



***Society of Cable
Telecommunications
Engineers***

**ENGINEERING COMMITTEE
Data Standards Subcommittee**

AMERICAN NATIONAL STANDARD

ANSI/SCTE 159-2 2017

**Multimedia Application and Service Part 2:
IPCablecom Multimedia Web Services**

NOTICE

The Society of Cable Telecommunications Engineers (SCTE) Standards and Operational Practices (hereafter called “documents”) are intended to serve the public interest by providing specifications, test methods and procedures that promote uniformity of product, interchangeability, best practices and ultimately the long term reliability of broadband communications facilities. These documents shall not in any way preclude any member or non-member of SCTE from manufacturing or selling products not conforming to such documents, nor shall the existence of such standards preclude their voluntary use by those other than SCTE members.

SCTE assumes no obligations or liability whatsoever to any party who may adopt the documents. Such adopting party assumes all risks associated with adoption of these documents, and accepts full responsibility for any damage and/or claims arising from the adoption of such documents.

Attention is called to the possibility that implementation of this document may require the use of subject matter covered by patent rights. By publication of this document, no position is taken with respect to the existence or validity of any patent rights in connection therewith. SCTE shall not be responsible for identifying patents for which a license may be required or for conducting inquiries into the legal validity or scope of those patents that are brought to its attention.

Patent holders who believe that they hold patents which are essential to the implementation of this document have been requested to provide information about those patents and any related licensing terms and conditions. Any such declarations made before or after publication of this document are available on the SCTE web site at <http://www.scte.org>.

All Rights Reserved

© Society of Cable Telecommunications Engineers, Inc. 2017
140 Philips Road
Exton, PA 19341

Note: DOCSIS® is a registered trademark of Cable Television Laboratories, Inc., and is used in this document with permission.

Contents

1	SCOPE	4
1.1	INTRODUCTION AND PURPOSE	4
1.2	ORGANIZATION OF DOCUMENT	5
1.3	REQUIREMENTS	5
2	REFERENCES	6
2.1	NORMATIVE REFERENCES	6
2.2	INFORMATIVE REFERENCES	6
2.3	REFERENCE ACQUISITION	6
3	TERMS AND DEFINITIONS	7
4	ABBREVIATIONS AND ACRONYMS	8
5	TECHNICAL OVERVIEW	10
5.1	SERVICE GOALS	10
5.2	APPLICATION CONSIDERATIONS	10
5.2.1	Protocol Design Considerations	10
5.3	ARCHITECTURE AND POLICY MODEL	13
5.3.1	Subscriber	15
5.3.2	Subscription	15
5.3.3	Service	15
5.3.4	PolicySet	15
5.3.5	PolicyGroup	15
5.3.6	PolicyRule	15
5.3.7	PolicyCondition	15
5.3.8	Classifier	15
5.3.9	CompoundPolicyCondition	15
5.3.10	PolicyAction	16
5.3.11	FlowSpecTrafficProfile	16
5.3.12	DOCSISServiceClassTrafficProfile	16
5.3.13	DOCSISSpecificParameterizationTrafficProfile	16
5.3.14	CompoundPolicyAction	16
6	WS INTERFACE DESCRIPTION	17
6.1	WS PROFILE	17
6.1.1	Encoding	17
6.1.2	SOAP standard	17
6.1.3	Transport layer	17
6.1.4	WS-Addressing	17
6.1.5	WS-Security: Username Token Profile	18
6.1.6	Support for Event subscription	18
6.2	SERVICES	19
6.2.1	Element list	19
6.3	WS PROTOCOL OPERATION	28
6.3.1	Operation Sequence for Requesting a Service	28
6.3.2	Procedures for Querying a Service Status	31
6.3.3	Procedures for Modifying a Service Request	32
6.3.4	Procedure for Deleting a Service Request	32
6.3.5	Procedures for Synchronization	33
6.3.6	Procedures for Subscribing to Events	34

6.3.7	<i>Generating and Notifying Events</i>	34
6.3.8	<i>Error Procedures</i>	34
7	SECURITY REQUIREMENTS	37
7.1	INTRODUCTION	37
7.2	TRANSPORT ENCRYPTION AND MUTUAL ENDPOINT AUTHENTICATION	37
7.3	MESSAGE AUTHENTICATION & AUTHORIZATION.....	38
8	OPEN ISSUES	39
ANNEX A	XML SCHEMA (NORMATIVE)	40
ANNEX B	WSDL SPECIFICATION (NORMATIVE)	50
APPENDIX I	WEB SERVICES EVENTING (WS-EVENTING)	53

Figures

FIGURE 1 - END TO END QOS.....	4
FIGURE 2 - STATELESS SCD RENDEZVOUS PROTOCOL EXAMPLE	11
FIGURE 3 - IPCABLECOM MULTIMEDIA ARCHITECTURE.....	13
FIGURE 4 - POLICY MODEL	14
FIGURE 5 - CONTEXTID CREATION	23

Tables

TABLE 1 - WEB SERVICE OPERATIONS	19
TABLE 2 - RESERVERESOURCESREQUEST MESSAGE ARGUMENTS	29
TABLE 3 - RESERVERESOURCESRESPONSE MESSAGE ARGUMENTS	30
TABLE 4 - COMMITRESOURCESREQUEST MESSAGE ARGUMENTS	31
TABLE 5 - COMMITRESOURCESRESPONSE MESSAGE ARGUMENTS.....	31
TABLE 6 - QUERYAVAILABLESERVICES RESPONSE MESSAGE ARGUMENTS	32
TABLE 7 - RELEASERESOURCES REQUEST MESSAGE ARGUMENTS.....	33
TABLE 8 - QUERYCONTEXTS REQUEST MESSAGE ARGUMENTS.....	33
TABLE 9 - QUERYCONTEXTS RESPONSE MESSAGE ARGUMENTS	33
TABLE 10 - SERVER ERRORS	36
TABLE 11 - CLIENT ERRORS	36

1 SCOPE

NOTE: This document is identical to SCTE 159-2 2010 except for informative components which may have been updated such as the title page, NOTICE text, headers and footers. No normative changes have been made to this document.

1.1 Introduction and Purpose

This specification provides a simple, open interface between a generic Application Server (AS) and an IPCablecom Multimedia Application Manager (AM). Specifically, this specification defines a common Web Service (WS) interface to the IPCablecom Multimedia Application Manager (AM) that enables an AS to dynamically request network resources on the cable operator's access network. The primary goal of this interface is to allow AS developers to rapidly create new applications in shorter timeframes and without having a deep knowledge of the cable operator's access technology. This interface is based on the SOAP/eXtensible Markup Language (SOAP/XML).

As indicated above, the scope of this specification is limited to the interface between the AS and the AM. The following figure identifies all the interfaces that are involved in this scenario. It also provides a high-level view of the involved operations:

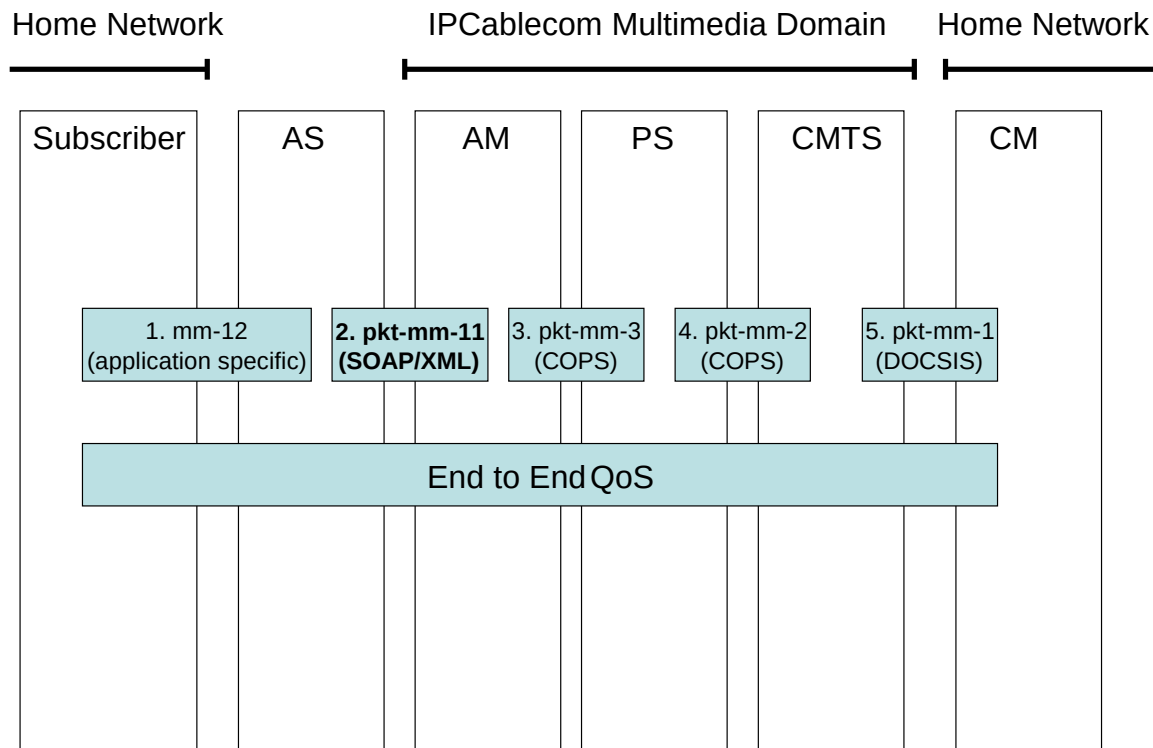


Figure 1 - End to End QoS

The operations are:

1. Subscriber requests an application from an AS, using an application specific interface (mm-12) such as HTTP, SIP or others.
2. The AS requests access network resources via the pkt-mm-11 interface.
3. The AM requests policy changes through the pkt-mm-3 interface.
4. The Policy Server sets gates on the CMTS through the pkt-mm-2 interface.
5. The CMTS installs service flows to/from the CM through the pkt-mm-1 interface.

In case of success, the AM begins with the "forwarding" to the application and the end-to-end QoS is established.

This document is intended for the following audience:

- Application developer on the AS side.
- AM vendors providing this interface.

1.2 Organization of document

Section 5 provides one use case, which shows that this interface has to be usable for applications, requiring the rendezvous protocol. It also provides a policy information model, which was used for developing this interface.

Section 6 defines the message interface including the WS requirements this interface has to fulfill.

Section 7 outlines the security requirements for the interface.

1.3 Requirements

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

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

2 REFERENCES

2.1 Normative References

The following documents contain provisions, which, through reference in this text, constitute provisions of this standard. At the time of publication, the editions indicated were valid. All standards are subject to revision, and users of this document are encouraged to investigate the possibility of applying the most recent editions of the documents listed below.

- [HTTPS] IETF RFC 2818, HTTP Over TLS, May 2000.
- [MM] SCTE 159-2 2010, IPCablecom Multimedia Specification
- [SEC] SCTE 165-10 2009, IPCablecom 1.5 Part 10:Security Specification
- [SOAP] SOAP Version 1.2:
W3C Recommendation, "SOAP Version 1.2 Part 1: Messaging Framework", 24 June 2003.
W3C Recommendation, "SOAP Version 1.2 Part 2: Adjuncts", 24 June 2003.
- [WSS] OASIS standard, "Web Services Security: Username Token Profile 1.0", OASIS Standard 200401, March 2004.

2.2 Informative References

- [CIM] DMTF Common Information Model (CIM) Specification, +CIM Schema: Version 2.9.1, January 2005.
- [RFC2222] IETF RFC 2222, Simple Authentication and Security Layer (SASL), October 1997.
- [RFC2617] IETF RFC 2617, HTTP Authentication: Basic and Digest Access Authentication, June 1999.
- [UPNP] UPnP QoS Manager:1, Service Document, December 16, 2004.
- [WSA] W3C Candidate Recommendation, Web Services Addressing (WS-Addressing), 17 August 2005.
- [WSDL] Web Services Description Language (WSDL) 1.1. <http://www.w3.org/TR/wsdl>
- [WSE] Appendix I of this document, Web Services Eventing (WS-Eventing), Public Draft Release, August 2004.
- [XML] Extensible Markup Language (XML) 1.0. <http://www.w3.org/TR/REC-xml>
- [XPath] W3C Recommendation, "XML Path Language (XPath)", 16 November 1999.

2.3 Reference Acquisition

- Internet Engineering Task Force (IETF), Internet: <http://www.ietf.org>
- Distributed Management Task Force, Inc. (DMTF), Internet: <http://www.dmtf.org>
- World Wide Web Consortium, Internet: <http://www.w3.org>
- Organization for the Advancement of Structured Information Standards (OASIS), Internet: <http://www.oasis-open.org>

3 TERMS AND DEFINITIONS

This specification uses the following terms:

Application Manager	A system that interfaces to Policy Server(s) for requesting QoS-based service on behalf of an end-user or network management system.
Cable Modem Termination System	Device at a cable head-end which implements the DOCSIS® RFI MAC protocol and connects to CMs over an HFC network.
Policy Server	A system that primarily acts as an intermediary between AM(s) and CMTS(s). It applies network policies to AM requests and proxies messages between the AM and CMTS.
Quality of Service	Method used to reserve network resources and guarantee availability for applications.

4 ABBREVIATIONS AND ACRONYMS

This specification uses the following abbreviations:

AM	Application Manager
AS	Application Server
BEEP	Blocks Extensible Exchange Protocol
CIM	Common Information Model
CM	DOCSIS® Cable Modem
CMTS	Cable Modem Termination System
COPS	Common Open Policy Service. Defined in RFC2748.
DMTF	Distributed Management Task Force, Inc.
EP	SIP Edge Proxy
FQDN	Fully Qualified Domain Name
HTTP	Hypertext Transport Protocol
IETF	Internet Engineering Task Force
IP	Internet Protocol
MAC	Media Access Control
POP3	Post Office Protocol version 3
PS	Policy Server
QoS	Quality of Service
OP	Originating Proxy
RFC	Request for Comments
RSVP	Resource Reservation Protocol. Defined in RFC2205.
SIP	Session Initiation Protocol
SMTP	Simple Mail Transport Protocol
TCP	Transmission Control Protocol
TP	Terminating Proxy

UA	User Agent
UDP	User Datagram Protocol
URI	Universal Resource Identifier
URL	Universal Resource Locator
VOD	Video On Demand
VoIP	Voice over IP
W3C	World Wide Web Consortium
WS	Web Service
WSDL	Web Service Description Language
XML	eXtensible Markup Language

5 TECHNICAL OVERVIEW

This section consists of the background material used for writing this specification. Some readers may find the policy model useful to understand the detailed protocol interface specifications that follow in Section 6. This section also provides insight into the service goals and application considerations that this WS interface design addresses.

5.1 Service Goals

This document provides technical requirements for application servers in order to request DOCSIS-based network resources. It is a goal of this specification that application developers are able to rapidly develop new applications without being experts in the access technology of the cable operator's network. The simplicity of this interface is achieved by introducing an abstract service layer, which provides an access point to the underlying network policies. In the simplest case, application developers only need pass subscriber and service identifiers as arguments when requesting network resources.

5.2 Application Considerations

This WS interface specification needs to accommodate a wide range of existing applications and be generic enough for future applications. The goal of this section is to present generic application considerations and to outline how their needs are met given the presently defined IPCablecom Multimedia framework. This WS interface has considered applications with the following characteristics:

- End-client only application invocation (e.g., turbo button, media streaming, etc.) where resource requests, such as QoS, originate predominantly from the subscriber's end client.
- Rendezvous protocols where one end client may not know the IP addresses or other needed attributes of other end clients until communication with these other end clients has occurred (e.g., some "call/session control" protocols).
- Application designs where the application is stateful (e.g., most VoIP softswitch designs).
- Application designs where the application is stateless (e.g., many stateless HTTP and SIP designs).

Rendezvous protocols are important for VoIP, videoconferencing and some types of VoD applications. Stateless applications are also foundational in the design of some highly scalable Internet system architectures.

The characteristics of rendezvous and non-rendezvous applications and stateful and stateless applications are important design considerations for this WS interface definition. Additionally the semantics of the services defined in the WS interface has also accommodated these characteristics.

Because rendezvous applications are such an important application class, it is worthwhile to explicitly outline their generic behavior in order that the service semantics (described later in this document) will become apparent. The next section outlines characteristics of these rendezvous applications. This next section also highlights the so-called two-phase "reserve then commit" resource model used in IPCablecom Multimedia. This two-phase model also impacts the semantics behind the "reserve" and "commit" service definitions in the web services interface that is defined later in this document.

5.2.1 Protocol Design Considerations

A stateless Service Control Domain (SCD) rendezvous application in an IPCablecom Multimedia environment is used here as an example. This example is modeled after HTTP or SIP operation and is greatly simplified; only particular IPCablecom Multimedia and proxy elements are shown to illustrate the Application Server (AS) to

Applications Manager (AM) interface. In this example, the AS is a stateless HTTP or SIP proxy element. Commentary for the slightly less complex stateful SCD case is deferred to the end of this section.

For purposes of clarity, both ends of the communication, the origination end and the termination end are depicted here in Figure 2. However, only the origination end is considered here for the purposes of the web services interface explanation. The AS in this case is the Originating Proxy (OP). Figure 2 depicts the relevant application signaling and web services operations. Only Messages 2, 7 and 10 between the AS and AM are part of this web services interface specification.

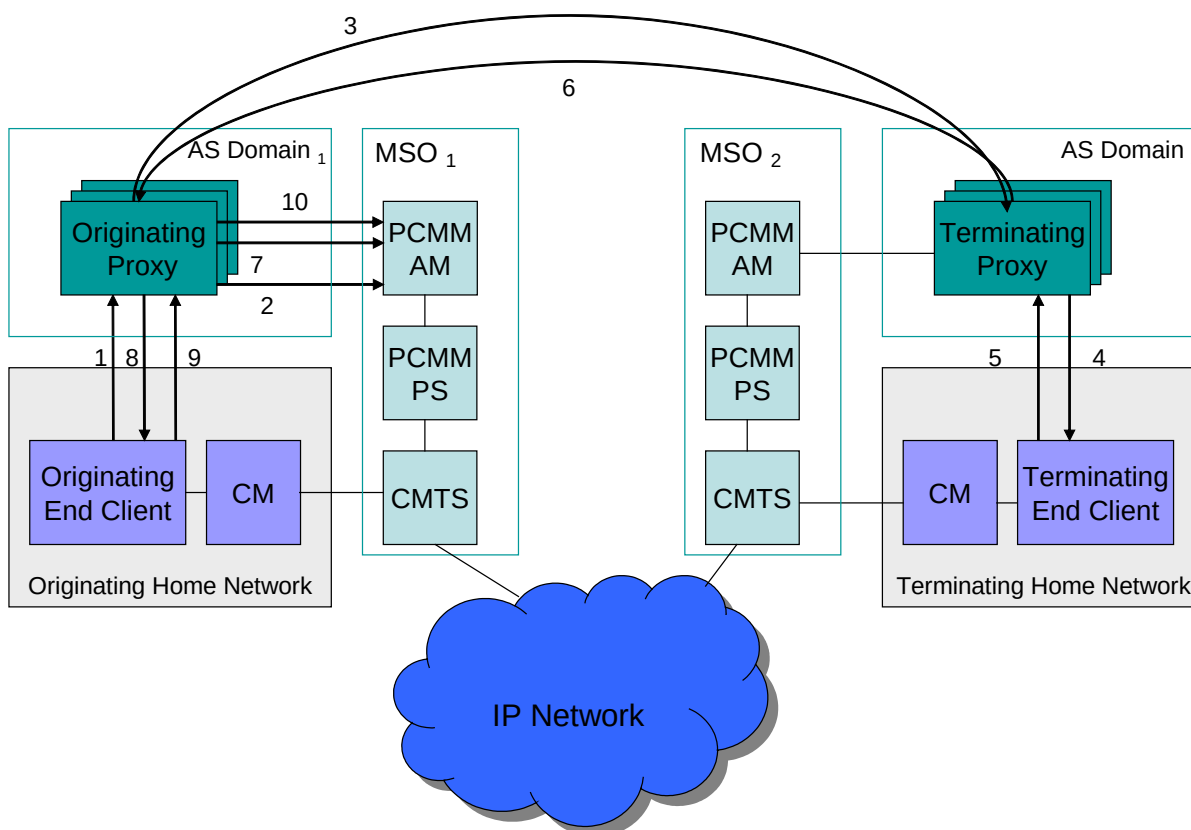


Figure 2 - Stateless SCD Rendezvous Protocol Example

- Message 1:** The Originating End Client initiates communications to some end client, depicted here as the Terminating End Client. Depending on the application involved, this step usually involves the resolution of a URI or URL (e.g., a SIP telephone URL) to one or more potential terminating end clients. Indeed, resolution to multiple terminating end clients may occur at a later stage of signaling (e.g., after Message 3).
- Message 2:** At this step, the OP may not know many of the Terminating End Client attributes (such as their IP addresses or ports) as the terminating end client(s) have not yet been contacted. However, the OP (acting here as an AS) sends a ReserveResourcesRequest message via the WS interface to the AM for the purpose of reserving resources, such as QoS, for the originating end of the communication. At the AM, this session is identified via the BaseID component of the ContextID (6.2.1.2), which has been provided to it by the OP.

- Message 3:** The OP forwards the message and it eventually reaches the Terminating Proxy (TP). At this step, it is instructive to note that, due to forking and other features of these protocols, there may be multiple TPs.
- Message 4:** The TP sends the message to the Terminating End Client. At this step it is instructive to note that there may be multiple Terminating End Clients served by a singular TP. The key point here is to note that a singular "Message 1" may result in a multiplicity of "Message 4" messages to multiple Terminating End Clients.
- Message 5:** The Terminating End Client replies to "Message 4". The reply contains the IP address of the Terminating End Client and optionally other session or other end client attributes.
- Message 6:** The TP forwards the response to the OP. The response includes the IP address of the Terminating End Client.
- Message 7:** For each message received from a different Terminating End Client, the OP sends an updated ReserveResource message via the WS interface to the AM with additional, previously undetermined, information such as the IP Address of the Terminating End Client. Each of these ReserveResource messages is associated with a different Terminating End Client. At the AM each of these reservations has a unique ContextID that is: 1) associated with the same "Message 2" reservation by virtue of each having the same "BaseID" portion of the ContextID, and 2) differentiated from each other by virtue of differing (and unique) "ModifierID" portions of the ContextID.
- Message 8:** The OP forwards the response from each Terminating End Client (Message 5 and 6) to the Originating End Client.
- Message 9:** The Originating End Client selects one (or more) of the responses from the Terminating End Clients to establish communication with the Terminating End Client and sends one (or more) messages to the OP.
- Message 10:** The OP sends a CommitResource message to the AM for each of the Terminating End Client sessions it wishes to communicate with. This action thereby commits the resources that were previously reserved in the ReserveResource of Message 2 and re-reserved in the updated ReserveResource Message 7.
- Message 11:** (Not shown) The OP forwards the (Message 9) message to the Terminating End Clients to complete the rendezvous signaling.

Note that Message 9 and 10 would also be sent to each Terminating End Clients that the Originating End Client did not wish to establish communication with. In this case, the action in Message 10 would be to send a ReleaseResource message via the WS interface to the AM indicating that it did not require the reserved resource for communication with these Terminating End Clients. Reserved resources are also released if there is not a timely CommitResource or ReserveResource sent to the AM (a ReserveResource message can be used to refresh an existing reservation).

This example illustrates that the WS interface must allow the ContextIDs (with BaseIDs and optional ModifierIDs) created by the AS and provided to the AM to evolve over time as the communication is fully established for the stateless Service Control Domain example provided here.

A slightly simpler model can be used for stateful Service Control Domains; in this case unique ContextIDs may optionally be created by the AM and provided to the AS for every candidate Terminating End Client.

Therefore, the WS interface allows the AS or AM to provide the ContextID, dependent on whether stateless or stateful SCD designs are employed by the application.

5.3 Architecture and Policy Model

ASes use this interface to request resources in the cable operator network on behalf of its clients. Looking at the architecture of the IPCablecom Multimedia environment and understanding the policy model, used by the IPCablecom Multimedia components, makes it easier to understand the WS operations and their arguments. In the following subsections, a policy model, which is inspired by the Common Information Model (CIM) Schema 2.9.1 [CIM], is discussed.

This document does not mandate the implementation of the CIM Policy model by the AM. The following model was used to develop this interface. It is up to AM vendors to decide exactly how the policies and services are modeled internally.

As illustrated in Figure 3, the AM retrieves the service and policy data from a repository. The interfaces to this repository are not within the scope of this specification.

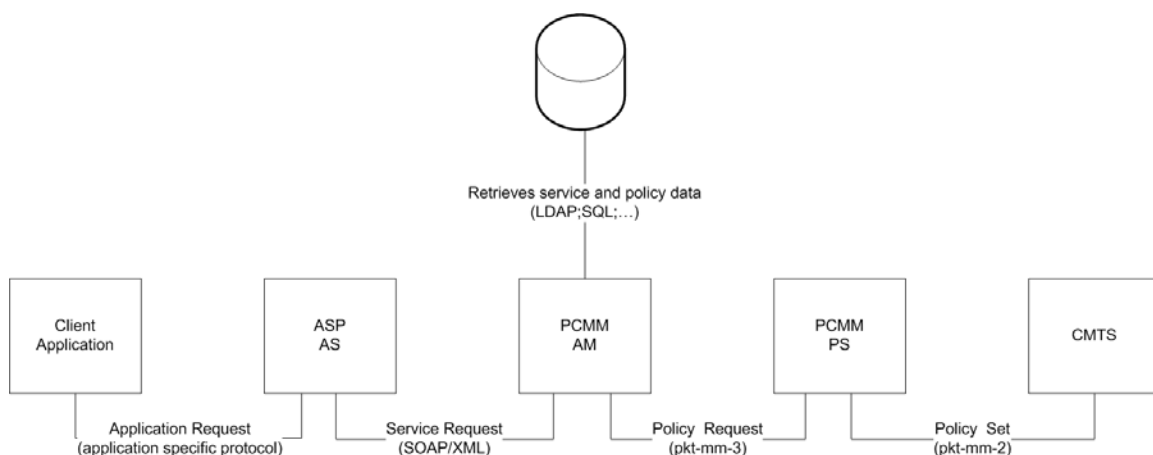


Figure 3 - IPCablecom Multimedia Architecture

This information is provisioned by the cable operator to the data repository, which could be a relational database, an LDAP server or any other storage medium. The AM converts this data to its internal information model, which includes all the relationships between the subscribers, offered services and defined policies.

These relationships are shown in Figure 4.

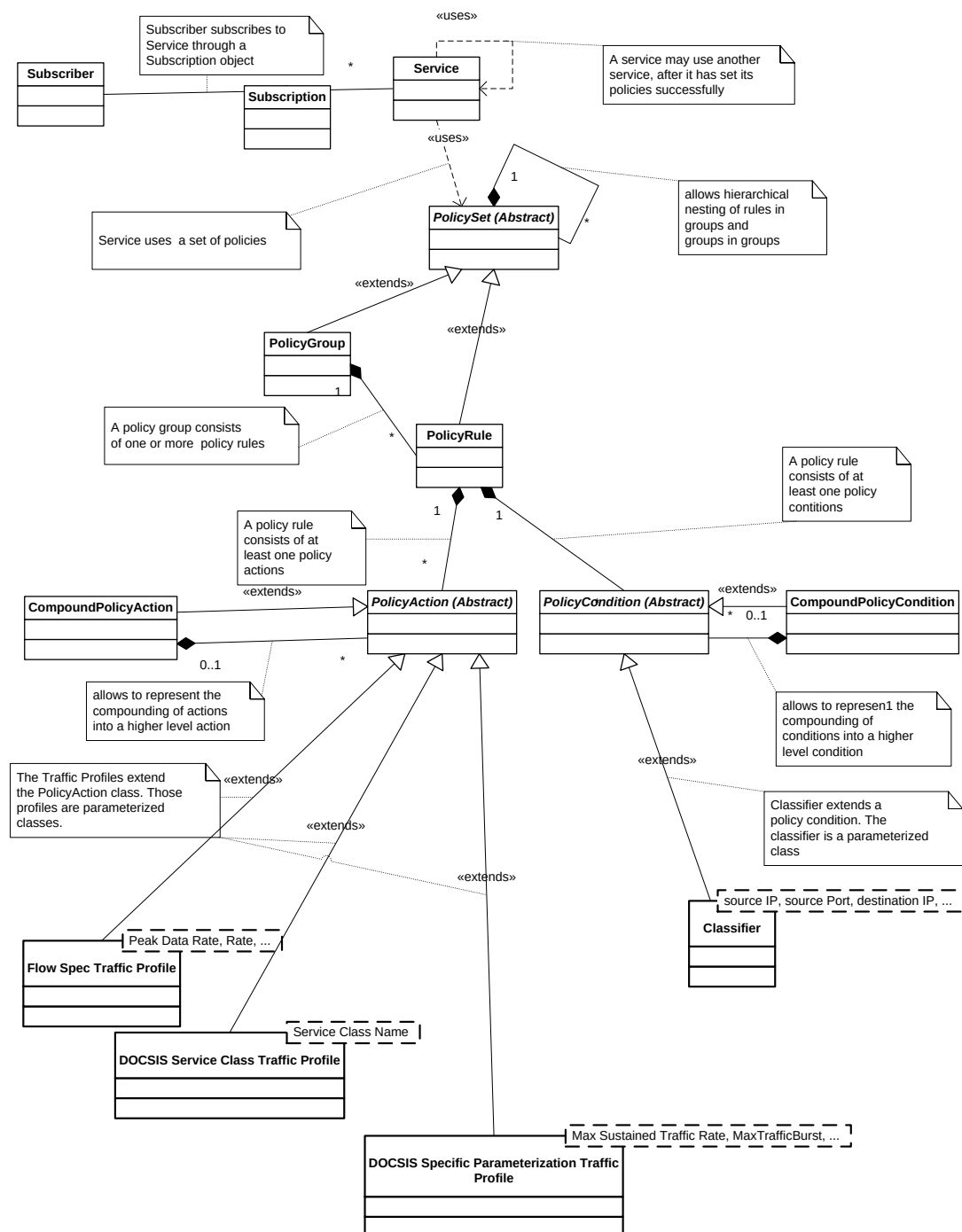


Figure 4 - Policy Model

After the AM retrieved all the data and converted the data internally into the object information model, depicted in Figure 4, it is able to send policy requests down to the PS. Those policy requests may be triggered by service

requests via the SOAP/XML interface from an AS. If the service or resource request only contains the identifier of the service and the identification of the end-customer, the AM sends the policy model with its default values (defined by the cable operator) down to the PS. The service requester has also the opportunity to "overwrite" some of those default values, by including the new values in the service request. The argument list is discussed later in Section 6.

The objects and relationships of the information model, depicted in Figure 4 are discussed in the remainder of this Section.

5.3.1 Subscriber

A subscriber is an entity that can subscribe to services. In this context, "subscribe to a service" means that a subscriber agreed to the terms of the service.

5.3.2 Subscription

A subscription is an object that represents an enrollment to a given service for a specific subscriber. In other words, a subscriber enrolls or subscribes to a service through a subscription object (e.g., the subscription object creates a relationship between the subscriber and a service).

5.3.3 Service

A service is an object that contains the information to represent the functionality offered to a subscriber. Service objects "use" a set of policies, which implement this offered functionality in the cable operator network.

5.3.4 PolicySet

The PolicySet object provides an abstraction for a set of rules to manage and control access to network resources. PolicyGroup and PolicyRule objects are derived from the PolicySet object.

5.3.5 PolicyGroup

A PolicyGroup object represents an aggregation of policy sets that have the same decision strategy. The decision strategy defines the evaluation method used for policies contained in the policy group (e.g., "first matching"). A PolicyGroup object can be seen as a container for a set of related policy rules and policy groups.

5.3.6 PolicyRule

This object is used to organize the conditions and actions that comprise the policy rule. PolicyRule objects consist of conditions that are used to match traffic and actions that specify the action to be taken if the traffic matches.

5.3.7 PolicyCondition

This abstract class is used to identify conditions. It is also the superior class of the Classifier object.

5.3.8 Classifier

Classifiers are used for defining 'classifiers' that a packet or packet flow must contain; for example, network protocol, source and destination addresses. This is a "dynamic" or "parameterized" object, meaning the values of those fields provided in a ResourceRequest can be dynamically substituted in place of those provided in the service definition.

5.3.9 CompoundPolicyCondition

CompoundPolicyCondition is used to represent compound conditions formed by aggregating simpler policy conditions.

5.3.10 PolicyAction

Policy actions are used to define the 'action' that the network takes on packets that match conditions. The PolicyAction object serves a superior class for FlowSpecTrafficProfile, DOCSISServiceClassTrafficProfile and DOCSISSpecificParameterizationTrafficProfile.

5.3.11 FlowSpecTrafficProfile

The FlowSpecTrafficProfile class defines the Traffic Profile of the IP flow associated with an IPCablecom Multimedia Gate through an RSVP-like parameterization scheme. This is a "dynamic" or "parameterized" class, meaning the values of attributes on an instance can be dynamically substituted. The list of attributes can be found in [MM].

5.3.12 DOCSISServiceClassTrafficProfile

The DOCSISServiceClassNameTrafficProfile contains the DOCSIS Service Class Name, indicating the DOCSIS Service Class to be used to describe the QoS attributes. The DOCSIS Service Class Name can be substituted.

5.3.13 DOCSISSpecificParameterizationTrafficProfile

This class allows defining a Traffic Profile, using explicit DOCSIS parameters of the DOCSIS flow. The exact list of attributes is specified in [MM].

5.3.14 CompoundPolicyAction

A CompoundPolicyAction is used to represent compound conditions formed by aggregating simpler policy actions.

6 WS INTERFACE DESCRIPTION

This section describes the interface between a generic Application Server (AS) and the Application Manager.

The section explains in detail the syntax of all the arguments or elements passed via this interface, and the semantics of the function calls that can be performed using this WS interface.

6.1 WS Profile

This section defines the WS standards this interface **MUST** comply with.

6.1.1 Encoding

This interface is presented as a sequence of XML-serialized messages. Application Managers and Application Servers **MUST** support SOAP document-style encoding, which is also known as "document/literal" or "doc/lit".

6.1.2 SOAP standard

Application Managers and Application Servers **MUST** conform to the SOAP 1.2 standard [SOAP]. This allows for extensibility and fault-transmission. This implies that the SOAP 1.2 processing model **MUST** be followed, including the "soap:role" (with default value "ultimateRecipient") and "soap:mustUnderstand" attributes in the SOAP header. The "soap:Fault" semantics and format **MUST** be followed. This is described in more detail in Section 6.3.6.

6.1.3 Transport layer

Application Managers **MUST** support [HTTPS] as the transport protocol and conform to the security requirements in Section 7.2 of this document.

Application Servers **MUST** support [HTTPS] as the transport protocol and conform to the security requirements in Section 7.2 of this document.

The SOAP 1.2 format is transport-agnostic; hence, this interface **MAY** in the future support transport protocols (e.g., BEEP, SMTP/POP3) other than HTTPS. However, specifications relating to alternate transports are not covered by this document.

6.1.4 WS-Addressing

Application Managers and Application Servers **MAY** support the use of WS-Addressing [WSA] with this SOAP interface.

Application Servers **SHOULD** support the use of WS-Addressing [WSA] with this SOAP interface if they use transport protocols other than HTTPS for SOAP transport. However, specifications relating to alternate transports are not covered by this document.

Application Managers **SHOULD** support the use of WS-Addressing [WSA] with this SOAP interface if they support transport protocols other than HTTPS for SOAP transport. However, specifications relating to alternate transports are not covered by this document.

If an Application Server supports WS-Addressing, it **MUST** tag each message with an identifier that is unique within that client - the MessageId (as defined in [WSA]) with a soap:mustUnderstand attribute set to value '0'.

If an Application Manager supports WS-Addressing, it MUST set the WS-Addressing "RelatesTo" header element of response messages to reflect the MessageId, if present, of the request message. If an AM does not support WS-Addressing, it must treat any WS-Addressing related headers like MessageId in accordance with the soap:mustUnderstand attribute specified for those headers in conformance with the SOAP [SOAP] specification.

6.1.5 WS-Security: Username Token Profile

Application Managers MUST require the specification of the Username from the WS-Security: Username Token Profile [WSS].

Application Servers MUST provide the Username in the SOAP header for every request as per [WSS]. The Username will be a value assigned by the cable operator and identifying an AS.

Application Managers MAY provide an authentication and authorization mechanism based on [WSS] as described in Section 7.3 of this specification.

6.1.6 Support for Event subscription

This specification provides a way for the AS to subscribe to events indicating autonomous changes to sessions by the network. This allows the AS to be notified of changes to requested services (such as when a CMTS preempts a gate or a gate timer expires or the usage limit on a gate is exceeded).

The AM MAY support this eventing feature.

The WS-Eventing Specification [WSE] MUST be used as the basis for subscribing to events, in case the AM provides this feature. The WS-Eventing specification is described in Appendix I.

An AM supporting this feature MUST provide SOAP endpoints supporting the WS-Eventing WSDL port types EventSource and SubscriptionManager.

An AS that uses this Event subscription feature MUST support the use of WS-Addressing.

An AM that uses this Event subscription feature MUST support the use of WS-Addressing.

6.1.6.1 IPCablecom Multimedia Event Filter Dialect

The WS-Eventing specification allows the AS to specify filters to allow an AS to only receive events matching certain criteria. The WS-Eventing allows multiple filter dialects to be used for filtering events. It also defines one such dialect to be the XPath specification [XPATH]. However, arbitrary XPath filters may be expensive for the AM to implement. Therefore this specification defines a filter dialect (called the IPCablecom Multimedia Event Filter dialect) identified by the namespace URL <http://www.cablelabs.com/PCMM/1.0/xsd/reg/CLAB-PCMM-WS> for use with this specification.

Filter criteria using this dialect are specified using the query-context-req request message as defined in Annex B. See Section 6.3.5.1 QueryContexts message for semantics of the matching operation for this filter. The following example depicts an event subscription with an event filter using this specification.

```
<Subscribe xmlns="http://schemas.xmlsoap.org/ws/2004/08/eventing"
  xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing">
  <Delivery Mode="http://schemas.xmlsoap.org/ws/2004/08/eventing/DeliveryModes/Push">
    <NotifyTo>
      <wsa:Address>http://www.example.org/PCMM/EventListener</wsa:Address>
    </NotifyTo>
  </Delivery>
  <Expires>P0Y0M0DT0H30M0S</Expires>
  <Filter Dialect=" http://www.cablelabs.com/PCMM/1.0/xsd/reg/CLAB-PCMM-WS ">
```

```

    <pcmm: QueryContextsReq xmlns:pcmm=" http://www.cablelabs.com/PCMM/1.0/xsd/reg/CLAB-PCMM-WS " >
      <pcmm:ServiceName>Turbo</pcmm: ServiceName>
      <pcmm:SubscriberID>...</pcmm: SubscriberID>
      <pcmm:ContextID><.../></pcmm:ContextID>
    </pcmm: QueryContextsReq>
  </Filter>
</Subscribe>

```

WS-Eventing EventSource implementations for an AM MUST support the IPCablecom Multimedia Event Filter dialect.

6.2 Services

This WS provides an interface for requesting (reserving and/or committing) and releasing network resources on the cable operator network. It also provides the ability to retrieve all the applications or services a given AS can request. Furthermore, actual sessions can be queried, based on a given end customer, given application or given session identifier. ASes also have the opportunity to register for events, and the AM delivers updates according to this registry. Those services require a set of elements (or arguments), which are discussed in more detail.

Table 1 - Web Service Operations

Operation Name	Style	Messages	
ReserveResources	Document	Input	ReserveResourcesRequest
		Output	ReserveResourcesResponse
CommitResources	Document	Input	CommitResourcesRequest
		Output	CommitResourcesResponse
ReleaseResources	Document	Input	ReleaseResourcesRequest
		Output	ReleaseResourcesResponse
QueryAvailableServices	Document	Input	QueryAvailableServicesRequest
		Output	QueryAvailableServicesResponse
QueryContexts	Document	Input	QueryContextsRequest
		Output	QueryContextsResponse
ResourceStateNotification	Document	Output	ResourceStateNotification

6.2.1 Element list

This section describes the elements, which are either passed in the request and response messages of this WS interface, defined in Section 6.3, or used for the authentication and authorization layer of individual SOAP (request) messages, described in 7.3.

6.2.1.1 Classifier

The Classifier element allows the AS to identify the traffic flow for which it is requesting resources. The AS MUST format the element as defined in Annex A.

The Classifier identifies a single flow in a given direction. The direction indicated in the Classifier provided by the client is considered the upstream or inbound direction. The inverse of this flow defined by replacing the source IP address and port, with the destination IP address and port and vice versa, identifies the downstream or outbound flow.

If the Classifier elements were defined during the service creation, the AS MAY omit the Classifier element. If the AS chooses to include the Classifier element in a resource request message, the AS MUST include only a single Classifier element with at least one sub element populated. The AS requires knowledge of whether or not a service definition requires Classifiers to be specified, and which particular elements are necessary.

The AS MUST NOT mix unicast and multicast destination IP addresses for a single flow.

For Multicast flows (where the Destination IP Address in the classifier identifies a specific Multicast IP address) the AS encodes the classifier as follows:

- The Source IP address & the Mask/Prefix Length combination MUST resolve to a specific valid Unicast IP address value i.e., not resolve into a range of addresses. If the Source IP address & the Mask/Prefix Length combination resolve to the All-Zeros IP Address, it indicates that all Source IP addresses are permitted, i.e., Any Source Multicast (ASM).
- The Destination IP address & the Mask/Prefix Length combination MUST resolve to a single specific Multicast address
- The Source and Destination Port Number fields in the classifier are ignored by the CMTS for a Multicast flow.

If the AS includes a Classifier element with the initial resource request, it MUST include a Classifier element with all subsequent resource requests. If the AS does not include a Classifier element in the initial resource request, it MAY include the Classifier element in a subsequent request.

The AM MUST use the following precedence rules when determining which value to use for each Classifier element:

- If the element is included in the Classifier, use the value defined in the Classifier element.
- Else if the element was defined during the service and policy definition, use the value defined in the service and policy definition.
- Else use the 'assumed value' defined for each element below.

There are two formats defined for Classifiers – an IPv4 format and an IPv6 format corresponding to the use of IPv4 or IPv6 addresses in the classifier.

6.2.1.1.1 IPv4 Classifier

The IPv4 Classifier is denoted by the element named 'Classifier' in the XML Schema. The contents of the IPv4 classifier element are described as follows:

- The protocol element identifies the type of protocol. Integer values range from 0 to 257. The value 0 indicates no protocol match. The value 256 indicates any protocol. The value 257 matches packets with protocol type TCP or UDP. If this element is omitted, its assumed value should be 0.
- The string value of the sourceIpAddress element identifies the IP address (as seen at the CMTS) of the originator of the IP flow. A value of "0.0.0.0" indicates any IP address. If this element is omitted, its assumed value should be "0.0.0.0", and the assumed value of sourceIpMask should be "0.0.0.0".
- The string element sourceIpMask is used for identifying the mask value for the source IP address in order to classify multiple IP addresses for a single Classifier object. A value of "0.0.0.0" indicates any subnet. This element MUST NOT be included if sourceIpAddress is not included. If sourceIpAddress is omitted, or the sourceIpAddress has a value of "0.0.0.0", then the assumed value of sourceIpMask should be "0.0.0.0". Otherwise, if this element is omitted and sourceIpAddress is not omitted, then its assumed value should be "255.255.255.255".
- The element sourcePortStart identifies the start or low-end of the UDP or TCP port ranges for matching the IP flow. If this element is omitted its assumed value should be 0.

- The element `sourcePortEnd` identifies the end or high-end of the UDP or TCP port ranges for matching the IP flow. If this element is omitted and the `sourcePortStart` is included, then its assumed value should be the same as the value of `sourcePortStart`. If this element is omitted and the `sourcePortStart` is omitted, then its assumed value should be 65535.
- The string value of the `destinationIpAddress` element specifies the matching value of the IP address of the termination point of the IP flow. A value of "0.0.0.0" indicates any IP address. If this element is omitted, its assumed value should be "0.0.0.0", and the assumed value of `destinationIpMask` should be "0.0.0.0".
- The string element `destinationIpMask` is used for identifying a subnet for the destination IP address in order to classify multiple IP addresses for a single Classifier object. A value of "0.0.0.0" indicates any subnet. This element MUST NOT be included if `destinationIpAddress` is not included. If `destinationIpAddress` is omitted, or the `destinationIpAddress` has a value of "0.0.0.0", then the assumed value of `destinationIpMask` should be "0.0.0.0". Otherwise, if this element is omitted, and `destinationIpAddress` is not omitted, then its assumed value should be "255.255.255.255".
- The element `destinationPortStart` identifies the low-end of the UDP or TCP port ranges for matching the IP flow. If this element is omitted, its assumed value should be 0.
- The element `destinationPortEnd` identifies the high-end of the UDP or TCP port ranges for matching the IP flow. If this element is omitted, and the `sourcePortStart` is included, then its assumed value should be the same as the value of `sourcePortStart`. If this element is omitted and the `sourcePortStart` is omitted, then its assumed value should be 65535.

6.2.1.1.2 *Ipv6 Classifier*

The IPv6 Classifier is denoted by the element named 'IPv6Classifier' in the XML Schema. The contents of the IPv6 classifier element are described as follows:

- The `nextHeader` element identifies the type of transport protocol the classifier should match. It performs the same function as the `protocol` field in IPv4 classifier. The `protocol` element identifies the type of protocol. Integer values range from 0 to 257. The value 0 indicates no protocol match. The value 256 indicates any protocol. The value 257 matches packets with protocol type TCP or UDP. If this element is omitted, its assumed value should be 0.
- The string value of the `sourceIpAddress` element identifies the IP address (as seen at the CMTS) of the originator of the IP flow. An IP address value consisting of 16 zero octets (i.e., '::' or equivalent string representation) indicates the classifier matches any source IP address.
- The numeric element `sourcePrefixLen` is used for identifying the number of significant prefix bits of the `sourceIpAddress` to classify multiple IP addresses for a single Classifier object. A value of 0 indicates that the classifier matches any address. This element MUST NOT be included if `sourceIpAddress` is not included. If `sourceIpAddress` has a value of "::" or equivalent, then the assumed value of `sourcePrefixLen` is 0. Otherwise, if this element is omitted and `sourceIpAddress` has a value other than '::' or equivalent, then `sourcePrefixLen` is assumed to have a value of 128.
- The element `sourcePortStart` identifies the start or low-end of the UDP or TCP port ranges for matching the IP flow. If this element is omitted its assumed value should be 0.
- The element `sourcePortEnd` identifies the end or high-end of the UDP or TCP port ranges for matching the IP flow. If this element is omitted and the `sourcePortStart` is included, then its assumed value should be the same as the value of `sourcePortStart`. If this element is omitted and the `sourcePortStart` is omitted, then its assumed value should be 65535.
- The string value of the `destinationIpAddress` element identifies the IP address (as seen at the CMTS) of the terminator of the IP flow. An IP address value consisting of 16 zero octets (i.e., '::' or equivalent string representation) indicates that the classifier matches any destination IP address.
- The numeric element `destinationPrefixLen` is used for identifying the number of significant prefix bits of the `destinationIpAddress` to classify multiple IP addresses for a single Classifier object. A value of 0 indicates that

the classifier matches any address. This element **MUST NOT** be included if destinationIpAddress is not included. If destinationIpAddress has a value of "::" or equivalent, then the assumed value of destinationPrefixLen is 0. Otherwise, if this element is omitted and destinationIpAddress has a value other than '::' or equivalent, then destinationPrefixLen is assumed to have a value of 128.

- The element destinationPortStart identifies the low-end of the UDP or TCP port ranges for matching the IP flow. If this element is omitted its assumed value should be 0.
- The element destinationPortEnd identifies the high-end of the UDP or TCP port ranges for matching the IP flow. If this element is omitted and the sourcePortStart is included, then its assumed value should be the same as the value of sourcePortStart. If this element is omitted and the sourcePortStart is omitted, then its assumed value should be 65535.
- The elements trafficClassMask, trafficClassLow, trafficClassHigh identify the values of the traffic class field in the IP header that match this classifier. If trafficClassMask is omitted, it is assumed to have a value of 0. If trafficClassMask is 0 or unspecified, trafficClassHigh and trafficClassLow **MUST NOT** be specified. If trafficClassLow is omitted, and trafficClassMask has a non-zero value, trafficClassLow is assumed to have a value of 0. If trafficClassHigh is omitted, and trafficClassMask has a non-zero value, trafficClassHigh is assumed to have a value of 255.
- The element flowLabel identifies the IPv6 flow label value that the classifier matches. If this element is omitted, comparison of the IPv6 flow label is irrelevant for this entry.

6.2.1.2 ContextID

The ContextID element is the identifier that associates the resources of a given session. The ContextID element consists of a baseId element and optional idExtension elements. The same ContextID or an extended version of the same ContextID (extended via new idExtensions) **MUST** be used in subsequent resource request messages when resources are modified after the initial request.

The AS **MAY** assign a ContextID for a context it creates by providing a value for the ContextID element for the network resources request. If the AS chooses to provide the ContextID, it **MUST** do so such that the ContextID is guaranteed to be unique at a given time across all Application Servers that use the same AS Username. If the AS chooses to provide the ContextID, it **MUST** format the element as defined in Annex A. The AS **MAY** choose to create optional idExtension elements as described in 6.2.1.2.1. When using optional idExtension elements, the ContextID so created must also be unique; uniqueness of ContextIDs with optional idExtensions is defined in the following section. If the AS chooses to provide the ContextID, it **SHOULD** ensure ContextIDs created are not reused for a reasonable period of time (e.g., weeks) after the context corresponding to them no longer exist.

If the AS provided a ContextID value to the AM, the AM **MUST** use this ContextID and **MUST NOT** overwrite or replace a ContextID value submitted to it by the AS.

The AM **MUST** assign a ContextID for the requested context if the AS did not provide a ContextID value in the network resource request. If the AM assigns the ContextID, it **MUST** format the element as defined in Annex A. The AM **MUST** do so such that the ContextID so created is guaranteed to be unique at a given time across all Application Servers that use a particular AS Username value. If the AM assigns the ContextID, it **MAY** choose to create optional idExtension attributes. When using optional idExtension attributes, the ContextID so created must also be unique; uniqueness of ContextIDs with optional idExtensions is defined in the following section. If the AM chooses to provide the ContextID, it **SHOULD** ensure that ContextID values created are not reused for a reasonable period of time (e.g., weeks) after the context corresponding to them no longer exist.

The AM **SHOULD** treat the combination of the AS Username (provided in SOAP header as per [WSS]) and ContextID service attributes as the unique identifier for a context.

6.2.1.2.1 Context ID: Uniqueness and Use in Identifying sessions

This section defines how to identify sessions by the ContextID element and defines uniqueness for the ContextID. The capability of the AS to provide the ContextID is useful for the stateless rendezvous protocol case of Section 5.2.1. For both the stateless and stateful rendezvous protocol cases, a multiplicity of potential contexts may be created that are associated with the same originating request due to forking of the original request. Such a case is shown below in Figure 5.

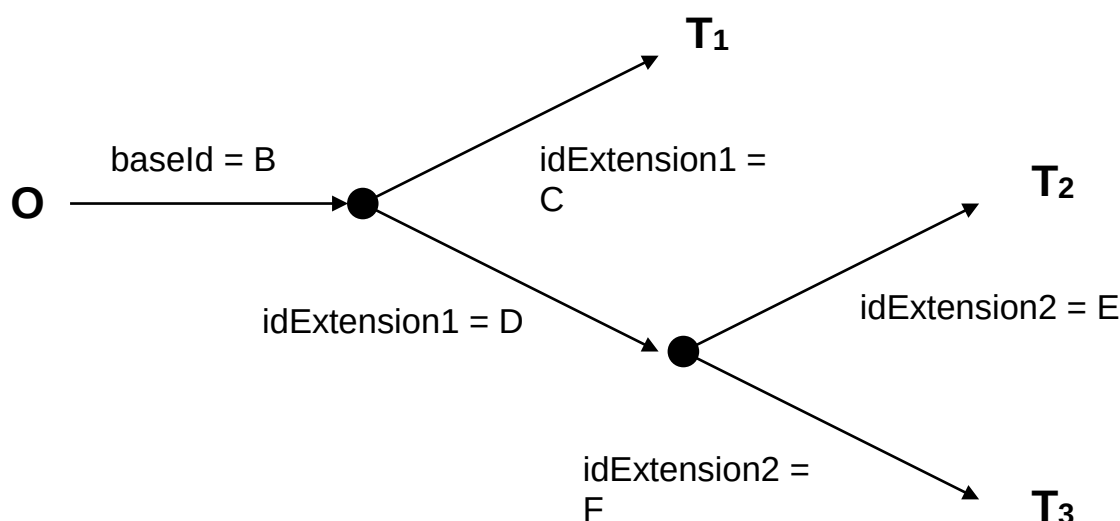


Figure 5 - ContextID creation

In this figure, a session request from an originating end client ("O") was forked to three potential terminating clients ("T₁", "T₂" and "T₃"). The idExtension elements of the ContextID are used to differentiate the legs for such cases. For example the session leg to T₂ is {B,D,E}. In this notation the elements in the brackets denote the ContextID and is of the form ContextID = {baseId, idExtension1, idExtension2, ...}. Likewise, the session leg to T₃ is {B,D,F} and the session leg to T₁ is simply {B,C}. Note that although some legs have a different number of idExtension elements, all three ContextID attributes share the same baseId element, and are unique. The AM can therefore associate all the different ContextIDs for these three session legs with the same outbound originating request by the baseId element.

For the case where the AM received the ContextID from the AS, the AM MUST perform accounting of the individual session legs and perform required operations for each leg individually (e.g., reservation timeouts) and associate forked requests that have originated from the same originating end client request (by virtue of such requests having the same baseId).

The AS MUST perform this accounting dynamically for those legs added by virtue of additional optional-id attributes being added to previously defined ContextIDs in subsequent web services operations.

The ContextID is considered unique when the combination of the base-id attribute together with the same number of optional id-extension attributes is unique. Whenever a new branch is formed, a new level of id-extension attribute MUST be created so that the new ContextID has the same BaseID and idExtension attributes of the previous level. Whenever a new leaf is formed from an existing branch, a new level of idExtension MUST NOT be created and a new, unique idExtension at the existing level MUST be created.

6.2.1.2.2 ContextID: Wildcarding operations

As all sessions are uniquely associated via the ContextID and the WS messages described later will refer to the ContextID, it is useful to define wildcarding operation on the ContextID here. Using the notation where the ContextID = {baseId, idExtension1, idExtension2, ...} the wildcarding operation is highly dependent on where the wildcard is placed. Deleting resources for contentId = {B,D,*} for the case above will delete resources for T₂ and T₃, while leaving resources for T₁ unaffected. Deleting resources for {B,*} will delete all resources for this context. Deleting resources for {*} will delete all resources for a given SubscriberID (and given AS Username and serviceName) and should normally only be invoked by administrative operations.

Therefore, the AS SHOULD normally use wildcarding of the form {B,*} to affect a web services operation on all sessions with baseId = B (i.e., the * wildcard matches on one, many or null). Note that ContextID = {B} does not match the ContextID for any of the call legs in the example above, as the ContextID is defined as including all optional idExtensions defined up to the present. In the above example, ContextID = {B} may have at one time existed prior to the ContextID being extended by optional idExtensions.

Considering optional idExtension elements, wildcarding of the form {B,*} is highly RECOMMENDED for all AS operations on all sessions with baseId = {B} (i.e., even for applications that do not use optional-id elements).

6.2.1.3 ContextInfo

An array of ContextInfo elements is returned in the QueryContexts response message (described in 6.3.5.1). The ContextInfo element contains the ContextID element, described in 6.2.1.2, and the element ContextStatus, which identifies the status of the associated IPCablecom Multimedia gate for the session. A context can be in a reserved or committed state.

The AM MUST format the ContextInfo element as defined in Annex A.

6.2.1.4 QoSChangeEvent

The QoSChangeEvent element describes the change in state for a flow of a session. QoSChangeEvent is present in notification messages sent to ASes which have subscribed to the state of the resources. The AM MUST format the QoSChangeEvent element as defined in Annex A.

The QoSChangeEvent element contains a direction element indicating whether this change is for the upstream / downstream flows and consists of two sub elements.

The QoSChangeType element indicates the type of change being signaled. This attribute is an enumeration, the AM MUST indicate one of the following states:

- 1 = Idle/Closed
- 2 = Authorized
- 3 = Reserved
- 4 = Committed
- 5 = Committed-Recovery

Reason is a 2-byte unsigned integer field, which MUST indicate one of the following reasons for this update:

- 1 = Close initiated by CMTS due to reservation reassignment
- 2 = Close initiated by CMTS due to lack of DOCSIS MAC-layer responses