International Telecommunication Union

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

J.177
(11/2005)

SERIES J: CABLE NETWORKS AND TRANSMISSION OF TELEVISION, SOUND PROGRAMME AND OTHER MULTIMEDIA SIGNALS

IPCablecom

IPCablecom CMS subscriber provisioning specification

ITU-T Recommendation J.177

ITU-T Recommendation J.177

IPCablecom CMS subscriber provisioning specification

Summary

This Recommendation defines the interface used between the Call Management Server (CMS) and Provisioning Server (PS) for the exchange of service provisioning information. The interface employs a Web Service model. Specified in Web Service Description Language 1.1 (WSDL 1.1), the interface transports XML encoded objects within Simple Object Access Protocol 1.1 (SOAP 1.1) encoded messages over an HTTP 1.1 transport. This interface is secured via IPSec.

Source

ITU-T Recommendation J.177 was approved on 29 November 2005 by ITU-T Study Group 9 (2005-2008) under the ITU-T Recommendation A.8 procedure.

CONTENTS

ITU-T Recommendation J.177

IPCablecom CMS subscriber provisioning specification

1 Scope

1.1 Purpose of this Recommendation

This Recommendation defines the interface used between the Call Management Server (CMS) and Provisioning Server (PS) for the exchange of service provisioning information. The interface employs a Web Service model. Specified in Web Service Description Language 1.1 (WSDL 1.1), the interface transports XML encoded objects within Simple Object Access Protocol 1.1 (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 IPCablecom features and specific vendor extensions.

1.2 Scope of this Recommendation

The scope of this Recommendation is limited to the provisioning of an IPCablecom 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 back-office 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.

2 References

2.1 Normative references

The following ITU-T Recommendations and other references contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published. The reference to a document within this Recommendation does not give it, as a stand-alone document, the status of a Recommendation.

- ITU-T Recommendation J.161 (2001), *Audio codec requirements for the provision of bidirectional audio service over cable television networks using cable modems.*
- ITU-T Recommendation J.162 (2005), *Network call signalling protocol for the delivery of time-critical services over cable television networks using cable modems.*
- ITU-T Recommendation J.170 (2005), *IPCablecom security specification*.
- IETF RFC 1123 (1989), *Requirements for Internet Hosts – Application and Support.*
- XML Protocol. http://www.w3.org/2000/xp.

2.2 Informative References

- ITU-T Recommendation J.160 (2005), *Architectural framework for the delivery of time-critical services over cable television networks using cable modems.*
- ITU-T Recommendation J.167 (2005), *Media terminal adapter (MTA) device provisioning requirements for the delivery of real-time services over cable television networks using cable modems.*
- IETF RFC 3588 (2003), *Diameter Base Protocol.*
- Simple Object Access Protocol. http://www.w3.org/TR/SOAP.
- Web Services Description Language. http://www.w3.org/TR/wsdl.

2.3 Reference acquisition

IETF RFCs:

- 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

This Recommendation defines the following terms:

3.1 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.

3.2 endpoint: A Terminal, Gateway or Multipoint Conference Unit.

3.3 Internet key exchange: A key-management mechanism used to negotiate and derive keys for SAs in IPSec.

3.4 local number portability: Allows a customer to retain the same number when switching from one local service provider to another.

3.5 media gateway: Provides the bearer circuit interfaces to the PSTN and transcodes the media stream.

3.6 pre-shared key: A shared secret key passed to both parties in a communication flow, using an unspecified manual or out-of-band mechanism.

3.7 registration, admissions and status: RAS Channel is an unreliable channel used to convey the RAS messages and bandwidth changes between two H.323 entities.

4 Abbreviations, acronyms and conventions

4.1 Abbreviations and acronyms

This Recommendation uses the following abbreviations:

AAA	Authentication, Authorization and Accounting
BPP	Basic POTS Provisioning
CFP	Call Feature Provisioning
CID	Circuit ID
CM	DOCSIS Cable Modem
CMS	Cryptographic Message Syntax

CMS	Call Management Server
CMTS	Cable Modem Termination System
Codec	COder-DECoder
CSR	Customer Service Representative
DOCSIS™	Data-Over-Cable Service Interface Specifications
DQoS	Dynamic Quality of Service
ESP	IPSec Encapsulating Security Payload
FQDN	Fully Qualified Domain Name
HTTP	Hypertext Transfer Protocol
IC	Inter-exchange Carrier
IETF	Internet Engineering Task Force
IKE	Internet Key Exchange
IP	Internet Protocol
IPSec	Internet Protocol Security
ITU	International Telecommunication Union
ITU-T	International Telecommunication Union – Telecommunication Standardization Sector
LNP	Local Number Portability
MC	Multipoint Controller
MG	Media Gateway
MGC	Media Gateway Controller
MGCP	Media Gateway Control Protocol
MIB	Management Information Base
MTA	Multimedia Terminal Adapter
NCS	Network Call Signalling
OSS	Operations Support System
PCM	Pulse Code Modulation
PCSP	IPCablecom CMS Subscriber Provisioning
PS	Provisioning Server
PSTN	Public Switched Telephone Network
QCIF	Quarter Common Intermediate Format
RAS	Registration, Admissions and Status
RFC	Request for Comments
SDP	Session Description Protocol
SNMP	Simple Network Management Protocol
SOAP	Simple Object Access Protocol
STP	Signalling Transfer Point

TD	Timeout for Disconnect
TFTP	Trivial File Transfer Protocol
TLV	Type-Length-Value
UDP	User Datagram Protocol
VoIP	Voice over Internet Protocol

4.2 Conventions

Throughout this Recommendation, 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 Recommendation.
"MUST NOT"	This phrase means that the item is an absolute prohibition of this Recommendation.
"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 behaviour is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behaviour 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.

5 Background

5.1 Service goals

Text deleted.

5.2 IPCablecom reference architecture

Figure 1 shows the reference architecture for the IPCablecom Network. Refer to the IPCablecom Architecture Recommendation, ITU-T Rec. J.160, for more detailed information on this reference architecture.

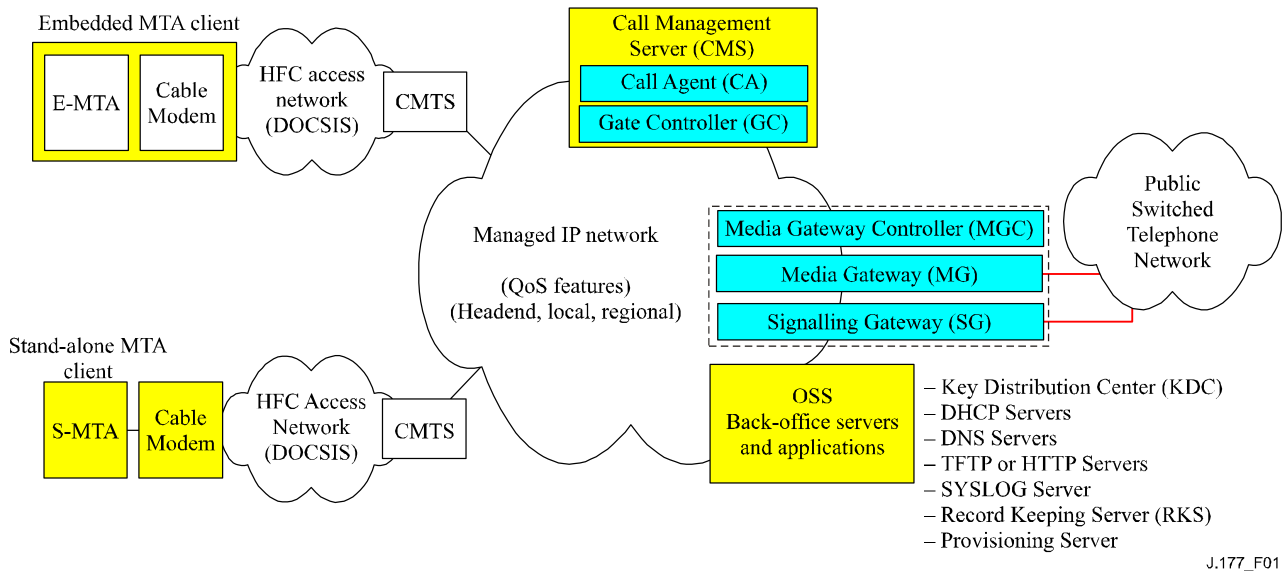

Figure 1/J.177 – IPCablecom 1.0 network component reference model (partial)

5.3 Components and interfaces

Provisioning is defined as the operations necessary to provide a specified service to a customer. IPCablecom service provisioning can be viewed as two distinct operations: MTA provisioning and CMS subscriber provisioning. Figure 2 presents the provisioning-related interfaces maintained by the Provisioning Server (PS) and other authorized back-office components to various IPCablecom elements. Interfaces not explicitly labelled are undefined and out of scope for IPCablecom.

This Recommendation 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).

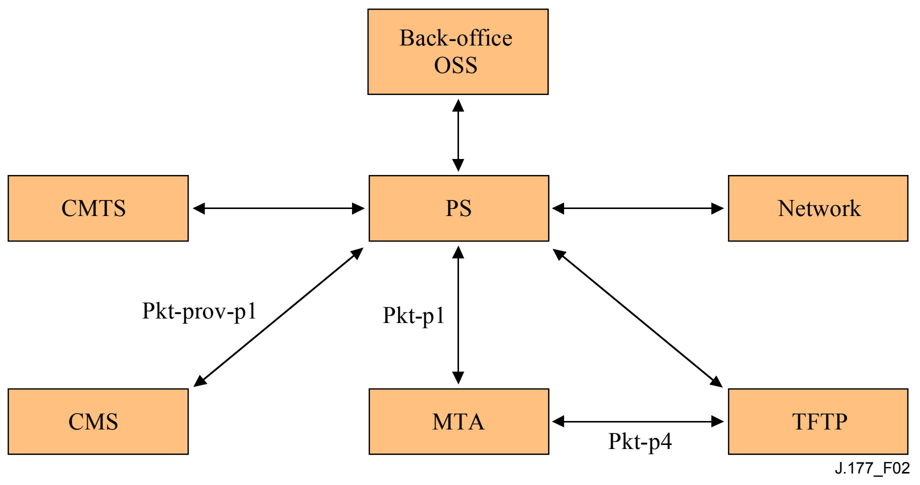

Figure 2/J.177 – Provisioning component interfaces

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 IPCablecom 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 IPCablecom elements. IPCablecom 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 ITU-T Rec. J.167 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 ITU-T Rec. J.160. This component provides call control and signalling-related services for the MTA and CMTS components in the IPCablecom network.

5.4.4 MTA

A Media Terminal Adapter is an IPCablecom client device that contains a subscriber-side interface to the customer's CPE (e.g., telephone) and a network-side signalling interface to call control elements in the network. This component is described in ITU-T Rec. J.160.

5.4.5 TFTP

A configuration file service that is the basis for most device configuration in an IPCablecom network. This could be a stand-alone 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 ITU-T Rec. J.167.

5.5.2 Pkt-p4

This interface is defined in ITU-T Rec. J.167.

5.5.3 Pkt-prov-p1

The interface defined in this Recommendation.

6 Assumptions

- The back-office components are 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 IPCablecom Security Recommendation, ITU-T Rec. J.170.

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 services and bills to, 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 IPCablecom.

7.2 Equipment set-up 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 set-up affects two major components in an IPCablecom environment.

- Customer premises equipment. For IPCablecom, this is the MTA. The provisioning for the MTA is defined in ITU-T Rec. J.167 and is not discussed in this Recommendation.
- 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 IPCablecom 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 set up translation tables enabling the CMS to route calls to the appropriate device/port, given a specific telephone number. BPP for each customer is required before that customer can receive any calls in an IPCablecom 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 from 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 IPCablecom 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 IPCablecom call feature.
- The interface's data model MUST be extensible.

  The present focus of the interface is telephony data. However, to the greatest extent possible, the interface SHOULD be extensible for future IPCablecom multimedia services. It is desirable to have a single, extensible provisioning data model and transport to support all IPCablecom 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 being taken out of service MUST be carefully documented.
- The interface MUST be capable of accommodating present (NCS) and future signalling 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 clause provides a high-level description of the PCSP data model and its XML encoding. It is intended as descriptive and non-authoritative. The normative, authoritative definition of the data model and its encoding is found in the PCSP XML schema in Annex A.

9.1 Overview

The data model for IPCablecom CMS provisioning is displayed in Figure 3. It consists of two categories of entities:

- Objects;
- Relations between objects.

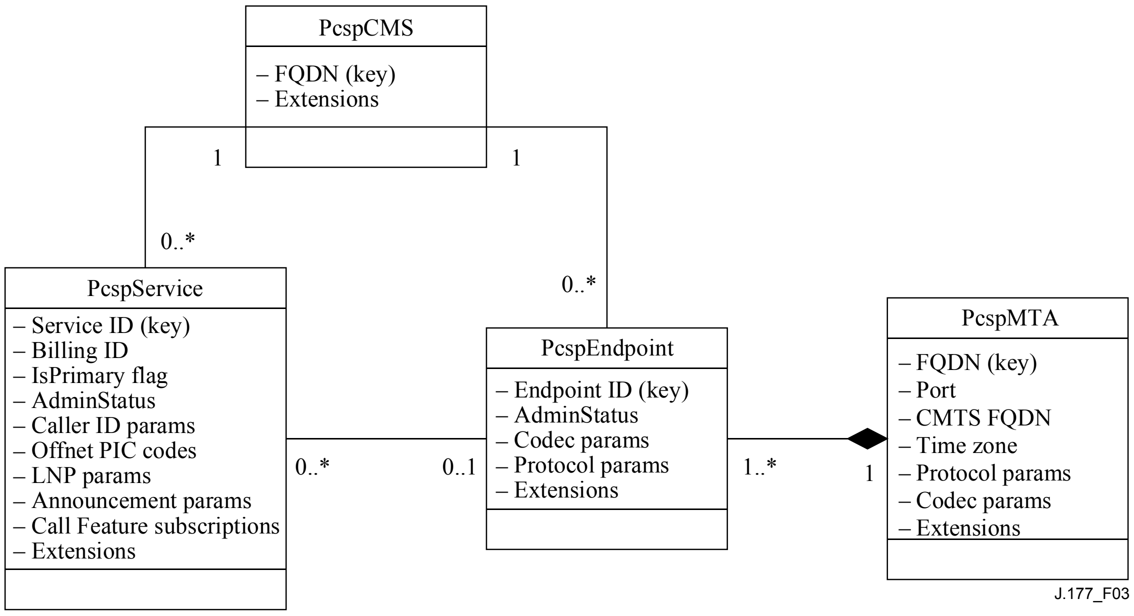

Figure 3/J.177 – CMS provisioning data model

The following entities MUST be supported:

- The PcspService object is the entity to which an IPCablecom 1.0 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 3, 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 endpoint 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 an IPCablecom 1.0 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;
- Signalling 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 Signalling between the CMS and the MTA. By default, IPSec is turned on for 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 PcspEndpoint 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 PcspCms 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 3, the lines connecting the classes represent object "relations" (sometimes called associations). The relations depicted in Figure 3 MUST be supported:

- Service/CMS – A typical CMS will own a block of phone numbers;
- Endpoint/CMS – An endpoint requires a CMS for signalling 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

Annex A 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 tear down 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 I.

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 II 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 IV.

The interface is modelled 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 (only objects are supported – 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 IPCablecom CMS provisioning, the CMS and PS assume the role requirements specified in Table 1.

Table 1/J.177 – 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 behaviour 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 4 displays all the required messaging roles.

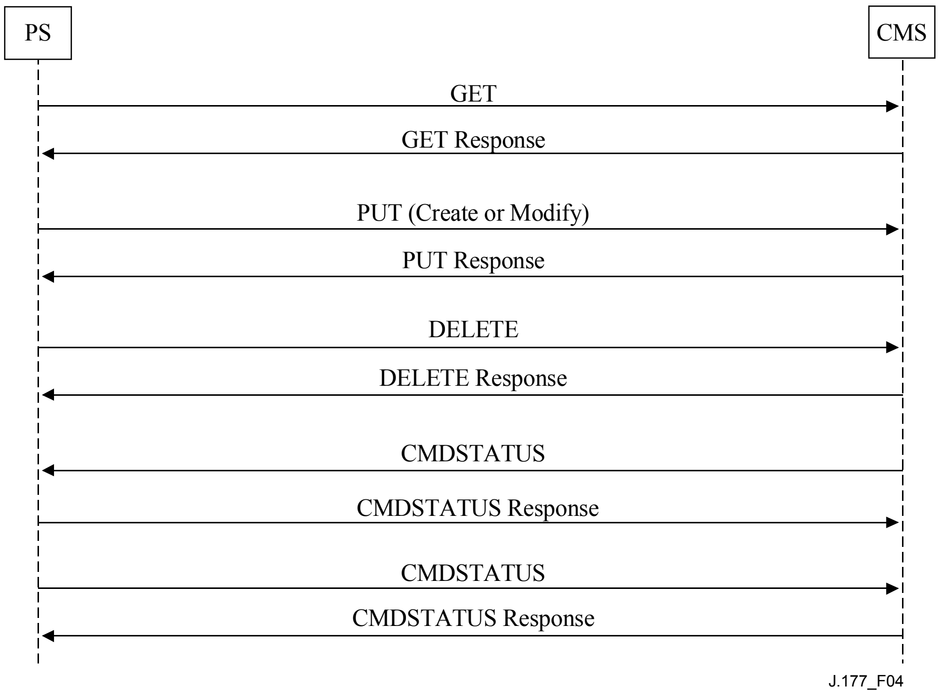

Figure 4/J.177 – Required messaging flows

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 accept, 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 the WSDL definition presented in Annex B.

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 ITU-T Rec. J.170 for full details.

Annex A

PCSP XML schema

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
   IPCablecom 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
      IPCablecom™ Audio/Video Codecs Specification J.161.
      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" 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="CfACRestrict">
   <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="CfAC-R">
   <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="CfACRestrict" minOccurs="0"/>
            <xs:element ref="CfAC-R" 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
         "IPCablecom 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. 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 – A 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.

         Inter-exchange 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>
```

Annex B

WSDL specification for PCSP messaging

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
   The IPCablecom CMS Subscriber Provisioning interface.
   Specified in Web Service Description Language 1.1.
-->
<definitions name="PcspI01Service" targetNamespace="unique_fully_qualified_namespace"
xmlns:tns="unique_fully_qualified_namespace" 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/">
<!--  NOTE: the parameter <unique_fully_qualified_namespace>MUST be replaced by fully qualified
unique identifiers for the actual implementation.
-->
   <!--
      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.IPCablecom.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.IPCablecom.com/pcsp/i01"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
         </input>
         <output name="Get0Out">
            <soap:body use="encoded" amespace="http://www.IPCablecom.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.IPCablecom.com/pcsp/i01"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
         </input>
         <output name="Put1Out">
            <soap:body use="encoded" namespace="http://www.IPCablecom.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.IPCablecom.com/pcsp/i01"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
         </input>
         <output name="Delete2Out">
            <soap:body use="encoded" namespace="http://www.IPCablecom.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.IPCablecom.com/pcsp/i01"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
         </input>
         <output name="CmdStatus3Out">
            <soap:body use="encoded" namespace="http://www.IPCablecom.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>IPCablecom CMS Subscriber Provisioning Service I01</documentation>
      <port name="PcspI01Service" binding="tns:PcspI01Service">
         <soap:address location=""/>
      </port>
   </service>
</definitions>
```

Appendix I

Sample entity encodings

I.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>
```

I.2 PcspEndpoint 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>
```

I.3 PcspMta 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>
```

I.4 PcspCms 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>
```

I.5 PcspRelation 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 II

Sample object extension

II.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
<?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>
```

II.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 III

Data encoding evaluation

Options considered for the encoding of data objects and messaging (see Table III.1).

III.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 to this work. It is a public standard – it is not the proprietary development of any company. The v1.0 specification was accepted by the W3C as a Recommendation on 10 Feb. 1998. The specification may be found at http://www.w3.org/TR/REC-xml.

III.2 ASN.1/BER

ASN.1 is a text-based, platform-independent syntax used to represent arbitrary data structures. It is popularly employed for SNMP MIB representations. The Basic Encoding Rules (BER) is a simple 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.

III.3 Proprietary ASCII

Proprietary encodings are out of scope.

III.4 SDP (Session Description Protocol)

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

III.5 RADIUS

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

III.6 SQL

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

III.7 Options summary

Table III.1/J.177 – Data encoding options

Option	Pro	Con
XML	Flexible, ASCII tag-based. Provides syntax checking through use of schema/DTD. Easy to extend without affecting 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 is not 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

Table III.1/J.177 – Data encoding options

Option	Pro	Con
SDP (Session Description Protocol)		Not flexible in terms 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.

III.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 IV

Transport protocol evaluation

IV.1 TFTP with IPSec

Trivial File Transfer Protocol (TFTP) is already in use within the IPCablecom infrastructure (DOCSIS). It is intended as a lightweight file transfer protocol.

IV.2 Batched RADIUS – Multiple records in single request via event messages

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 predefined 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, and it is very limited in new tags (it can only define a total of 255 tags, compared 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

IV.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+)). See the IETF RFC Diameter Base Protocol.

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

IV.4 Distributed object systems

IV.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 Unix 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).

IV.4.2 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.

IV.5 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. In a typical HTTP transaction:

1) client establishes 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 (e.g., 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.

IV.6 Options summary

See Table IV.1.

Table IV.1/J.177 – Transport options

Option	Pro	Con
TFTP with IPSec	Lightweight. Already implemented for DOCSIS.	Does not provide a two-way communication.
RADIUS	Flexible: includes vendor-specific and customer-specific fields. IPCablecom 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.

Table IV.1/J.177 – Transport options

Option	Pro	Con
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 regarding interoperability or various CORBA/ORB products.
DCOM	DCOM and CORBA/IIOP are similar technologies.	Basically a Microsoft Windows-only option. Will not 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 such 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.

IV.7 Recommendation: HTTP 1.1

Based on the analysis above, it is recommended that HTTP 1.1 be the transport protocol used.

BIBLIOGRAPHY

– IETF RFC 2139 (1997), *RADIUS Accounting*.

– COMMON LANGUAGE® General Codes: Telecommunications Service Providers IAC Codes, Exchange Carrier Names, Company Codes – *Telcordia and Region Number*: BR-751-100-112, Issue 4, April 2002.

– Telcordia Local Exchange Routing Guide (LERG), *Telcordia Technologies, Inc*.

SERIES OF ITU-T RECOMMENDATIONS

Series A	Organization of the work of ITU-T
Series D	General tariff principles
Series E	Overall network operation, telephone service, service operation and human factors
Series F	Non-telephone telecommunication services
Series G	Transmission systems and media, digital systems and networks
Series H	Audiovisual and multimedia systems
Series I	Integrated services digital network
Series J	**Cable networks and transmission of television, sound programme and other multimedia signals**
Series K	Protection against interference
Series L	Construction, installation and protection of cables and other elements of outside plant
Series M	Telecommunication management, including TMN and network maintenance
Series N	Maintenance: international sound programme and television transmission circuits
Series O	Specifications of measuring equipment
Series P	Telephone transmission quality, telephone installations, local line networks
Series Q	Switching and signalling
Series R	Telegraph transmission
Series S	Telegraph services terminal equipment
Series T	Terminals for telematic services
Series U	Telegraph switching
Series V	Data communication over the telephone network
Series X	Data networks, open system communications and security
Series Y	Global information infrastructure, Internet protocol aspects and next-generation networks
Series Z	Languages and general software aspects for telecommunication systems

Printed in Switzerland
Geneva, 2006