st attempt – 1 sec and the last attempt – 16 secs.
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 head-end 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.
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 unidirectional sequence of packets associated with a Service ID and a QoS. Multiple multimedia streams may be carried in a single DOCSIS Flow. Also known as a DOCSIS-QoS "service 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 suite of recommendations require the use of the ITU-T H.225/ITU-T H.245 protocol for communication control between an 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 and 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 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 Bode. 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 head-end which implements the DOCSIS RFI MAC protocol and connects to CMs over an HFC network.
CMSS	CMS-to-CMS 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.
CSR	Customer Service Representative
DA	Directory Assistance
DE	Default. This is a DiffServ Per Hop Behavior.
DES	Data Encryption Standard
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.
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.
DSFID	Downstream Service Flow ID. See SFID
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
FEID	Financial Entity ID
FGD	Feature Group D signaling
FQDN	Fully Qualified Domain Name. Refer to IETF RFC 821 for details.
GC	Gate Controller
GTT	Global Title Translation
HFC	Hybrid Fiber/Coaxial cable). An HFC system is a broadband bi-directional shared media transmission system using fiber trunks between the head-end 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 Telecommunications Union
ITU-T	International Telecommunications Union–Telecommunications 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.
lsb	Least significant bit
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.
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.
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 head-end 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 and 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 numbers (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 Identifier
OSP	Operator Service Provider
OSS	Operations Support Systems. The back-office software used for configuration, performance, fault, accounting, and security management.
PAL	Phase Alternate Line. The European color television format that evolved from the American NTSC standard.
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.
PCSP	PacketCable CMS Subscriber Provisioning.
PDU	Protocol Data Unit
PHB	Per-Hop Behavior
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.
PS	Provisioning server.
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 2138 and RFC 2139) 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 3550.
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. Any 32-bit SFID must not conflict with a zero-extended 14-bit SID. SFIDs are considered to be in either the upstream direction (USFID) or downstream direction (DSFID). USFIDs and DSFIDs 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 variant 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
USFID	Upstream Service Flow ID. See SFID
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 decided to prepare 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®).. This is shown in simplified form in Figure 1.

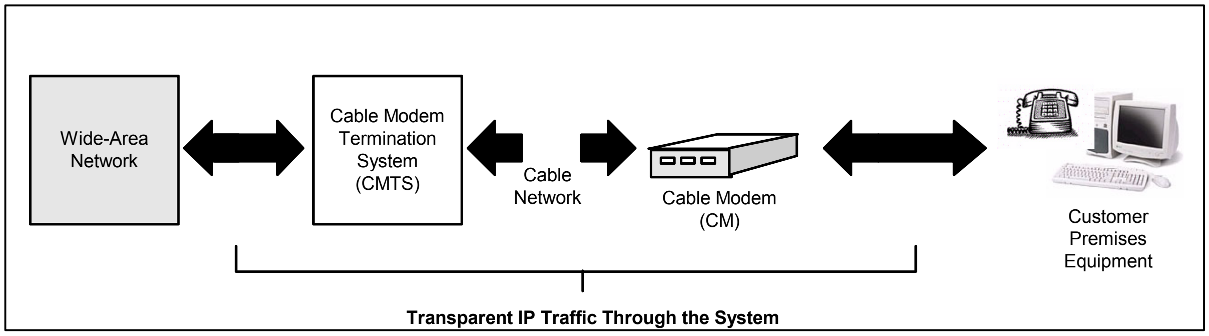
Wide-Area Network
Cable Modem Termination System (CMTS)
Cable Network
Cable Modem (CM)
Customer Premises Equipment
Transparent IP Traffic Through the System

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). At customer locations, the interface is called the cable-modem-to-customer-premises-equipment interface (CMCI) and is specified in [10]. 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 PacketCable Reference Architecture

Figure 2 shows the reference architecture for the PacketCable 1.5 Network. Refer to the PacketCable Architecture Document [1] for more detailed information on this reference architecture.

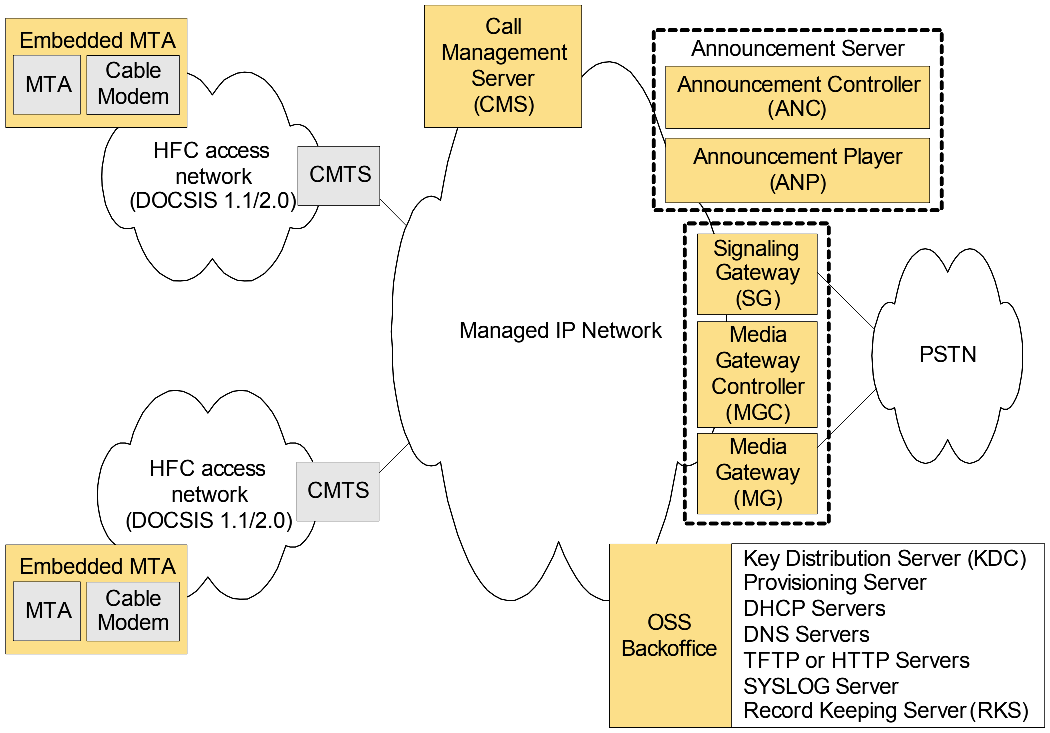
Embedded MTA
MTA
Cable Modem
HFC access network (DOCSIS 1.1/2.0)
CMTS
Call Management Server (CMS)
Announcement Server
Announcement Controller (ANC)
Announcement Player (ANP)
Signaling Gateway (SG)
Media Gateway Controller (MGC)
Media Gateway (MG)
PSTN
Managed IP Network
HFC access network (DOCSIS 1.1/2.0)
CMTS
Embedded MTA
MTA
Cable Modem
OSS Backoffice
Key Distribution Server (KDC)
Provisioning Server
DHCP Servers
DNS Servers
TFTP or HTTP Servers
SYSLOG Server
Record Keeping Server (RKS)

Figure 2. PacketCable 1.5 Network Component Reference Model

5.3 Components and Interfaces

Provisioning is defined as the operations necessary to provide a specified service to a customer. PacketCable service provisioning can be viewed as two distinct operations: MTA provisioning and CMS subscriber provisioning. Figure 3 presents the provisioning related interfaces maintained by the PS and other authorized Back Office Components to various PacketCable elements. Interfaces not explicitly labeled are undefined and out of scope for PacketCable.

This document is intended to set the requirements for the provisioning interface between the CMS and the PS or optionally other authorized Back Office Components. (Pkt-prov-p1).

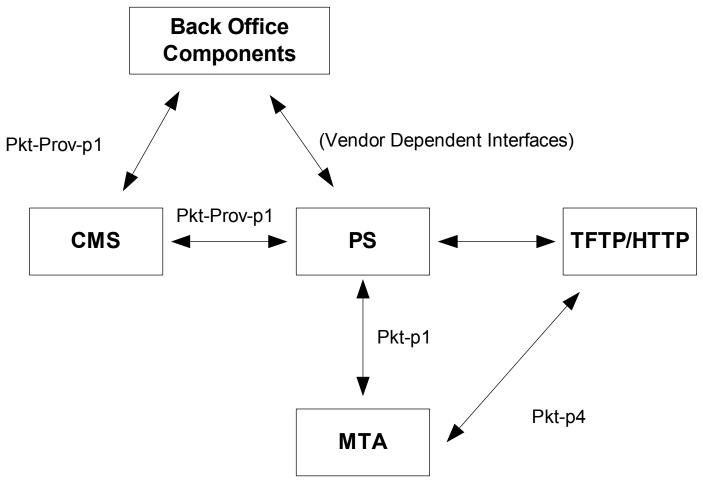

Figure 3. Provisioning Component Interfaces

5.4 Components

5.4.1 Back Office Components (Service Provider Business and Service Management Systems)

These are the Back Office Components that a service provider uses to manage customers and other components that make up their business. These systems provide the PacketCable provisioning process with service orders or optionally individual workflow tasks to activate services for customers. These systems may also receive accounting or usage data used to create customer-billing events.

5.4.2 Provisioning Server

This system forms the interface between the provider's Back Office Components and some or all of the PacketCable elements. PacketCable does not address the implementation of this system or its relationship to other OSSs that a service provider might employ.

The Provisioning Server is defined in [2] as consisting of a provisioning application that contains provisioning logic and a provisioning SNMP entity that provides access to active components. Here we will refer to the Provisioning Server without distinguishing between these two entities.

5.4.3 CMS

The Call Management Server component is described in [1]. This component provides call control and signaling-related services for the MTA and CMTS components in the PacketCable network.

5.4.4 MTA

A Media Terminal Adapter is a PacketCable client device that contains a subscriber-side interface to the customer's CPE (e.g., telephone) and a network side signaling interface to call control elements in the network. This component is described in [1].

5.4.5 TFTP

A configuration file service that is the basis for most device configuration in a PacketCable network. This could be a standalone TFTP Service that delivers statically-defined files to devices or a dynamic service that creates configurations on-the-fly from other data sources.

5.5 Interface Descriptions

5.5.1 Pkt-p1

This interface is defined in [2].

5.5.2 Pkt-p4

This interface is defined in [2].

5.5.3 Pkt-prov-p1

The interface defined in this document.

6 ASSUMPTIONS

- The Back Office components is responsible for coordinating endpoint updates with affected network entities (MTAs, CMTSs, etc.) and the CMS.
- CMS will not play a manager role nor does it specify SNMP communications to an MTA during CMS provisioning.
- The CMS and PS reside in the same secure provisioning domain. Security related information will be outlined in the PacketCable Security Specification [3].

7 SUBSCRIBER PROVISIONING

Subscriber provisioning consists of:

- Customer record/billing support.
- Equipment setup/configuration.

7.1 Customer Records (Billing)

Establishment of a customer record that contains the information needed to deliver service, bill, and collect payment from a customer. Customer record creation/billing is considered part of the back office OSS application and is currently out of scope for PacketCable.

7.2 Equipment Setup and Configuration

This may include physical installation and/or connection of equipment as well as any software and/or database updates necessary to actually deliver the service to the customer. Equipment setup affects two major components in a PacketCable environment.

- Customer premises equipment. For PacketCable, this is the MTA. The provisioning for the MTA is defined in [2] and will not be discussed in this document.
- Call Management Server. Provisioning of the CMS itself can be broken down into two main areas: basic POTS provisioning and call feature provisioning.

7.2.1 CMS Basic POTS Provisioning (BPP)

BPP provides the CMS with the minimal set of data necessary for routing of simple telephony service (POTS) in the PacketCable network. This minimal set of data consists of a telephone number mapped to its associated MTA's FQDN and NCS endpoint identifier. This data will be used to setup translation tables enabling the CMS to route calls to the appropriate device/port given a specific telephone number. BPP provisioning for each customer is required before that customer can receive any calls in a PacketCable network.

7.2.2 CMS Call Feature Provisioning (CFP)

In addition to BPP, CFP is performed to provide call features to a customer. CFP is more complicated than BPP as the parameters passed may vary on a feature-by-feature basis and may also be dependent on vendor specific implementations.

7.3 Static Versus Dynamic Subscriber Provisioning Data

Data required by the CMS for subscriber provisioning falls into two classifications:

1. Static, billed, permanently assigned service state. This data does not change call to call. Examples would be DQoS settings, call feature subscribed/non-subscribed states, caller ID information, etc.

2. Dynamic, non-billed, semi permanent service state. Often this information is subscriber alterable, either at an endpoint via *XX key code or via a web interface into the CMS. An example would be the user settable parameters of a call feature, such as Call Forward Busy Line (CFBL). The CFBL forwarding number is dynamic, non-billed service state. The subscribed/non-subscribed state of CFBL is static data maintained by the PS.

In the PacketCable CMS/PS scope, the PS owns all static provisioning state, and the CMS owns all dynamic provisioning state.

8 REQUIREMENTS

8.1 General Requirements

- The interface MUST make no assumptions regarding PS and CMS implementation technologies.

  Multiple partnering vendors will undoubtedly provide CMS and PS implementations on various hardware, software, and development language platforms. A platform and language neutral interface is required.

- The interface MUST support Basic POTS Provisioning.

  The interface's data model MUST include the minimum amount of information required to support basic POTS service.

- The interface MUST support Call Feature Provisioning.

  The interface's data model MUST support subscription to any PacketCable call feature.

- The interface's data model MUST be extensible.

  The present focus of the interface is telephony data. However, to the extent possible, the interface should be extensible for future PacketCable Multimedia services. It is desirable to have a single, extensible provisioning data model and transport to support all PacketCable features and capabilities, some of which are not yet defined.

- The interface MUST not impact any MTA operations in progress.

  Endpoint specific data may be added, deleted, or modified on the MTA without affecting other MTA endpoints or sessions in progress. CMS endpoint provisioning scenarios that would result in an endpoint/MTA to be taken out of service must be carefully documented.

- The interface MUST be capable of accommodating present (NCS) and future signaling protocols.

8.2 Transport Requirements

- The transport MUST make no assumptions regarding the physical networking infrastructure between a PS and a CMS.

  It is anticipated that multiple service providers will be interoperating over a single access network. Thus, multiple enterprises will be communicating, potentially using CMS and PS implementations from various vendors, over various network infrastructures (firewalls, proxies, etc.). The CMS/PS transport protocol should facilitate the ability to penetrate arbitrary network infrastructure.

- The transport MUST support unidirectional transfer of single data model objects from the PS to the CMS.

- The transport MUST support efficient streaming of multiple data model objects from the PS to the CMS.
- The transport MAY support unidirectional transfer of single data model objects from the CMS to the PS.
- The transport MAY support efficient streaming of multiple data model objects from the CMS to the PS.
- The transport MUST include semantics to support new, updated, and removed data model objects.
- The transport MUST support informational requests between PS and CMS.
- The transport MUST handle conditions such as CMS busy, errors etc.
- The transport MUST provide positive/negative acknowledgement of operation received.

  The transport MUST implement an at-least-once type of message semantics. The sender MUST not discard its request until the receiver acknowledges it (acknowledgements are not acknowledged.) The transport must be able to detect data corruption during transport, etc. and notify the sender of such conditions.
- The transport MUST provide positive/negative acknowledgement of operation handled.
- The PS MUST be able to initiate a transfer of data model objects ("push").
- The transport MUST be secure.

9 DATA MODEL

This section provides a high level description of the PCSP data model and its XML encoding. It is intended as descriptive and non-authoritative. The authoritative definition of the data model and its encoding is found in the PCSP XML schema in Appendix I

9.1 Overview

The data model for PacketCable CMS provisioning is displayed in Figure 4. It consists of two categories of entities:

- Objects
- Relations between objects

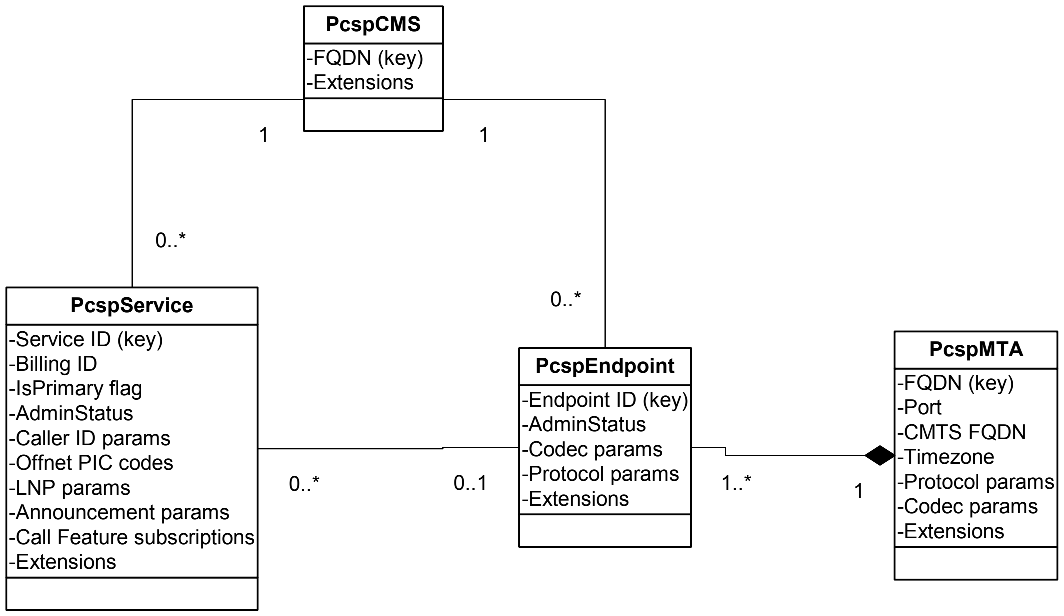

Figure 4. CMS Provisioning Data Model

The following entities MUST be supported:

- The PcspService object is the entity to which a PacketCable 1.5 customer subscribes. It represents a phone number and all related functionality (call features, etc).
- A PcspMTA object represents a Media Terminal Adapter, which aggregates one or more endpoints physically contained within the MTA.
- The PcspEndpoint object represents a physical endpoint on an MTA/Gateway.
- A PcspCMS object maintains associations between endpoints/CMSs and services/CMSs.
- PcspRelations represent associations between objects. In Figure 4, they are presented as connections between objects.

PcspService and PcspEndpoint are distinct objects in order to support multiple services (phone numbers) per endpoint. Distinct PcspMta and PcspEndpoint objects allow an MTA's endpoints to be managed by different service providers. The PcspCms object essentially maintains a collection of endpoints and services.

All objects are extensible.

9.1.1 PcspService Object

The service object is the entity to which a PacketCable 1.5 customer subscribes. It represents a phone number and all related functionality. The data model allows more than one service to be provisioned to a single endpoint.

The PcspService object contains the following generic information (for complete details, see the PCSP XML schema):

- ServiceId - a unique identifier for the service.
- BillingId - the identifier of another service, which will be billed for activity on this service.
- IsPrimary flag - with multiple services provisioned upon an endpoint, one service MUST have this flag set to indicate the default service to use for outgoing calls.
- PrimaryRingPattern - index into MTA cadence table, selecting a ring pattern for this service.
- Administrative status of this service (suspended, enabled, number has changed. etc.).
- DisplayName - the display information used for Call Name Delivery feature (CNAM).
- DisplayNumber - the display information used for Call Number Delivery feature (CND).
- Announcement settings (enable, language, time zone, etc.).
- Carrier codes (long distance carrier code, intra-lata carrier code, international carrier code).
- Local number portability control (porting status, STP lookup flag, etc).
- Call features - A service includes a list of subscribed call feature objects
- Extensions - This object is extensible in two locations: the main body of the object and the call feature list.

9.1.2 PcspMta Object

A Media Terminal Adapter aggregates one or more endpoints (physically contained within the MTA). It contains the following generic information (for complete details, see the PCSP XML schema):

- MTA's FQDN, uniquely identifying this MTA.
- MTA's NCS listener port (default: 2427)
- FQDN of controlling CMTS.

- Time zone within which this MTA is physically located.
- Signaling protocol designation. This is the default protocol selection for all contained endpoints, unless overridden by an individual endpoint.
- Codec designation - default codec selection for all contained endpoints, unless overridden by an individual endpoint.
- IPSec Control Flag - The IPSec Control Flag indicates whether IPSec is used for NCS Signaling between the CMS and the MTA. By default, IPSec is turned on all endpoints, but can be provisioned otherwise on a per endpoint basis.
- MTA Profile Name - Optional; An MTA Profile Indicator identifiable by the CMS.
- A single point for extension.

9.1.3 PktcEndpoint Object

An endpoint is a physical port on a MTA/Gateway. It contains the following generic information (for complete details, see the PCSP XML schema):

- EndpointId - uniquely identifies this endpoint.
- Signalling protocol selection. Optionally overrides MTA setting.
- Administrative status of the endpoint (disconnected, normal service, test mode, etc.).
- Codec selection. Optionally overrides MTA setting.
- IPSec Control Flag - Optionally overrides the MTA setting.
- A single point for extension.

9.1.4 PktcCMS Object

This object maintains associations between Endpoints/CMSs and Services/CMSs. It contains the following generic information (for complete details, see the PCSP XML schema):

- FQDN uniquely identifying this CMS.
- A single point for extension.

9.1.5 Inter Object Relationships

Within Figure 4, the lines connecting the classes represent object "relations" (sometimes called associations). The relations depicted in Figure 4 MUST be supported:

- Service/CMS. A typical CMS will own a block of phone numbers.
- Endpoint/CMS. An endpoint requires a CMS for signaling purposes.
- Service/Endpoint. A phone number must be attached to a physical endpoint.
- Endpoint/MTA. MTAs physically contain endpoints.

9.2 Relations are encoded using the PcspRelation entity.XML Encoding

Objects of the data model will be encoded using XML.

9.2.1 The PCSP XML Schema

Appendix I contains the PCSP XML schema. The schema defines the XML encoding syntax for the following entities (the entities MUST conform to the schema):

- The PcspService, PcspEndpoint, PcspMta, and PcspCms objects. These are the main data model objects.

- PcspRelation. This is used to establish or teardown relations between objects.
- PcspImportExport. A general purpose document format that can contain a large number of objects or relations. This will typically be used to export full data sets from a PS to a CMS.

The schema SHOULD be employed by validating XML parsers to determine syntactic correctness of encoded entities.

9.2.2 Sample PCSP Entity Encodings

Sample XML encodings of all the PCSP data model entities can be found in Appendix II.

9.2.3 Object Extensions

The PCSP XML schema permits extensions for all objects (PcspService, PcspEndpoint, PscpMta, and PcspCms). Extensions are accomplished via the <Extension> element placed in each object. Most objects specify this element at the end of the main body of the object. PcspService includes an additional <Extension> element at the end of the call feature list.

There are a few simple rules for the <Extension> element.

- All <Extension> elements MUST specify a namespace definition.
- All sub-elements of <Extension> must be namespace qualified.

These two rules permit the XML parsing system to validate the <Extension> content against a vendor supplied XML schema file. Appendix III contains an extension example.

10 MESSAGING

10.1 Overview

The PCSP interface follows a Web Service paradigm. The interface employs SOAP 1.1 messages to transfer XML encoded entities (from the PCSP data model) between client and server. Messages are transported between client and server using the HTTP 1.1 protocol. For a complete discussion of the transport considerations, see Appendix VI.

The interface is modeled on a synchronous request/response pattern (or remote procedure call – RPC). The following messaging patterns are supported between client and server.

- PUT message. Client writes one or more XML encoded objects or relations to the server. Both creation of new objects and modification of existing objects are supported.
- DELETE message. Client requests one or more objects or relations be deleted from the server.
- GET message. Read one or more XML encoded objects from the server (objects only…relations are not supported).
- CMDSTATUS message. Used to transfer "out of band" commands and status between the client and server. Client can notify server of various state conditions. Client can command server to perform various actions. This message is vendor extensible.

10.2 CMS and PS Messaging Role Requirements

In general, both CMS and PS MAY be implemented to fully support client and server messaging roles. However, within the scope of PacketCable CMS provisioning, the CMS and PS assume the role requirements specified below.

Table 1. CMS and PS Messaging Roles

Message	CMS as client	CMS as server	PS as client	PS as server
GET	OPTIONAL	MUST	MUST	OPTIONAL
PUT	OPTIONAL	MUST	MUST	OPTIONAL
DELETE	OPTIONAL	MUST	MUST	OPTIONAL
CMDSTATUS	MUST	MUST	MUST	MUST

The following points should be noted:

- A CMS MUST support the server role for GET, PUT, and DELETE.
- A PS MUST support the client role for GET, PUT, and DELETE.
- CMS and PS MUST support client and server roles for CMDSTATUS.
- All other behavior is OPTIONAL.

These requirements enforce provisioning data flows from the PS to the CMS and also ensure that the CMS is not required to push dynamic data changes (user adjustable call feature changes, etc.) back to the PS.

The PS is able to read specific objects from the CMS. This use case is supported primarily to allow the PS to retrieve user call feature settings ("dynamic data") owned by the CMS. This is accomplished by reading specific PcspService objects from the CMS.

Figure 5 displays all the required messaging roles.

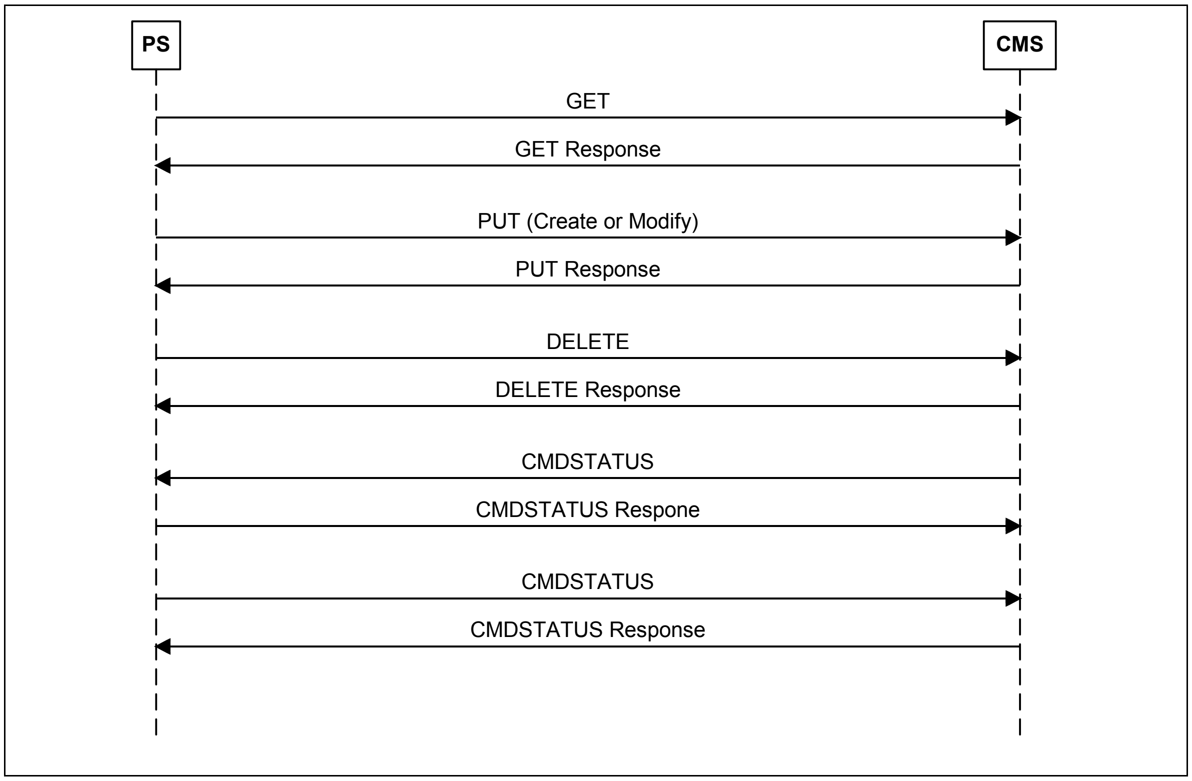

Figure 5. Required Messaging Flows

10.3 WSDL Specification

The PCSP interface is specified using Web Service Description Language 1.1. Just as with a Corba IDL, the WSDL interface definition specifies the remote methods on the interface, the arguments the methods accepts, the return values from the methods, and any interface specific data types that must be defined. Additionally, the WSDL definition also specifies the message encoding format (SOAP 1.1) and the transport binding (HTTP 1.1).

The WSDL is fed into various Web Services toolkits, available for most operating systems and languages, to automatically generate client interface stubs, server skeletons, and SOAP marshalling support.

PCSP clients and servers MUST conform to this WSDL definition presented in Appendix IV.

11 SECURITY

The PCSP interface is secured using the IPSec ESP protocol in transport mode. Key management is implemented using IKE with pre-shared keys. This security infrastructure is already used at the CMS for various interfaces. See [3] for full details.

Appendix I - PCSP XML Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
    PacketCable CMS Subscriber/Service Provisioning (PCSP) schema.

    PCSP defines a messaging interface and an XML encoding format for objects
    transmitted over that interface. This schema defines the XML encoding syntax for
    the objects transmitted over the PCSP interface.

    Encodings for PcspService, PcspEndpoint, PcspMta, and PcspCms objects are specified.
    A PcspRelation encoding describes associations between objects.
    A PcspImportExport encoding is used to produce instance documents containing large numbers of objects.
-->
<xs:schema targetNamespace="http://www.cablelabs.com/Pcsp/I01/schema"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns="http://www.cablelabs.com/Pcsp/I01/schema"
elementFormDefault="qualified">
    <!--
        ================= TYPE DEFINITIONS =====================
    -->
    <!--
        A non-empty string.
    -->
    <xs:simpleType name="nonEmptyString">
        <xs:restriction base="xs:string">
            <xs:minLength value="1"/>
        </xs:restriction>
    </xs:simpleType>
    <!--
        A Service ID.
        A non null string containing a "format" attribute (an enumeration), which defaults to NSN.
    -->
    <xs:complexType name="ServiceIdType">
        <xs:simpleContent>
            <xs:extension base="nonEmptyString">
                <xs:attribute name="format" default="NSN">
                    <xs:simpleType>
                        <xs:restriction base="xs:string">
                            <xs:enumeration value="NSN"/>
                            <xs:enumeration value="E164"/>
                            <xs:enumeration value="ENUM"/>
                            <xs:enumeration value="URL"/>
                        </xs:restriction>
                    </xs:simpleType>
                </xs:attribute>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
    <!--
        A relation operation type.
        Used to indicate relation is being "added" or "deleted".
    -->
    <xs:simpleType name="RelationOpType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="add"/>
            <xs:enumeration value="delete"/>
        </xs:restriction>
    </xs:simpleType>
    <!--
        An enumeration of legal object "class" names.
    -->
    <xs:simpleType name="classType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="PcspService"/>
            <xs:enumeration value="PcspCms"/>
            <xs:enumeration value="PcspEndpoint"/>
            <xs:enumeration value="PcspMta"/>
        </xs:restriction>
    </xs:simpleType>
    <!--
        A list of object keys.
    -->
    <xs:complexType name="ListOfKeys">
        <xs:sequence>
            <xs:element name="Key" type="xs:string" minOccurs="1" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
    <!--
        Codec types.
        An enumeration that matches the PktcCodecType object from the
        PacketCable™ Audio/Video Codecs Specification PKT-SP-CODEC1.5-D02-041210.
        This enumeration should be kept in sync with the aforementioned specification.

        For convenience, value definitions are repeated here...

             1: other.
             2: unknown.
             3: G729
             4: reserved
             5: G729E
             6: PCMU
             7: G726-32
             8: G728
             9: PCMA
            10: G726-16
            11: G726-24
            12: G726-40
            13. iLBC
            14. BV-16

        In the case of the PcspEndpoint object, a codec value of 2 (unknown) shall be
        interpreted as "use the MTA's codec specification".
    -->
    <xs:simpleType name="codecType">
        <xs:restriction base="xs:integer">
            <xs:enumeration value="1"/>
            <xs:enumeration value="2"/>
            <xs:enumeration value="3"/>
            <xs:enumeration value="4"/>
            <xs:enumeration value="5"/>
            <xs:enumeration value="6"/>
            <xs:enumeration value="7"/>
            <xs:enumeration value="8"/>
            <xs:enumeration value="9"/>
            <xs:enumeration value="10"/>
            <xs:enumeration value="11"/>
            <xs:enumeration value="12"/>
            <xs:enumeration value="13"/>
            <xs:enumeration value="14"/>
        </xs:restriction>
    </xs:simpleType>
    <!--
        Signalling protocol designations.
        PcspEndpoint employs "MtaDefault" to force use of the MTA's default protocol setting.
    -->
    <xs:simpleType name="protocolType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="MCGP 1.0 NCS 1.0"/>
            <xs:enumeration value="MtaDefault"/>
        </xs:restriction>
```

```
</xs:simpleType>
<!--
    Numeric timezone designation per RFC 1123.
-->
<xs:simpleType name="timezoneType">
    <xs:restriction base="xs:string">
        <xs:pattern value="[\+\-]\d{4}"/>
    </xs:restriction>
</xs:simpleType>
<!--
    Local number PortingStatus

        0: not ported.
        1: ported in (owned by another TSP)
        2: ported out (loaned to another TSP)
-->
<xs:simpleType name="portingStatusType">
    <xs:restriction base="xs:integer">
        <xs:enumeration value="0"/>
        <xs:enumeration value="1"/>
        <xs:enumeration value="2"/>
    </xs:restriction>
</xs:simpleType>
<!--
    ================= SUPPORTING ELEMENT DEFINITIONS =================
-->
<!--
    Network announcement control. Contains...

    Language - per XML schema language type.

    Timezone - see previous definition.
-->
<xs:element name="Announcements">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="Language" type="xs:language"/>
            <xs:element name="Timezone" type="timezoneType"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
<!--
    Interexchange carrier codes.  Used to route offnet regional, long distance,
    and international calls to specific carriers.

    PIC - Predesignated interexchange carrier (long distance).

    LPIC - Predesignated intra-lata carrier

    IPIC - Predesignated international carrier
-->
<xs:element name="InterExchange">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="PIC" type="xs:string"/>
            <xs:element name="LPIC" type="xs:string"/>
            <xs:element name="IPIC" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
<!--
    Local Number Portability parameters.

    PortingStatus - see portingStatusType.

    LNPT - LNP trigger determines if number is in transition.
```

```
            false/0: no STP lookup required.
            true/1: STP lookup required to determine LRN of destination switch.
-->
<xs:element name="LNP">
      <xs:complexType>
            <xs:sequence>
                  <xs:element name="PortingStatus" type="portingStatusType"/>
                  <xs:element name="LNPT" type="xs:boolean"/>
            </xs:sequence>
      </xs:complexType>
</xs:element>
<!--
      Vendor Extension element.

      Used within the PcspService, PcspCms, PcspMta, and PcspEndpoint objects to enable vendor extensions.
      Also used to extend the call feature list within the PcspService object.
-->
<xs:element name="Extension">
      <xs:complexType>
            <xs:sequence>
                  <xs:any namespace="##any" processContents="strict" minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
      </xs:complexType>
</xs:element>
<!--
      ====================== Call Features. ==========================

      A service includes a list of call feature objects, each encoding one of the call features
      described in PKT-TR-VOIPBRF-R01-000608 and PKT-TR-VOIPERF-R01-000831.

      Each call feature includes its "static" state data (owned by the PS):
            Feature name (implicitly as the element name),
            Subscribed/non subscribed state,
            Administrative state the feature.

      Many call features include just this information.
      Absence of a specific call feature implies the feature is not subscribed.
      The subscribed state is used to indicate that an
      explicitly listed call feature is not subscribed (an atypical case).

      Several features extend the "static" parameter set with feature specific data.
      This feature specific data is typically configured by the user (via handset or by calling a CSR).
      The PCSP spec classifies the user adjustable data as "dynamic", meaning that it is owned
      by the CMS.  Changes to the dynamic data in the CMS are not required to be pushed back to the PS.
-->
<!--
      Always –
            false/0: Subscriber may change forward-to number.
            true/1: Service provider (only) may change forward-to number
-->
<xs:element name="Always" type="xs:boolean"/>
<!--
      ForwardTo - Service Id to which call will be forwarded.
      Note: empty strings are allowed.
-->
<xs:element name="ForwardTo" type="xs:string"/>
<!--
      ListOfServiceId - a list of Service Ids.
-->
<xs:element name="ListOfServiceId">
      <xs:complexType>
            <xs:sequence>
                  <xs:element name="ServiceId" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
      </xs:complexType>
</xs:element>
```

```
<!--
     ListOfSpeedDial - list of Service Ids / speed dial # pairs.
     Each pair contains a one or two digit speed dial number and its associated service id.
-->
<xs:element name="SdPair">
     <xs:complexType>
          <xs:sequence>
               <xs:element name="SdNum">
                    <xs:simpleType>
                         <xs:restriction base="xs:integer">
                              <xs:minInclusive value="0"/>
                              <xs:maxInclusive value="99"/>
                         </xs:restriction>
                    </xs:simpleType>
               </xs:element>
               <xs:element name="ServiceId" type="xs:string"/>
          </xs:sequence>
     </xs:complexType>
</xs:element>
<xs:element name="ListOfSpeedDial">
     <xs:complexType>
          <xs:sequence>
               <xs:element ref="SdPair" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
     </xs:complexType>
</xs:element>
<!--
     The definition of each of the supported call features...

     All call features can be considered in two parts.
     1. Common section containing the administrative state of the feature (the "static" data).
     2. An optional section containing feature specific parameters, typically set by the end user (the "dynamic"
data).
-->
<!--
     The "base object" for all call features, containing:

     Subscribed -
          0/false: feature is not subscribed
          1/true: feature is subscribed.

     UsageBilling -
          0/false: do not generate billing records on feature usage
          1/true: generate billing records on feature usage.

     AdminStatus -
               0: feature is suspended by service provider.
               1: feature is enabled by service provider..

     In general, presence of a call feature implies that it is subscribed.
     The  Subscribed flag is supported for the atypical case of wanting to
     indicate that an explicitly listed call feature is not subscribed.
-->
<xs:complexType name="CfBase">
     <xs:sequence>
          <xs:element name="Subscribed" type="xs:boolean"/>
          <xs:element name="UsageBilling" type="xs:boolean" minOccurs = "0"/>
          <xs:element name="AdminStatus">
               <xs:simpleType>
                    <xs:restriction base="xs:int">
                         <xs:enumeration value="0"/>
                         <xs:enumeration value="1"/>
                    </xs:restriction>
               </xs:simpleType>
          </xs:element>
     </xs:sequence>
```

```
</xs:complexType>
<!--
    "CND" Calling Number Delivery
-->
<xs:element name="CfCND">
    <xs:complexType>
        <xs:complexContent>
            <xs:extension base="CfBase"/>
        </xs:complexContent>
    </xs:complexType>
</xs:element>
<!--
    "CNAM": Calling Name Delivery
-->
<xs:element name="CfCNAM">
    <xs:complexType>
        <xs:complexContent>
            <xs:extension base="CfBase"/>
        </xs:complexContent>
    </xs:complexType>
</xs:element>
<!--
    "CIDCW": Calling Identity Delivery on Call  Waiting
-->
<xs:element name="CfCIDCW">
    <xs:complexType>
        <xs:complexContent>
            <xs:extension base="CfBase"/>
        </xs:complexContent>
    </xs:complexType>
</xs:element>
<!--
    "CW": Call Waiting
-->
<xs:element name="CfCW">
    <xs:complexType>
        <xs:complexContent>
            <xs:extension base="CfBase"/>
        </xs:complexContent>
    </xs:complexType>
</xs:element>
<!--
    "CCW": Cancel Call Waiting (*70)
-->
<xs:element name="CfCCW">
    <xs:complexType>
        <xs:complexContent>
            <xs:extension base="CfBase"/>
        </xs:complexContent>
    </xs:complexType>
</xs:element>
<!--
    "CFV": Call Forwarding Variable and Usage- Sensitive Call Forwarding (*72/*73)

    Extends CfBase with the following:

    Active -
        0/false: user has deactivated feature (*73).
        1/true: user has activated feature (*72).
-->
<xs:element name="CfCFV">
    <xs:complexType>
        <xs:complexContent>
            <xs:extension base="CfBase">
                <xs:sequence>
                    <xs:element name="UserParams" minOccurs="0">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="Active" type="xs:boolean"/>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
<!--
    "AR": Automatic Recall (*69)
-->
<xs:element name="CfAR">
    <xs:complexType>
        <xs:complexContent>
            <xs:extension base="CfBase"/>
        </xs:complexContent>
    </xs:complexType>
</xs:element>
<!--
    "AC": Automatic Callback (*66)
-->
<xs:element name="CfAC">
    <xs:complexType>
        <xs:complexContent>
            <xs:extension base="CfBase"/>
        </xs:complexContent>
    </xs:complexType>
</xs:element>
<!--
    "VMWI": Visual Message Waiting Indicator

    Extends CfBase with the following:

    Indicator Type -
            0: None.
            1: Stutter Dial tone Only
            2: Message Lamp Only
            3: Both Stutter Dial tone and Message Lamp
-->
<xs:element name="CfVMWI">
    <xs:complexType>
        <xs:complexContent>
            <xs:extension base="CfBase">
                <xs:sequence>
                    <xs:element name="UserParams" minOccurs="0">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element name="Type">
                                    <xs:simpleType>
                                        <xs:restriction base="xs:int">
                                            <xs:enumeration value="0"/>
                                            <xs:enumeration value="1"/>
                                            <xs:enumeration value="2"/>
                                            <xs:enumeration value="3"/>
                                        </xs:restriction>
                                    </xs:simpleType>
                                </xs:element>
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:element>
<!--
    "COT": Customer Originated Trace (*57)
-->
<xs:element name="CfCOT">
    <xs:complexType>
        <xs:complexContent>
            <xs:extension base="CfBase"/>
        </xs:complexContent>
    </xs:complexType>
</xs:element>
<!--
    "TWC": Three-Way Calling / Usage-Sensitive Three-Way Calling (*71)
-->
<xs:element name="CfTWC">
    <xs:complexType>
        <xs:complexContent>
            <xs:extension base="CfBase"/>
        </xs:complexContent>
    </xs:complexType>
</xs:element>
<!--
    "RACF": Remote Activation of Call Forwarding
-->
<xs:element name="CfRACF">
    <xs:complexType>
        <xs:complexContent>
            <xs:extension base="CfBase"/>
        </xs:complexContent>
    </xs:complexType>
</xs:element>
<!--
    "OCAA": Outside Calling Area Alerting
-->
<xs:element name="CfOCAA">
    <xs:complexType>
        <xs:complexContent>
            <xs:extension base="CfBase"/>
        </xs:complexContent>
    </xs:complexType>
</xs:element>
<!--
    "CIES": Calling Identity with Enhanced Screening
-->
<xs:element name="CfCIES">
    <xs:complexType>
        <xs:complexContent>
            <xs:extension base="CfBase"/>
        </xs:complexContent>
    </xs:complexType>
</xs:element>
<!--
    "ACR": Anonymous Call Rejection (*77 / *87)

    Extends CfBase with the following:

    Active -
        0/false: user has deactivated feature (*87).
        1/true: user has activated feature (*77).
-->
<xs:element name="CfACR">
    <xs:complexType>
        <xs:complexContent>
            <xs:extension base="CfBase">
                <xs:sequence>
```

```
                        <xs:element name="UserParams" minOccurs="0">
                            <xs:complexType>
                                <xs:sequence>
                                    <xs:element name="Active" type="xs:boolean"/>
                                </xs:sequence>
                            </xs:complexType>
                        </xs:element>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>
    <!--
        "AC-R": Automatic Callback – Restrict
    -->
    <xs:element name="CfACRestrict">
        <xs:complexType>
            <xs:complexContent>
                <xs:extension base="CfBase"/>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>
    <!--
        "ACB": Automatic Recall Blocking
    -->
    <xs:element name="CfACB">
        <xs:complexType>
            <xs:complexContent>
                <xs:extension base="CfBase"/>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>
    <!--
        "CIDB" Calling Identity Delivery Blocking (*67 / *82).

        Extends CfBase with the following:

        Flag -
            "PUBLIC": deliver Caller ID info
            "ANONYMOUS": do not deliver Caller ID info.
    -->
    <xs:element name="CfCIDB">
        <xs:complexType>
            <xs:complexContent>
                <xs:extension base="CfBase">
                    <xs:sequence>
                        <xs:element name="UserParams" minOccurs="0">
                            <xs:complexType>
                                <xs:sequence>
                                    <xs:element name="Flag">
                                        <xs:simpleType>
                                            <xs:restriction base="xs:string">
                                                <xs:enumeration value="PUBLIC"/>
                                                <xs:enumeration value="ANONYMOUS"/>
                                            </xs:restriction>
                                        </xs:simpleType>
                                    </xs:element>
                                </xs:sequence>
                            </xs:complexType>
                        </xs:element>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>
    <!--
```

```
        "CFBL" Call Forwarding Busy Line ( *68 / *40 / *88 ).

        Extends CfBase with the following "dynamic", user
        adjustable parameters (owned by the CMS).

        Active -
            0/false: user has deactivated feature (*88).
            1/true: user has activated feature (*68/*40).

        Always - see previous definition.

        ForwardTo - see previous definition.
    -->
    <xs:element name="CfCFBL">
        <xs:complexType>
            <xs:complexContent>
                <xs:extension base="CfBase">
                    <xs:sequence>
                        <xs:element name="UserParams" minOccurs="0">
                            <xs:complexType>
                                <xs:sequence>
                                    <xs:element name="Active" type="xs:boolean"/>
                                    <xs:element ref="Always"/>
                                    <xs:element ref="ForwardTo" minOccurs="0"/>
                                </xs:sequence>
                            </xs:complexType>
                        </xs:element>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>
    <!--
        "CFDA" Call Forwarding Don't Answer (*68 / *42 / *88)

        Extends CfBase with the following "dynamic", user
        adjustable parameters (owned by the CMS):

        Active -
            0/false: user has deactivated feature (*88).
            1/true: user has activated feature (*68/*42).

        Always - see previous definition.

        RingPeriod - number of ringing cycles after which forwarding is activated.

        ForwardTo - see previous definition.
    -->
    <xs:element name="CfCFDA">
        <xs:complexType>
            <xs:complexContent>
                <xs:extension base="CfBase">
                    <xs:sequence>
                        <xs:element name="UserParams" minOccurs="0">
                            <xs:complexType>
                                <xs:sequence>
                                    <xs:element name="Active" type="xs:boolean"/>
                                    <xs:element ref="Always"/>
                                    <xs:element name="RingPeriod" type="xs:int"/>
                                    <xs:element ref="ForwardTo" minOccurs="0"/>
                                </xs:sequence>
                            </xs:complexType>
                        </xs:element>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>
    <!--
        "CFC" Call Forwarding Combination

        Extends CfBase with the following "dynamic", user
        adjustable parameters (owned by the CMS):

        Active -
            0/false: user has deactivated feature (*88).
            1/true: user has activated feature (*68).

        Always - see previous definition.

        RingPeriod - number of ringing cycles after which forwarding is activated.

        ForwardTo - see previous definition.
    -->
    <xs:element name="CfCFC">
        <xs:complexType>
            <xs:complexContent>
                <xs:extension base="CfBase">
                    <xs:sequence>
                        <xs:element name="UserParams" minOccurs="0">
                            <xs:complexType>
                                <xs:sequence>
                                    <xs:element name="Active" type="xs:boolean"/>
                                    <xs:element ref="Always"/>
                                    <xs:element name="RingPeriod" type="xs:int"/>
                                    <xs:element ref="ForwardTo" minOccurs="0"/>
                                </xs:sequence>
                            </xs:complexType>
                        </xs:element>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>
    <!--
        "SCF" Selective Call Forwarding (*63/*83).

        Extends CfBase with the following "dynamic", user
        adjustable parameters (owned by the CMS):

        Active -
            0/false: user has deactivated feature (*83).
            1/true: user has activated feature (*63).

        ListOfServiceId - list of service identifiers that will be forwarded.  See previous element definition.

        ForwardTo - the service to which to forward. See previous element definition.
    -->
    <xs:element name="CfSCF">
        <xs:complexType>
            <xs:complexContent>
                <xs:extension base="CfBase">
                    <xs:sequence>
                        <xs:element name="UserParams" minOccurs="0">
                            <xs:complexType>
                                <xs:sequence>
                                    <xs:element name="Active" type="xs:boolean"/>
                                    <xs:element ref="ListOfServiceId" minOccurs="0"/>
                                    <xs:element ref="ForwardTo" minOccurs="0"/>
                                </xs:sequence>
                            </xs:complexType>
                        </xs:element>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>
    <!--
        "SCA" Selective Call Acceptance (*64 / *84 ).

        Extends CfBase with the following "dynamic", user
        adjustable parameters (owned by the CMS):

        Active -
            0/false: user has deactivated feature (*84).
            1/true: user has activated feature (*66).

        ListOfServiceIds - list of service identifiers that will be accepted.  See previous element definition.
    -->
    <xs:element name="CfSCA">
        <xs:complexType>
            <xs:complexContent>
                <xs:extension base="CfBase">
                    <xs:sequence>
                        <xs:element name="UserParams" minOccurs="0">
                            <xs:complexType>
                                <xs:sequence>
                                    <xs:element name="Active" type="xs:boolean"/>
                                    <xs:element ref="ListOfServiceId" minOccurs="0"/>
                                </xs:sequence>
                            </xs:complexType>
                        </xs:element>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>
    <!--
        "SCR" Selective Call Rejection (*60 / *80 ).

        Extends CfBase with the following "dynamic", user
        adjustable parameters (owned by the CMS):

        Active -
            0/false: user has deactivated feature (*80).
            1/true: user has activated feature (*60).

        ListOfServiceIds - list of service identifiers that will be rejected.  See previous element definition.
    -->
    <xs:element name="CfSCR">
        <xs:complexType>
            <xs:complexContent>
                <xs:extension base="CfBase">
                    <xs:sequence>
                        <xs:element name="UserParams" minOccurs="0">
                            <xs:complexType>
                                <xs:sequence>
                                    <xs:element name="Active" type="xs:boolean"/>
                                    <xs:element ref="ListOfServiceId" minOccurs="0"/>
                                </xs:sequence>
                            </xs:complexType>
                        </xs:element>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>
    <!--
        "DRCW" Distinctive Ringing/Call Waiting (*61 / *81)

        Extends CfBase with the following "dynamic", user
        adjustable parameters (owned by the CMS):

        Active -
            0/false: user has deactivated feature (*81).
            1/true: user has activated feature (*61).

        ListOfServiceIds - list of incoming service identifiers  that will receive the distinctive ring
            treatment (vs. standard power ring or call waiting tone).  See previous element definition.
    -->
    <xs:element name="CfDRCW">
        <xs:complexType>
            <xs:complexContent>
                <xs:extension base="CfBase">
                    <xs:sequence>
                        <xs:element name="UserParams" minOccurs="0">
                            <xs:complexType>
                                <xs:sequence>
                                    <xs:element name="Active" type="xs:boolean"/>
                                    <xs:element ref="ListOfServiceId" minOccurs="0"/>
                                </xs:sequence>
                            </xs:complexType>
                        </xs:element>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>
    <!--
        "SPCALL" Speed Calling (*74 / *75)

        Extends CfBase with the following "dynamic", user
        adjustable parameters (owned by the CMS):

        ListOfSpeedDial - see previous element definition.
    -->
    <xs:element name="CfSPCALL">
        <xs:complexType>
            <xs:complexContent>
                <xs:extension base="CfBase">
                    <xs:sequence>
                        <xs:element name="UserParams" minOccurs="0">
                            <xs:complexType>
                                <xs:sequence>
                                    <xs:element ref="ListOfSpeedDial"/>
                                </xs:sequence>
                            </xs:complexType>
                        </xs:element>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>
    <!--
        "RDA" Residence Distinctive Alerting Service.
    -->
    <xs:element name="CfRDA">
        <xs:complexType>
            <xs:complexContent>
                <xs:extension base="CfBase"/>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>
    <!--
```

```
    "LSR" Line Service Restriction.

    Extends CfBase with the following "dynamic", user
    adjustable parameters (owned by the CMS):

    BlkDomLongDist - block for outgoing domestic long distance calls.
        0/false: not blocked.
        1/true: blocked.
    BlkIntlLongDist - block for outgoing international long distance calls.
        0/false: not blocked.
        1/true: blocked.
    BlkPayPerCall - block for outgoing pay per calls (900/976).
        0/false: not blocked.
        1/true: blocked.
    BlkOperatorAssist - block for outgoing operator assisted calls.
        0/false: not blocked.
        1/true: blocked.
    BlkDirAssist - block for outgoing directory assistance calls.
        0/false: not blocked.
        1/true: blocked.
    BlkTollFree - block for outgoing toll free calls.
        0/false: not blocked.
        1/true: blocked.
    Active -
        0/false: user has deactivated feature (*82).
        1/true: user has activated feature.
    PIN – code to enter to deactivate blocking
    ServiceList - list of service identifiers for domestic long distance calls that are always allowed.
-->
<xs:element name="CfLSR">
    <xs:complexType>
        <xs:complexContent>
            <xs:extension base="CfBase">
                <xs:sequence>
                    <xs:element name="UserParams" minOccurs="0">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element name="BlkDomLongDist" type="xs:boolean" minOccurs="0"/>
                                <xs:element name="BlkIntLongDist" type="xs:boolean" minOccurs="0"/>
                                <xs:element name="BlkPayPerCall" type="xs:boolean" minOccurs="0"/>
                                <xs:element name="BlkOperatorAssist" type="xs:boolean" minOccurs="0"/>
                                <xs:element name="BlkDirAssist" type="xs:boolean" minOccurs="0"/>
                                <xs:element name="BlkTollFree" type="xs:boolean" minOccurs="0"/>
                                <xs:element name="PIN" type="xs:string" minOccurs ="0"/>
                                <xs:element name="Active" type="xs:boolean"/>
                                <xs:element ref="ListOfServiceId" minOccurs="0"/>
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:element>
<!--
    "DND" Do Not Disturb

    Extends CfBase with the following "dynamic", user
    adjustable parameters (owned by the CMS):

    Active -
        0/false: user has deactivated feature.
        1/true: user has activated feature.

    WeekDayStartTod1 - week day start time for DND.
    WeekDayStopTod1 - week day stop time for DND.
```

```
        WeekDayStartTod2 - week day start time for DND.
        WeekDayStopTod2 - week day stop time for DND.
        WeekEndStartTod1 - week end start time for DND.
        WeekEndStopTod1 - week end stop time for DND.
        WeekEndStartTod2 - week end start time for DND.
        WeekEndStopTod2 - week end stop time for DND.
    -->
    <xs:element name="CfDND">
        <xs:complexType>
            <xs:complexContent>
                <xs:extension base="CfBase">
                    <xs:sequence>
                        <xs:element name="UserParams" minOccurs="0">
                            <xs:complexType>
                                <xs:sequence>
                                    <xs:element name="Active" type="xs:boolean"/>
                                    <xs:element name="WdStartTod1" type="xs:time" minOccurs="0"/>
                                    <xs:element name="WdStopTod1" type="xs:time" minOccurs="0"/>
                                    <xs:element name="WdStartTod2" type="xs:time" minOccurs="0"/>
                                    <xs:element name="WdStopTod2" type="xs:time" minOccurs="0"/>
                                    <xs:element name="WeStartTod1" type="xs:time" minOccurs="0"/>
                                    <xs:element name="WeStopTod1" type="xs:time" minOccurs="0"/>
                                    <xs:element name="WeStartTod2" type="xs:time" minOccurs="0"/>
                                    <xs:element name="WeStopTod2" type="xs:time" minOccurs="0"/>
                                </xs:sequence>
                            </xs:complexType>
                        </xs:element>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>
    <!--
        "COC" Curfew on Calls.

        Extends CfBase with the following "dynamic", user
        adjustable parameters (owned by the CMS):

        Active -
            0/false: user has deactivated feature.
            1/true: user has activated feature.

        StartTod - start time for COC.
        StopTod - stop time for COC.
        ServiceList - list of service identifiers for incoming and outgoing services which are
            allowed to bypass the NSA.
    -->
    <xs:element name="CfCOC">
        <xs:complexType>
            <xs:complexContent>
                <xs:extension base="CfBase">
                    <xs:sequence>
                        <xs:element name="UserParams" minOccurs="0">
                            <xs:complexType>
                                <xs:sequence>
                                    <xs:element name="Active" type="xs:boolean"/>
                                    <xs:element name="StartTod" type="xs:time"/>
                                    <xs:element name="StopTod" type="xs:time"/>
                                    <xs:element ref="ListOfServiceId" minOccurs="0"/>
                                </xs:sequence>
                            </xs:complexType>
                        </xs:element>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
```

```
</xs:element>
<!--
    "NSA" No Solicitation Announcement

    Extends CfBase with the following "dynamic", user
    adjustable parameters (owned by the CMS):

    Active -
        0/false: user has deactivated feature.
        1/true: user has activated feature.

    StartTod - start time for COC.
    StopTod - stop time for COC.
    ServiceList - list of service identifiers for incoming and outgoing services which are
        allowed to bypass the NSA.
-->
<xs:element name="CfNSA">
    <xs:complexType>
        <xs:complexContent>
            <xs:extension base="CfBase">
                <xs:sequence>
                    <xs:element name="UserParams" minOccurs="0">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element name="Active" type="xs:boolean"/>
                                <xs:element name="StartTod" type="xs:time"/>
                                <xs:element name="StopTod" type="xs:time"/>
                                <xs:element ref="ListOfServiceId" minOccurs="0"/>
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:element>
<!--
    A list of call features.  The list may contain at most 1 of each of the
    features outlined above, along with any vendor extension call features.
-->
<xs:element name="ListOfCallFeatures">
    <xs:complexType>
        <xs:all>
            <xs:element ref="CfCND" minOccurs="0"/>
            <xs:element ref="CfCNAM" minOccurs="0"/>
            <xs:element ref="CfCIDCW" minOccurs="0"/>
            <xs:element ref="CfCW" minOccurs="0"/>
            <xs:element ref="CfCCW" minOccurs="0"/>
            <xs:element ref="CfCFV" minOccurs="0"/>
            <xs:element ref="CfAR" minOccurs="0"/>
            <xs:element ref="CfAC" minOccurs="0"/>
            <xs:element ref="CfVMWI" minOccurs="0"/>
            <xs:element ref="CfCOT" minOccurs="0"/>
            <xs:element ref="CfTWC" minOccurs="0"/>
            <xs:element ref="CfRACF" minOccurs="0"/>
            <xs:element ref="CfOCAA" minOccurs="0"/>
            <xs:element ref="CfCIES" minOccurs="0"/>
            <xs:element ref="CfACR" minOccurs="0"/>
            <xs:element ref="CfACRestrict" minOccurs="0"/>
            <xs:element ref="CfACB" minOccurs="0"/>
            <xs:element ref="CfCIDB" minOccurs="0"/>
            <xs:element ref="CfCFBL" minOccurs="0"/>
            <xs:element ref="CfCFDA" minOccurs="0"/>
            <xs:element ref="CfCFC" minOccurs="0"/>
            <xs:element ref="CfSCF" minOccurs="0"/>
            <xs:element ref="CfSCA" minOccurs="0"/>
```

```
                <xs:element ref="CfSCR" minOccurs="0"/>
                <xs:element ref="CfDRCW" minOccurs="0"/>
                <xs:element ref="CfSPCALL" minOccurs="0"/>
                <xs:element ref="CfRDA" minOccurs="0"/>
                <xs:element ref="CfLSR" minOccurs="0"/>
                <xs:element ref="CfDND" minOccurs="0"/>
                <xs:element ref="CfCOC" minOccurs="0"/>
                <xs:element ref="CfNSA" minOccurs="0"/>
                <xs:element ref="Extension" minOccurs="0"/>
            </xs:all>
        </xs:complexType>
    </xs:element>
    <!--
        ====================== MAIN OBJECT DEFINITIONS ======================

        There are 6 encodings defined in the PCSP schema.

        The 4 main object encodings:

            PcspCms - a CMS.  A collection of Services and Endpoints.
            PcspService - represents a phone number, its configuration, and call features.
            PcspMta - represents a physical MTA and its configuration.  A collection of Endpoints.
            PcspEndpoint - represents an Endpoint on an MTA.

        A PcspRelation object.  This object encodes the associations between objects.

        A PcspImportExport object. This is used to produce a bulk loading file for the CMS.
    -->
    <!--
        PcspRelation.

        The relation object specifies inter-object associations between the PcspCms,
        PcspService, PcspEndpoint, and PcspMta objects.

        The "relOp" attribute specified if the relation is being added or deleted.
    -->
    <xs:element name="PcspRelation">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="Class1" type="classType"/>
                <xs:element name="Key" type="xs:string"/>
                <xs:element name="Class2" type="classType"/>
                <xs:element name="ListOfKeys" type="ListOfKeys"/>
            </xs:sequence>
            <xs:attribute name="relOp" type="RelationOpType" use="required"/>
        </xs:complexType>
    </xs:element>
    <!--
        The PcspCms object.

        This object maintains associations between Endpoints, Services, and their managing CMSs.

        Contents...

        CmsFqdn - FQDN uniquely identifying this CMS.
    -->
    <xs:element name="PcspCms">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="CmsFqdn" type="nonEmptyString"/>
                <xs:element ref="Extension" minOccurs="0"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <!--
        The PcspEndpoint object.
        An endpoint is a physical port on a MTA/Gateway.

        Contents...

        EndpointId - Uniquely identifies this endpoint.   Format per
            "PacketCable Network Based Call Signalling Protocol Specification".
            Example: "aaln/1@mta01.cablelabs.com"

        AdminStatus –
            0: endpoint is disconnected
            1: normal - endpoint is in service
            2: test mode - endpoint is under test.

        Protocol - optional override for MTA protocol setting.

        Codec - optional override for MTA codec setting.

        IPSecControl – optional override for the MTA IPSecControl setting.
    -->
    <xs:element name="PcspEndpoint">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="EndpointId" type="nonEmptyString"/>
                <xs:element name="AdminStatus">
                    <xs:simpleType>
                        <xs:restriction base="xs:integer">
                            <xs:enumeration value="0"/>
                            <xs:enumeration value="1"/>
                            <xs:enumeration value="2"/>
                        </xs:restriction>
                    </xs:simpleType>
                </xs:element>
                <xs:element name="Protocol" type="protocolType" minOccurs="0"/>
                <xs:element name="Codec" type="codecType" minOccurs="0"/>
                <xs:element name="IPSecControl" type="xs:boolean" minOccurs="0"/>
                <xs:element ref="Extension" minOccurs="0"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <!--
        The PcspMta object.

        A Media Terminal Adapter aggregates one or more endpoints (physically contained within the MTA).

        Contents...

        MtaFqdn - MTA's FQDN, uniquely identifying this MTA.
        MtaPort - MTA's NCS listening port (default: 2427)
        CmtsFqdn - FQDN of controlling CMTS.  CMS needs this to establish MTA DQoS with correct CMTS.
        MtaProfile – MTA Profile Name - Optional; An MTA Profile Indicator identifiable by the CMS.
        Timezone - within which this MTA is physically located. Optional; If present overrides the CMS default
        setting for the time zone.  As per RFC 1123 numeric timezone format.
        Protocol - Optional; If present it must be set to "MGCP 1.0 NCS 1.0". This is the default for all contained
        endpoints.
        Codec - Optional; If present it is the default for all contained endpoints.
        IPSecControl – Optional; NCS IPSec Control Flag (default = True; IPSec enabled).
    -->
    <xs:element name="PcspMta">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="MtaFqdn" type="nonEmptyString"/>
                <xs:element name="ListenPort" type="xs:int" minOccurs="0"/>
                <xs:element name="CmtsFqdn" type="xs:string"/>
                <xs:element name="MtaProfile" type="xs:string" minOccurs="0"/>
                <xs:element name="Timezone" type="timezoneType" minOccurs="0"/>
                <xs:element name="Protocol" type="protocolType" minOccurs="0"/>
                <xs:element name="Codec" type="codecType" minOccurs="0"/>
                <xs:element name="IPSecControl" type="xs:boolean" minOccurs="0"/>
                <xs:element ref="Extension" minOccurs="0"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <!--
        The PcspService object.

        Contents...

            ServiceId - unique identifier for the service.

            AdminStatus -
                0: suspended (i.e. bill not paid).
                1: enabled (normal state).
                2: number has changed.
                3: out of service.
                4: unassigned.

            BillingId - An telephone number identifying another service to be billed instead of this service.

            ExternalId - an arbitrary string used to carry such data as subscriber ID, etc.

            IsPrimary - With multiple services provisioned upon an endpoint, one service MUST
                        have this flag set to indicate  the default service to use for outgoing calls.
                false/0: this service is not a primary service.
                true/1: this service is a primary service.

            PrimaryRing - Primary Ringing Pattern ID. Index into MTA cadence table, selecting ring pattern for this
            service.  Optional if the "Is Primary" flag is set to False. If not present, the CMS must use its normal
            ring pattern

            DisplayName - Used for Call Name Delivery feature (CNAM)

            DisplayNumber - Used for Call Number Delivery feature (CND)

            Password - various call features require a password before any alterations are permitted.

            Network announcement control. See previous definition. Optional; if not present the CMS must use its
            default settings.

            Interexchange codes and Local Number Portability settings. See previous definitions. Optional; if not
            present the CMS must not assign any inter-exchange codes to the service.

            Call features.  See previous definitions.
    -->
    <xs:element name="PcspService">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="ServiceId" type="ServiceIdType"/>
                <xs:element name="AdminStatus">
                    <xs:simpleType>
                        <xs:restriction base="xs:integer">
                            <xs:enumeration value="0"/>
                            <xs:enumeration value="1"/>
                            <xs:enumeration value="2"/>
                            <xs:enumeration value="3"/>
                            <xs:enumeration value="4"/>
                        </xs:restriction>
                    </xs:simpleType>
                </xs:element>
                <xs:element name="BillingId" type="ServiceIdType"/>
                <xs:element name="ExternalId" type="xs:string"/>
                <xs:element name="IsPrimary" type="xs:boolean"/>
```

```
                <xs:element name="PrimaryRing" type="xs:string" minOccurs="0"/>
                <xs:element name="DisplayName" type="xs:string"/>
                <xs:element name="DisplayNumber" type="xs:string"/>
                <xs:element name="Password" type="xs:string"/>
                <xs:element ref="Announcements" minOccurs="0"/>
                <xs:element ref="InterExchange" minOccurs="0"/>
                <xs:element ref="LNP"/>
                <xs:element ref="ListOfCallFeatures"/>
                <xs:element ref="Extension" minOccurs="0"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <!--
        Import/Export file format.
        Used to transfer one or more objects and relations between PS/CMS.

        NOTE: PcspCms is not included. There is currently no reason for a CMS to obtain
        its own CMS object from the PS.
    -->
    <xs:element name="PcspImportExport">
        <xs:complexType>
            <xs:choice minOccurs="0" maxOccurs="unbounded">
                <xs:element ref="PcspService"/>
                <xs:element ref="PcspEndpoint"/>
                <xs:element ref="PcspMta"/>
                <xs:element ref="PcspRelation"/>
            </xs:choice>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

Appendix II - Sample Entity Encodings

II.1 PcspService Object Example

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
    Example Service object encoding.

    Default and "pcsp" namespace is set to PcspI01.
    "pcsp" namespace is a convenience, allowing vendor extensions
    to reference elements from the main PCSP schema.

-->
<PcspService xmlns="http://www.cablelabs.com/Pcsp/I01/schema"
xmlns:pcsp="http://www.cablelabs.com/Pcsp/I01/schema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:noNamespaceSchemaLocation="PcspI01.xsd">
    <!--
        A sample Service object.
    -->
    <ServiceId format="NSN">9785551212</ServiceId>
    <AdminStatus>1</AdminStatus>
    <BillingId>9785550000</BillingId>
    <ExternalId>0123456789</ExternalId>
    <IsPrimary>true</IsPrimary>
    <PrimaryRing>IndexIntoCadenceTable</PrimaryRing>
    <DisplayName>John Q Public</DisplayName>
    <DisplayNumber>(978)-555-1212</DisplayNumber>
    <Password>45hjg3j6gkg6h54j6gkj3g6</Password>
    <Announcements>
        <Language>EN</Language>
        <Timezone>+0500</Timezone>
    </Announcements>
    <InterExchange>
        <PIC>0123</PIC>
        <LPIC>0123</LPIC>
        <IPIC>0123</IPIC>
    </InterExchange>
    <LNP>
        <PortingStatus>0</PortingStatus>
        <LNPT>0</LNPT>
    </LNP>
    <ListOfCallFeatures>
        <CfCND>
            <Subscribed>true</Subscribed>
            <AdminStatus>1</AdminStatus>
        </CfCND>
        <CfCIDB>
            <Subscribed>0</Subscribed>
            <AdminStatus>1</AdminStatus>
            <UserParams>
                <Flag>PUBLIC</Flag>
            </UserParams>
        </CfCIDB>
        <CfCFBL>
            <Subscribed>true</Subscribed>
            <AdminStatus>1</AdminStatus>
            <UserParams>
                <Active>true</Active>
                <Always>0</Always>
                <ForwardTo>9785551212</ForwardTo>
            </UserParams>
        </CfCFBL>
```

```
<CfSPCALL>
    <Subscribed>0</Subscribed>
    <AdminStatus>1</AdminStatus>
    <UserParams>
        <ListOfSpeedDial>
            <SdPair>
                <SdNum>1</SdNum>
                <ServiceId>9785551212</ServiceId>
            </SdPair>
            <SdPair>
                <SdNum>3</SdNum>
                <ServiceId>9785551000</ServiceId>
            </SdPair>
        </ListOfSpeedDial>
    </UserParams>
</CfSPCALL>
<CfRDA>
    <Subscribed>1</Subscribed>
    <AdminStatus>1</AdminStatus>
</CfRDA>
<CfLSR>
    <Subscribed>1</Subscribed>
    <AdminStatus>1</AdminStatus>
    <UserParams>
        <BlkDomLongDist>1</BlkDomLongDist>
        <BlkIntLongDist>1</BlkIntLongDist>
        <BlkPayPerCall>1</BlkPayPerCall>
        <BlkOperatorAssist>1</BlkOperatorAssist>
        <BlkDirAssist>1</BlkDirAssist>
        <BlkTollFree>1</BlkTollFree>
        <ListOfServiceId>
            <ServiceId>9895551001</ServiceId>
            <ServiceId>9895551002</ServiceId>
            <ServiceId>9895551003</ServiceId>
        </ListOfServiceId>
    </UserParams>
</CfLSR>
<CfDND>
    <Subscribed>1</Subscribed>
    <AdminStatus>1</AdminStatus>
    <UserParams>
        <Active>true</Active>
        <WdStartTod1>00:00:00+05:00</WdStartTod1>
        <WdStopTod1>06:00:00+05:00</WdStopTod1>
        <WdStartTod2>18:00:00+05:00</WdStartTod2>
        <WdStopTod2>20:00:00+05:00</WdStopTod2>
        <WeStartTod1>00:00:00+05:00</WeStartTod1>
        <WeStopTod1>09:00:00+05:00</WeStopTod1>
        <WeStartTod2>18:00:00+05:00</WeStartTod2>
        <WeStopTod2>20:00:00+05:00</WeStopTod2>
    </UserParams>
</CfDND>
<CfCOC>
    <Subscribed>1</Subscribed>
    <AdminStatus>1</AdminStatus>
    <UserParams>
        <Active>true</Active>
        <StartTod>00:00:00+05:00</StartTod>
        <StopTod>06:00:00+05:00</StopTod>
        <ListOfServiceId>
            <ServiceId>9895551001</ServiceId>
            <ServiceId>9895551002</ServiceId>
            <ServiceId>9895551003</ServiceId>
        </ListOfServiceId>
    </UserParams>
</CfCOC>
```

```
        <CfNSA>
            <Subscribed>1</Subscribed>
            <AdminStatus>1</AdminStatus>
            <UserParams>
                <Active>true</Active>
                <StartTod>00:00:00+05:00</StartTod>
                <StopTod>06:00:00+05:00</StopTod>
                <ListOfServiceId>
                    <ServiceId>9895551001</ServiceId>
                    <ServiceId>9895551002</ServiceId>
                    <ServiceId>9895551003</ServiceId>
                </ListOfServiceId>
            </UserParams>
        </CfNSA>
    </ListOfCallFeatures>
</PcspService>
```

II.2PcspEndpoint Object Example

```
<?xml version="1.0" encoding="UTF-8"?>
<PcspEndpoint xmlns="http://www.cablelabs.com/Pcsp/I01/schema"
xmlns:pcsp="http://www.cablelabs.com/Pcsp/I01/schema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
    <!--
        A sample Endpoint object.
    -->
    <EndpointId>aaln/1@mta01.cablelabs.com</EndpointId>
    <AdminStatus>2</AdminStatus>
    <Protocol>MtaDefault</Protocol>
    <Codec>2</Codec>
    <IPSecControl>true</IPSecControl>
</PcspEndpoint>
```

II.3PcspMTA Object Example

```
<?xml version="1.0" encoding="UTF-8"?>
<PcspMta xmlns="http://www.cablelabs.com/Pcsp/I01/schema"
xmlns:pcsp="http://www.cablelabs.com/Pcsp/I01/schema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:noNamespaceSchemaLocation="PcspI01.xsd">
    <!--
        A sample MTA object.
    -->
    <MtaFqdn>mta01.cablelabs.com</MtaFqdn>
    <ListenPort>2427</ListenPort>
    <CmtsFqdn>cmta01.cablelabs.com</CmtsFqdn>
    <Timezone>-0500</Timezone>
    <Protocol>MCGP 1.0 NCS 1.0</Protocol>
    <Codec>5</Codec>
    <IPSecControl>true</IPSecControl>

</PcspMta>
```

II.4PcspCMS Object Example

```
<?xml version="1.0" encoding="UTF-8"?>
<PcspCms xmlns="http://www.cablelabs.com/Pcsp/I01/schema"
xmlns:pcsp="http://www.cablelabs.com/Pcsp/I01/schema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
    <!--
        CMS object.

        Not much defined yet...just its key.
        Serves as a collection for Services and Endpoints.
    -->
    <CmsFqdn>cma01.cablelabs.com</CmsFqdn>
</PcspCms>
```

II.5PcspRelation Example

```
<?xml version="1.0" encoding="UTF-8"?>
<PcspRelation xmlns="http://www.cablelabs.com/Pcsp/I01/schema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.cablelabs.com/Pcsp/I01/schema PcspI01.xsd" relOp="add">
    <!--
        A PcspRelation.

        This relation associates several Endpoints to the Service "9785551212".
    -->
    <Class1>PcspService</Class1>
    <Key>9785551212</Key>
    <Class2>PcspEndpoint</Class2>
    <ListOfKeys>
        <Key>aaln/1@mta01.cablelabs.com</Key>
        <Key>aaln/1@mta02.cablelabs.com</Key>
        <Key>aaln/1@mta03.cablelabs.com</Key>
        <Key>aaln/1@mta04.cablelabs.com</Key>
    </ListOfKeys>
</PcspRelation>
```

Appendix III - Sample Object Extension

III.1 Extended PcspService Object Example

The following example illustrates the extension capabilities of the PCSP schema. The example extends a PcspService object with a new call feature and several new elements on the main body of the object.

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
    An example illustrating how to extend a Pcsp object.
    This example extends  the PcspService object with additional
    fields and call features.

    See details below.
-->
<PcspService xmlns="http://www.cablelabs.com/Pcsp/I01/schema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:pcsp="http://www.cablelabs.com/Pcsp/I01/schema">
    <!--
        The main body of the Service object is filled with sample data that will
        allow the object to validate.
    -->
    <ServiceId>5551212</ServiceId>
    <AdminStatus>0</AdminStatus>
    <BillingId>5551212</BillingId>
    <ExternalId>5551212</ExternalId>
    <IsPrimary>true</IsPrimary>
    <PrimaryRing/>
    <DisplayName/>
    <DisplayNumber/>
    <Password/>
    <Announcements>
        <Language>EN</Language>
        <Timezone>+0500</Timezone>
    </Announcements>
    <InterExchange>
        <PIC>0</PIC>
        <LPIC>0</LPIC>
        <IPIC>0</IPIC>
    </InterExchange>
    <LNP>
        <PortingStatus>1</PortingStatus>
        <LNPT>true</LNPT>
    </LNP>
    <!--
        A Service object can be extended in two locations:
            1. The main body of the object.
            2. The call feature list.

        Here we extend the set of call features with the CfXYZ call feature.

        1. The VendorExt element must specify a valid namespace for the extension's schema. This allows
            the parsing system to locate the schema file for the extension.
        2. Any content within the VendorExt element must be namespace qualified, enabling validation against
            the extension's schema.
    -->
    <ListOfCallFeatures>
        <Extension xmlns:ext="http://www.cablelabs.com/SampleExtension">
            <ext:CfXYZ>
                <ext:Subscribed>true</ext:Subscribed>
                <ext:Enabled>true</ext:Enabled>
            </ext:CfXYZ>
        </Extension>
    </ListOfCallFeatures>
    <!--
        Here, we extend the data content of main body of the Service object.
    -->
    <Extension xmlns:ext="http://www.cablelabs.com/SampleExtension">
        <ext:A>Sample extension A</ext:A>
        <ext:B>Sample extension B</ext:B>
        <ext:C>Sample extension C</ext:C>
    </Extension>
</PcspService>
```

III.2 The Extension Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
    The schema for the sample PcspService extension.

    This schema defines several extensions:

    A, B, and C for the main body of the Service object.
    Call feature CfXYZ for the Service's call feature list.
-->
<xs:schema targetNamespace="http://www.cablelabs.com/SampleExtension"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns="http://www.cablelabs.com/SampleExtension"
elementFormDefault="qualified">
    <xs:element name="A" type="xs:string"/>
    <xs:element name="B" type="xs:string"/>
    <xs:element name="C" type="xs:string"/>
    <xs:element name="CfXYZ">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="Subscribed" type="xs:boolean"/>
                <xs:element name="Enabled" type="xs:boolean"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

Appendix IV WSDL Specification For PCSP Messaging.

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
    The PacketCable CMS Subscriber Provisioning interface.
    Specified in Web Service Description Language 1.1.
-->
<definitions name="PcspI01Service" targetNamespace="http://www.packetcable.com/pcsp/i01"
xmlns:tns="http://www.packetcable.com/pcsp/i01" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns="http://schemas.xmlsoap.org/wsdl/">
    <!--
        The <types> section defines custom datatypes required by the interface.

        PCSPI01 requires two custom datatypes:
            PcspArg (and array of)
            PcspObj (and array of).

        // PcspArg (pseudo code)
        //
        class PcspArg
        {
            // EntityName and key of a specific object.
            // Wildcard are currently not permitted.
            // Key is ignored when entity is PcspRelation.
            //
            String entityName;
            String key;

            // Reserved for future use.  Set to 0 for now.
            //
            int flags;
        }

        // PcspObj (pseudo code).
        //
        class PcspObj
        {
            // EntityName and key of the specific object.
            // Key is ignored when entity is PcspRelation.
            //
            String entityName;
            String key;

            // cmdStatus:
            //    PcspObj as method output/result - MUST be set to one of the status codes specified below.
            //    PcspObj as input to Put() -  MUST be set to one of the following:
            //         1, create new object
            //         2, modify existing object.
            //  This field is ignored when entity is PcspRelation.
            //
            int cmdStatus;

            // XML encoding per PCSP Data Model Schema or 0 (null)
            //
            String xmlEncoding;
        }

        EntityNames; MUST be one of the following:

            "PcspService"
            "PcspMta"
            "PcspEndpoint"
            "PcspCms"
            "PcspRelation"

        Status codes:  Used for method output or contained in the cmdStatus field of a PcspObj result (output).

            0 , Operation succeeded
            1 , Object not found
            2 , Invalid Put() mode specified.
            3 , Object creation failed, object already exists
            4 , Read op failed
            5 , Create op failed
            6 , Modify op failed
            7 , Delete op failed
            8 , Internal problem.
            9 , Server Busy
            10, Unsupported operation.
            11, Vendor extension.

        ...extended as needed...
    -->
    <types>
        <schema xmlns="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.packetcable.com/pcsp/i01">
            <complexType name="PcspObj">
                <sequence>
                    <element name="entityName" type="string"/>
                    <element name="key" type="string"/>
                    <element name="cmdStatus" type="int"/>
                    <element name="xmlEncoding"  type="string"/>
                </sequence>
            </complexType>
            <complexType name="ArrayOfPcspObj">
                <complexContent>
                    <restriction base="soapenc:Array">
                        <attribute ref="soapenc:arrayType" wsdl:arrayType="tns:PcspObj[]"/>
                    </restriction>
                </complexContent>
            </complexType>
            <complexType name="PcspArg">
                <sequence>
                    <element name="entityName" type="string"/>
                    <element name="key" type="string"/>
                    <element name="flags" type="int"/>
                </sequence>
            </complexType>
            <complexType name="ArrayOfPcspArg">
                <complexContent>
                    <restriction base="soapenc:Array">
                        <attribute ref="soapenc:arrayType" wsdl:arrayType="tns:PcspArg[]"/>
                    </restriction>
                </complexContent>
            </complexType>
        </schema>
    </types>
    <!--
        Message section.

        Invoking a method on the interface involves two "messages"...an input message and an output message.
        "In" contains the set of input args to the method call.
        "Out" contains the return values.
    -->
    <message name="Get0In">
        <part name="args" type="tns:ArrayOfPcspArg"/>
    </message>
    <message name="Get0Out">
        <part name="Result" type="tns:ArrayOfPcspObj"/>
    </message>
```

```
<message name="Put1In">
    <part name="objs" type="tns:ArrayOfPcspObj"/>
</message>
<message name="Put1Out">
    <part name="Result" type="tns:ArrayOfPcspObj"/>
</message>
<message name="Delete2In">
    <part name="args" type="tns:ArrayOfPcspArg"/>
</message>
<message name="Delete2Out">
    <part name="Result" type="tns:ArrayOfPcspObj"/>
</message>
<message name="CmdStatus3In">
    <part name="isCmd" type="xsd:boolean"/>
    <part name="code" type="xsd:int"/>
    <part name="subCode" type="xsd:int"/>
    <part name="vendorExtension" type="xsd:string"/>
</message>
<message name="CmdStatus3Out">
    <part name="Result" type="xsd:int"/>
</message>
<!--
    Port type defines the interface.

    Each "operation" is a method on the interface, with associated input and output messages
    (args and return values).

    // The PCSP service interface (in pseudo code).
    //
    interface IPcspI01Service
    {
        // Get (read) one or more objects from the server.
        // EntityName of "PcspRelation' it not allowed (objects only)
        //
        PcspObj[] Get(PcspArg[] args);

        // Put (write) objects and relations to the server.
        //
        PcspObj[] Put(PcspObj[] objs);

        // Delete objects and relations from the server.
        //
        PcspObj[] Delete(PcspArg[] args);

        // Out-of-band command and status reporting.
        //
        // Predefined command codes:
        //    0 - extension command
        //
        // Predefined status codes:
        //    0 - extension status
        //
        int CmdStatust(boolean cmd,          // true for CMD, false for STATUS.
                                int code,         // CMD or STATUS code (see above).
                                int subCode          // SubCode.  Further refines code.
                                String extension);
}
-->
<portType name="PcspI01Service">
    <operation name="Get" parameterOrder="args">
        <input name="Get0In" message="tns:Get0In"/>
        <output name="Get0Out" message="tns:Get0Out"/>
    </operation>
    <operation name="Put" parameterOrder="objs">
        <input name="Put1In" message="tns:Put1In"/>
        <output name="Put1Out" message="tns:Put1Out"/>
        </operation>
        <operation name="Delete" parameterOrder="args">
            <input name="Delete2In" message="tns:Delete2In"/>
            <output name="Delete2Out" message="tns:Delete2Out"/>
        </operation>
        <operation name="CmdStatus" parameterOrder="isCmd code subCode vendorExtension">
            <input name="CmdStatus3In" message="tns:CmdStatus3In"/>
            <output name="CmdStatus3Out" message="tns:CmdStatus3Out"/>
        </operation>
    </portType>
    <!--
        Bind the interface ("portType") to transport specifics.
        Essentially, each method's input and output flow is bound as a
        remote procedure call using SOAP 1.1.
    -->
    <binding name="PcspI01Service" type="tns:PcspI01Service">
        <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
        <operation name="Get">
            <soap:operation soapAction="Get" style="rpc"/>
            <input name="Get0In">
                <soap:body use="encoded" namespace="http://www.packetcable.com/pcsp/i01"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
            </input>
            <output name="Get0Out">
                <soap:body use="encoded" namespace="http://www.packetcable.com/pcsp/i01"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
            </output>
        </operation>
        <operation name="Put">
            <soap:operation soapAction="Put" style="rpc"/>
            <input name="Put1In">
                <soap:body use="encoded" namespace="http://www.packetcable.com/pcsp/i01"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
            </input>
            <output name="Put1Out">
                <soap:body use="encoded" namespace="http://www.packetcable.com/pcsp/i01"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
            </output>
        </operation>
        <operation name="Delete">
            <soap:operation soapAction="Delete" style="rpc"/>
            <input name="Delete2In">
                <soap:body use="encoded" namespace="http://www.packetcable.com/pcsp/i01"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
            </input>
            <output name="Delete2Out">
                <soap:body use="encoded" namespace="http://www.packetcable.com/pcsp/i01"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
            </output>
        </operation>
        <operation name="CmdStatus">
            <soap:operation soapAction="CmdStatus" style="rpc"/>
            <input name="CmdStatus3In">
                <soap:body use="encoded" namespace="http://www.packetcable.com/pcsp/i01"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
            </input>
            <output name="CmdStatus3Out">
                <soap:body use="encoded" namespace="http://www.packetcable.com/pcsp/i01"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
            </output>
        </operation>
    </binding>
    <!--
        The top level definition of the PCSP I01 Service.

        Note that the <service> element does not contain an address. It is assumed that the actual address
        of the service will be set explicitly within the client and server.
    -->
    <service name="PcspI01Service">
        <documentation>PacketCable CMS Subscriber Provisioning Service I01</documentation>
        <port name="PcspI01Service" binding="tns:PcspI01Service">
            <soap:address location=""/>
        </port>
    </service>
</definitions>
```

Appendix V - Data Encoding Evaluation

Options considered for the encoding of data objects and messaging:

V.1 XML

XML is a standard meta-language that allows organizations to design their own markup languages for document publishing and data exchange. Such markups are text based: designed to be obvious to both people and processes. XML offers…

- Open, standards based, platform independent data exchange.
- Standardized parsers for putting data into memory.
- Standardized interfaces (tree-oriented and stream-oriented) for processing the data.
- Standardized ways to display data.
- Standardized ways to query data.
- Standardized ways to link data.
- Standardized training of people in both publishing and data processing.

The cost: somewhat larger encoding size and increased parsing overhead.

The XML specification is supervised by the XML Working Group of the World Wide Web Consortium (W3C). Special Interest Groups of experts from various fields contribute. It is a public standard —it is not the proprietary development of any company. The v1.0 specification was accepted by the W3C as Recommendation on Feb 10, 1998. The specification may be found at http://www.w3.org/TR/REC-xml.

V.2 ASN.1/BER

ASN.1 is a text based, platform independent syntax used to represent arbitrary data structures. Is it popularly employed for SNMP MIB representations. The Basic Encoding Rules is a simply recursive algorithm that produces a compact octet encoding from an ASN.1 description. BER encodes each item as a tag, indicating what type it is, a length indicating the size of the object, and a value, which contains the actual contents of the object.

See Table 2 at end of this section.

V.3 Proprietary ASCII

Proprietary encodings are out of scope.

V.4 SDP (session description protocol)

Not flexible in term of contents. Used primarily to describe streaming media capabilities.

V.5 RADIUS

Radius data encoding (TLV) is too primitive and cannot enforce sequencing. RADIUS is difficult and limited to code structures.

V.6 SQL

Tied to a specific relational database implementation/schema. Some vendors may already have databases deployed with incompatible schema.

V.7 Options Summary

Table 2. Data Encoding Options

Option	Pro	Con
XML	Flexible, ASCII tag-based Provides syntax checking through use of schema/DTD. Easy to extend without effecting transport. Allows for vendor extensions. Platform-independent Language-independent	Not secure, requires a secure transport layer. Will consume more CPU and network capacity than a binary encoding. Parse time, etc.
ASN.1/BER	Hierarchical structure for formatting of data. Defines a language for describing the data format (or "schema"). Data structures can be nested. Data is formatted in a platform independent way. Format can be extended IF the design includes a way of versioning the format so that the application knows which format it needs to use to parse the data content.	ASN.1 isn't easily extensible. Backward compatibility of format versions can be difficult to incorporate into the design and to implement. Most implementations use compiled binary level parsers for each schema, which means that defining flexible applications becomes quite difficult. Debugging applications and their interoperability can be difficult in that a small formatting error can render a data "packet" unparseable / unreadable.
Proprietary ASCII	Out of scope	Out of scope
SDP (Session Description)		Not Flexible in term of contents. Used primarily to describe streaming media capabilities.
RADIUS		Radius data encoding (TLV) is too primitive. Cannot enforce sequencing. Difficult and limited to code structures.
SQL		Tied to relational database schema implementation - which some vendors may not use.

V.8 Recommendation: XML

XML provides a platform-agnostic, technology-neutral form of structuring messages and packaging data. It is an excellent choice to send data between heterogeneous applications without each application having to know about the proprietary format of the other. Because XML is a structured language, it is a good fit for hierarchical types of messages. Data can be easily mapped to elements, so the XML document (as a tree structure) takes care of the hierarchy maintenance. The costs: increased wire payload sizes and increased marshalling times (parsing) for objects.

Appendix VI - Transport Protocol Evaluation

VI.1 TFTP with IPSEC

TFTP is already in use within the PacketCable infrastructure (DOCSIS). It is intended as a lightweight file transfer protocol.

VI.2 Batched RADIUS - multiple records in single request via event msgs.

RADIUS is an IETF standard created primarily to handle Internet dial-up authentication, authorization, and accounting. RADIUS is currently the de facto standard used by most router manufacturers for such activities. Several vendors of IP telephony gateway equipment are already utilizing RADIUS' support for vendor extensions to deliver the information needed for billing.

RADIUS defines both a transport protocol and a specification for message formats. As a transport protocol, RADIUS relies on user datagram protocol (UDP) for message broadcast and is port-based.

As a message format, the data is formatted based on tag-length-value (also called attribute-length-value). Standard authentication, authorization, and accounting tags are pre-defined and are minimally required. However, new attributes can be added without disturbing existing implementations of the protocol. RADIUS has a minimum total message length of 20 characters and a maximum length of 4096 characters. Individual data fields support 247 bytes of data, for example, a 247 character URL or filename.

RADIUS has very poor reliability characteristics and essentially non-existent error recovery, is very limited in new tags (can only define a total of 255) (compare that to 600 existing features in some PSTN class 5 switches)

For detailed information on RADIUS refer to:

http://info.internet.isi.edu/in-notes/rfc/files/rfc2139.txt
http://www.livingston.com/marketing/whitepapers/RADIUS_paper.html.

VI.3 Diameter

Diameter represents an activity in the working group of the IETF that is designed to be backward compatible to RADIUS. It is much more extensible, has increased security benefits, and is designed to minimize configuration. In addition, it supports cross-domain AAA very well by supporting a variety of security schemes such as public key, etc. Diameter supports fail-over to a backup server (it is designed for environments that have low failure requirements (99.99+)). It will be proposed as an RFC at the IETF meeting held in Oslo from July 12th through the 16th.

RADIUS/DIAMETER do not provide two-way communication (only acknowledgments) therefore it does not fulfill the requirements.

VI.4 Distributed Object Systems

VI.4.1 CORBA/IIOP

The distributed object technology championed by the Object Management Group. There are upwards of 800 OMG members behind this technology.

The Common Object Request Broker Architecture (CORBA) allows applications to communicate with one another by using an Object Request Broker (ORB). The ORB is middleware that establishes the client-server relationships between objects. The ORB registers clients and manages rights such as publish, subscribe and listener. Using an ORB, a client can transparently invoke a method on a server object, which can be on the same machine or across a network. The ORB intercepts the local call and is responsible for finding an object that can implement the request, pass it the parameters, invoke its method, and return the results.

The Interface Definition Language (IDL) is used to establish the ORB protocol contract between client and server objects. The ORB essentially hides the transportation details from the programmer. The IDL is complied into C++, Java, etc. implementations of client and server stubs, handling all data encoding/decoding chores required by the IIOP transport protocol used between clients and servers.

CORBA will handle the details of finding the server for a method call, transporting arguments from the client machine to the server machine, and transporting any return code back to the client machine.

ORBs are currently available from many vendors for more than three dozen hardware platforms and operating systems. CORBA is particularly popular on *nix platforms. However, in practice, ORB vendors compete on features. Persistent interoperability problems exist when one gets past the basics (security, etc.). The likelihood of two randomly selected ORBs being able to successfully communicate is low. From a development standpoint, CORBA tends to be very complex. Additionally, CORBA is a relatively expensive option (runtime and development licenses).

VI.5 DCOM

Microsoft's Distributed Component Object Model. Shares the following characteristics with CORBA:

- Separates object interface from implementation. This is accomplished using MIDL (Microsoft's IDL variant).
- Allows transparency of location. Clients invoke methods on remote objects without knowing which machine the remote object runs upon.
- Uniform exception handling scheme (DCOM method calls return a flat HRESULT return status)

However…

- DCOM is based on DCE ORPC transport protocol, which in incompatible with IIOP transport used by CORBA.
- DCOM is basically a Microsoft only technology. It is standard on Win95, Win98 and NT platforms.

VI.6 HTTP

Hypertext Transfer Protocol (HTTP) is the primary protocol for the World Wide Web. HTTP was designed to connect heterogeneous data sources together to create a distributed information system. It was also designed with extensibility in mind. A typical HTTP transaction:

1. Client established connection to the server

2. Client issues a request to the server (with URL parameters)

3. Server sends response containing status and requested URL

4. Either side can disconnect.

HTTP provides transaction headers for both client requests and server responses. The client transaction header can include parameters used to assist delivery of the desired information to the client (e.g., type of data format, language etc). The server transaction header can include parameters indicating information about the response (e.g., the status of the request (return code), the length of the data being sent, the content type, the language of the content, etc.).

Given the current state of the Web, HTTP is ubiquitous. It is also firewall friendly.

VI.7 Options Summary

Table 3. Transport Options

Option	Pro	Con
TFTP with IPSEC	Lightweight Already implemented for DOCCSIS	Doesn't provide two way communication
RADIUS	Flexible – includes vendor-specific and customer-specific fields. PacketCable may be able to define fields in this space. Used by many IP telephony accounting systems Already widely deployed on IP components such as routers.	Not all RADIUS products support AAA Application layer needs to handle reliability issues Inadequate built-in security, need an independent trust protocol or shared secret keys with edge routers
Socket based proprietary protocol with SSL		Proprietary. If pursued, we will probably end up writing most of what is currently available in SOAP or XMLP.
CORBA/IIOP	Easy to implement, details of name resolution, packaging parameters into messages and transport are all managed by the CORBA infrastructure	Tends to be a more expensive solution. The CORBA infrastructure must either be developed or purchased, and in either case it must be deployed with the application. There may still remain issues re: interoperability or various CORBA/ORB products.
DCOM	DCOM and CORBA/IIOP are similar technologies.	Basically a Microsoft Windows only option. Won't inter-operate with CORBA-IIOP.
HTTP	HTTP already widely deployed as an underlying transport protocol for exchange between data sources (web servers) and data consumers (client browsers). Data transmission based on simple transactions. Simple, stateless, ASCII text based protocol. Allows for easy data exchange through most currently deployed network infrastructure (firewalls, etc.). Client can use simple method calls when making requests as - GET, POST, HEAD, PUT, DELETE, LINK, and UNLINK.	Being Stateless, the protocol has no memory of the transaction once it finishes. Can it keep up with required CMS transaction rate ? Relatively expensive in terms of bandwidth and processing requirements.

VI.8 Recommendation: HTTP 1.1.

Appendix VII 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:

Eugene Nechamkin (Broadcom Corp.)
Geoff Devine (Cedar Point Communications)
Paul Duffy (Cisco Systems, Inc.) – Lead Author
Klaus Hermanns (Cisco Systems, Inc.)
Shengyou Zeng (Cisco Systems, Inc.)
Austin Platt (Jacobs Rimell)
Kerbey Altmann, (Sigma Systems, Inc.)
Peter Bates (Telcordia Technologies, Inc.)
Chad Stalsworth (Syndeo Corp.)

Sumanth Channabasappa, CableLabs, Inc.