Internet Draft
Expiration: December 2000
File: draft-durham-aaa-cops-ext-00.txt

David Durham
Hormuzd Khosravi
Intel
Walter Weiss
Lucent
Avri Doria
Nokia

COPS Usage for AAA

Last Updated: May 31, 2000

Status of this Memo

Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC-2119].

Abstract

This document describes how COPS can be used as a general purpose Authentication, Authorization, and Accounting (AAA) Protocol.

1. Introduction

This document describes how the COPS model can be extended from the client-server query-response usage to a brokered or proxied model as well. The basic model of interaction between a policy server and its clients is described within the framework document for policy based admission control [WRK].

A chief objective of policy control protocol is to begin with a simple but extensible design. The key features of COPS that are useful for the AAA Framework are as follows:

1. Simple, Lightweight Design: COPS is designed as a simple client/server protocol. It is thus lightweight and is very easy to implement.

2. Reliability: COPS uses TCP as its transport protocol for reliable exchange of messages between clients and a server. Therefore, no additional mechanisms are necessary for reliable communication. Other reliable transport protocols can also be used to carry COPS messages as the protocol is defined independently from the underlying transport.

3. Extensibility: The protocol is extensible in two ways. One, it is designed to leverage off self-identifying objects and can support diverse client specific information without requiring modifications to the COPS protocol itself. Two, the provisioning extensions define a generalized mechanism for communicating well-defined data between the client and server for the general administration, configuration, and enforcement of policies whether signaled or provisioned. Thus, the protocol may be extended for the administration of a variety of signaling protocols as well as the configuration of a device.

4. Security: COPS provides three security mechanisms. First, COPS directly provides message level security for authentication, replay protection, and message integrity common to all COPS implementations. COPS can also reuse existing protocols for security such as IPSEC [IPSEC] or TLS [COPSTLS] to authenticate and secure the communication between the PEP and the PDP. Finally, COPS can carry CMS [COPSCMS] enveloped objects

for end-to-end confidentiality, integrity, and non-repudiation.

5. Stateful Model: Clients are able to instantiate state within their server corresponding to their requests, and the client may

update or remove this state at any time. Additionally, the protocol is stateful in that it allows the server to respond to such requests with a decision or directive, and allows the server to change its decisions at any time while the state remains installed.

6. Fail Over: COPS provides a fast failover mechanism. It essentially uses keep-alive messages between the client and server to verify the connection. When a failure is detected, the client must try to reconnect to the remote server or attempt to connect to a backup/alternative server. Once a connection is reestablished, the internal state of the client can be resynchronized with the server only if needed.

1.1 COPS-AAA Model:

The COPS protocol supports two common models for policy control: Outsourcing and Provisioning. The Outsourcing Model is client driven where the client requests authorization from a remote server. COPS-PR [COPSPR] or the Provisioning Model is used by the client to request that the policy server proactively provision the policy client with its directives. Provisioning refers to the ability to issue directives to a device in a stateful manner such that each directive can be monitored, updated, or withdrawn as appropriate to the situation. Outsourcing refers to the ability of the client to issue multiple requests to receive authorization to use network resources or services and/or authenticate itself or a user. In the AAA model, COPS clients make one or more requests to their servers using COPS-PR defined data, and servers likewise respond with COPS-PR defined decisions depending on the type of request.

The COPS-PR Model is very suitable for the AAA framework. One of the main advantages that COPS-PR has is that it decouples the Information or Data Model from the protocol. For Example, COPS-PR uses PIBs or Policy Information Bases that define the Information Model for QoS Policy Provisioning among others. PIB data can be used to specify the information used by a client in its requests and accounting reports as well as by a server in its decisions. This makes the Model extensible and very flexible while using structured data. COPS supports the different COPS-PR client-types as non-overlapping and independent namespaces. Thus, the Information Model can be defined for different areas of policy provisioning (e.g. security) and authorization (e.g. NAS-AAA) while sharing the same underlying COPS protocol.

In COPS, each request-state identified by the COPS client-handle

corresponds to an individual AAA session. As each client-handle in the COPS-PR model represents a non-overlapping and independent namespace, each session can be dealt with independently.

There are also many instances where AAA clients defer to servers for decision support, or for requesting the additional information necessary for completing a process such as a dialup access request (login). In these scenarios, COPS is ideally suited for this paradigm, particularly since COPS was initially designed to meet this specific objective. No only can it provide Yes/No decisions to client requests, using its Information Model, sophisticated directives can be issued (such as allow access but only for 30 minutes, or specify the list of services available to the user).

1.2 Information Model:

An Information Model is a formalization of the data structures used to configure and manage network services, request access to services, and report service activity. Information Models have their origins in OO design and take advantage of all the capabilities inherent to Object Oriented principles. Of the capabilities provided within the OO framework, those most relevant to any AAA protocol (and COPS in particular) are inheritance, containment, and associations.

Inheritance and containment both provide a means for reusing various data structures. Inheritance provides the means for sharing those attributes common to many components while also allowing for the refinement of management interfaces that are specific to a given implementation. In some sense, inheritance is another way of expressing specific refinements or extensions.

Consider, for example, those attributes common to all authentication protocols. These attributes would be specified in the AuthenticationService class that forms the root for all authentication related management interfaces. This class may be refined (derived) in two peer classes: PapAuthenticationService and ChapAuthenticationService. Both derived classes include all the attributes defined in the AuthenticationService class. However, the ChapAuthenticationService class differs from the PapAuthentication class with respect to the PAP or CHAP specific attributes. The PAP class may, in turn, be derived to specify vendor or product specific extensions to PAP. The result is a tree of class definitions with very common, but very abstract interface attributes at the root of the tree and very implementation specific attributes at the leaves of the tree.

This ability to add refinement within a hierarchy of abstractions makes interface management far more flexible than it is with current management structure definitions while

maintaining interoperability. When advances in authentication technologies occur, the tree can be extended at the root, the limbs or the leaves of the tree depending on the degree to which these new technologies are different from those already defined.

Containment refers to the ability to embed specific types of structures within other structures. One could consider containment the ability to embed new complex data types within a class. Containment is similar to Hierarchies/Derivations in that the attributes of a contained class and superclass (parent of the hierarchy) both have the same life span as the attributes in the container or subclass. However, the difference between containment and inheritance is that a contained class reuses an unrelated data structure while an inherited class is a refinement of a related data structure. To continue the previous example, a PAP or CHAP class may contain an IP address class that describes the characteristics of an IP address (the mask, the class, whether it is multicast, etc.) Clearly an IP address is not a refinement of an AuthenticationService. However it is a complex or reusable structure that may be necessary to describe the AuthenticationService.

Associations represent the relationships between various independent structures. Unlike container class instances, which only exist as long as the contained class instance, associatied classes can exist independently of each other. For example, an IP address specified in an AuthenticationService is only relevant as long as an instance of the AuthenticatoinService exists. However, a user using the AthenticationService can exist irrespective of whether the AuthenticationService is active or not. Therefore, if it is appropriate to bind an AuthenticationService to a User, an association is the most appropriate means for capturing this relationship. Note that while associations are not necessary to any single transaction, it is potentially relevant to determining related, subsequent transactions. The scenarios for using associations are comparable to those that use RowPointers in MIBs.

Reusability is a key element to an Information Model. The ability to define a User that is consistent not only within a space like AAA, but also Security, QoS, address management, to name a few, is critical to the consistent administration of users and ability to allow various technologies such as AAA and QoS to operate collaboratively.

The Information or Data Model that is used with COPS-PR supports the AAA framework. The concept of a data model is a common data representation that fully describes a given set of standard AAA data structures and relationships. As additional data can be described using the underlying data representation, extensions to the data model are simple to add. Fundamentally, three sets of class namespaces can be defined for Authentication,

Authorization and Accounting respectively. The existing work done in defining Accounting Attributes can be used to define the accounting Model. The NASREQ, ROAMOPS and MOBILEIP working groups can jointly define their respective parts of the Model, such that it satisfies all of their data model requirements.

COPS simply provides the reliable transport semantics (Request, Decision, Report) and security mechanisms (native, TLS, and CMS) for communicating these respective parts of the model.

The Policy Information Base or PIB which is used for QoS Policy Provisioning uses SMI [V2SMI] for its data representation language and BER [BER] for data encoding. These useful mechanisms were adopted so that the experience, tools and code from the network management community can be reused. A similar approach can be used for the AAA framework but this is definitely not a restriction. COPS-PR supports different encoding schemes, for example XML string based encoding, by using the S-Type field in the COPS-PR protocol objects. Additionally, support for using the ADIF[] format within the encapsulating COPS base protocol objects can be defined as well as RADIUS AVPs among others as is appropriate for AAA.

2. COPS Broker/Proxy Support

COPS can support both the different types of brokers mentioned in the AAA framework i.e. Proxy Broker and Routing Broker.

Proxy Broker:

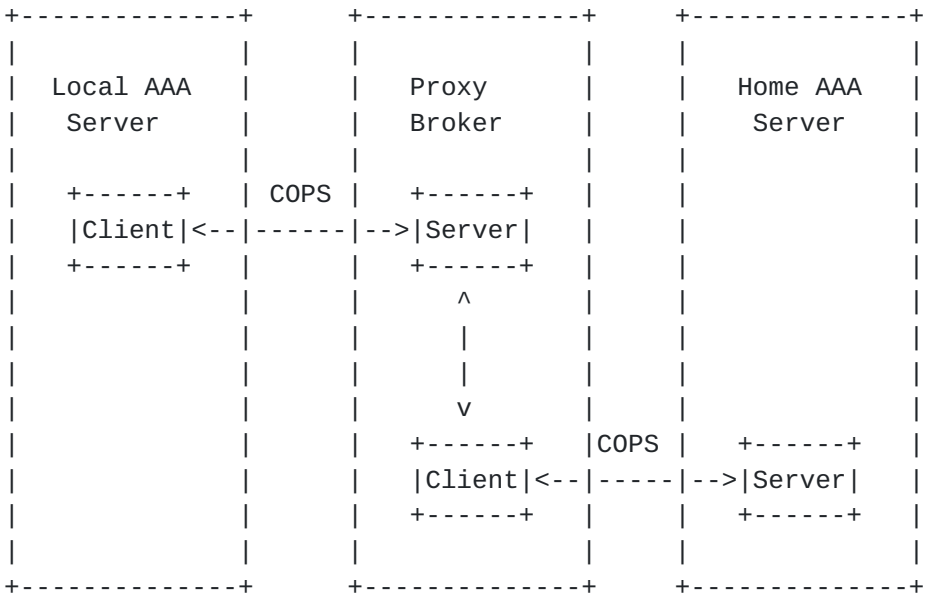


Figure 1: A COPS Proxy illustration.

A COPS Proxy Broker is similar to a transparent proxy and essentially performs routing functions. As shown in Figure 1, the proxy broker forwards a COPS Request from the Local AAA

Server to the Home AAA server. It acts as the COPS Server to receive the request at the Local Server and as the COPS Client to send the request to the Home Server.

It can also act like the LDP i.e. local decision point. The Proxy Broker can perform authentication for the COPS Request from the local server based on local policies. It can reject the request and send a COPS Decision message with the appropriate Error Object. The use of Proxy brokers reduces the amount of configuration information maintained on the AAA servers.

Information from the client that is intended for the home server only or information from the home domain intended for the client can be protected using CMS. CMS provides end-to-end security by protecting the COPS-PR objects within a COPS message. CMS can provide authentication, integrity, and confidentiality on an object-by-object basis.

This model can be extended to multiple brokers.

Redirect Broker:

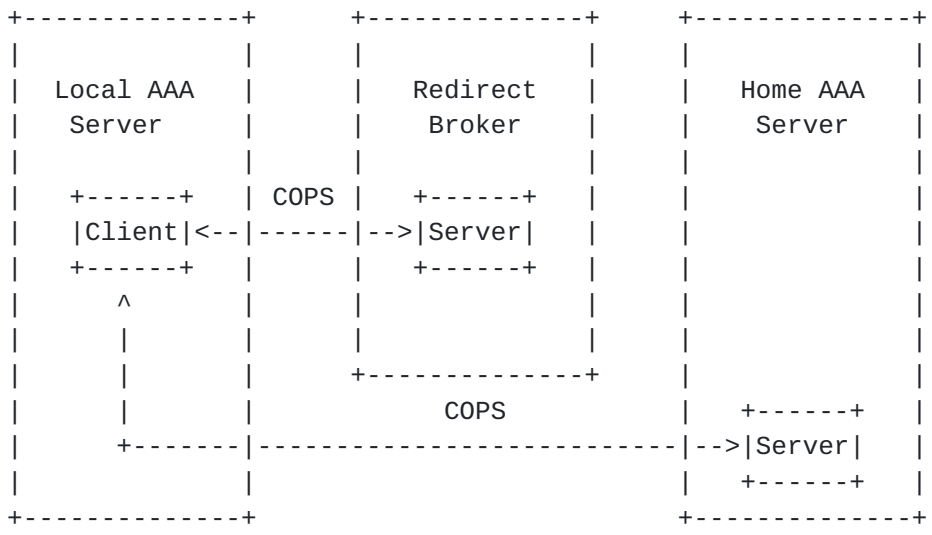


Figure 2: A COPS Redirect illustration.

A COPS Redirect broker performs redirect functions so that the AAA servers can communicate directly. As shown in Figure 2, when the Redirect broker receives a COPS Request from the Local AAA Server it sends back a Decision message with the "Redirect Request" command code (see section 3). The redirection information is part of the Data Model and is carried in the Decision message. The Local Server can then directly communicate with the Home Server.

The Redirect Broker also reduces the configuration information to be maintained on all the AAA servers.

As with the proxy mode, information from the client that is intended for the home server only or information from the home

domain intended for the client can be protected using CMS, or the COPS connection itself can be protected using TLS.

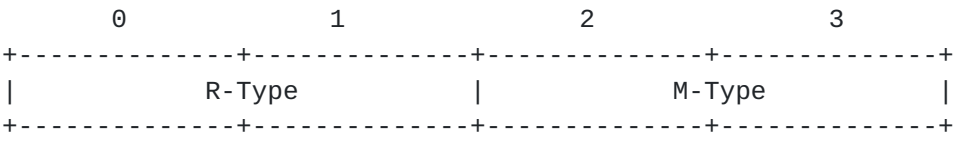
3. COPS Specific Object Formats

In this section, we define new C-Types and other enhancements for some COPS protocol objects that will be useful for the AAA framework.

3.1 Context Object (Context)

A new R-Type is added to the context object for identifying an authentication request form a client to its server.

C-num = 2, C-Type = 1



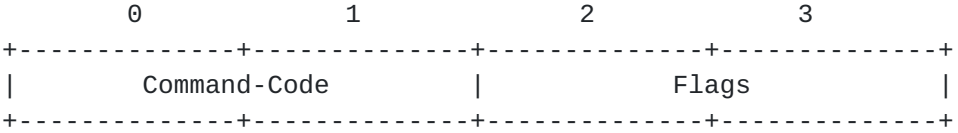
R-Type (Request Type Flag)

0x01 = Incoming-Message/Admission Control request
0x02 = Resource-Allocation request (Authorization)
0x04 = Outgoing-Message request
0x08 = Configuration request
0x10 = Authentication request

3.2 Decision Object (Decision)

Decision made by the PDP. Appears in replies from the server. The specific non-mandatory decision objects required in a decision to a particular request depend on the type of client.

C-Num = 6
C-Type = 1, Decision Flags (Mandatory)



Commands:
0 = NULL Decision (No configuration data available)
1 = Install (Admit request/Install configuration)

2 = Remove (Remove request/Remove configuration)
3 = Trigger Request
4 = Redirect Request
5 = Trigger Accounting Report

The Trigger Request command code is used when a COPS server wants the client to resend a Request message. This is used to for both re-authentication and re-authorization on demand within the AAA model.

The Redirect Request command code is used by a COPS server acting as a Redirect Broker to indicate that the client must redirect its Request message to another COPS server.

The Trigger Report is used by the server to force an accounting report to be issued by the client for the session.

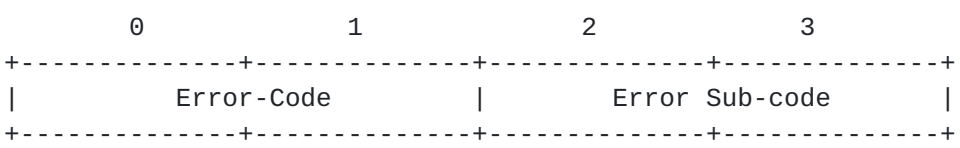
C-Type = 6, CMS Named Decision Data

Named configuration information is encapsulated in this version of the decision object in response to CMS secured requests. It is a variable length object and its internal format is specified in the CMS over COPS extension document [COPSCMS] and by [COPSPR]. It is optional in Decision messages and is interpreted relative to both the given context and decision flags.

3.3 Error Object (Error)

This object is used to identify a particular COPS protocol error.

C-Num = 8, C-Type = 1



Additional Error-Code:

16= CMS Authentication Failure [COPSCMS]

3.4 Client Specific Information Object (ClientSI)

The various types of this object are required for requests, and used in reports and opens when required. It contains client-type specific information.

C-Num = 9,

C-Type = 1-2, Defined by COPS

C-Type = 3, CMS Named ClientSI.

Variable-length field. Contains named configuration information protected by CMS [COPSCMS] and its format is defined by [COPSPR]. This encapsulating object is used for relaying structured

information about the PEP, a request, or configured state securely, end-to-end.

3.5 Report-Type Object (Report-Type)

The Type of Report on the request state associated with a handle:

C-Num = 12, C-Type = 1

Additional Report-Type:

4 = Acknowledged Accounting:
Accounting update for an installed state that is to be explicitly acknowledged by the server by issuing a Decision that specifies the received accounting information is to be removed by the client.

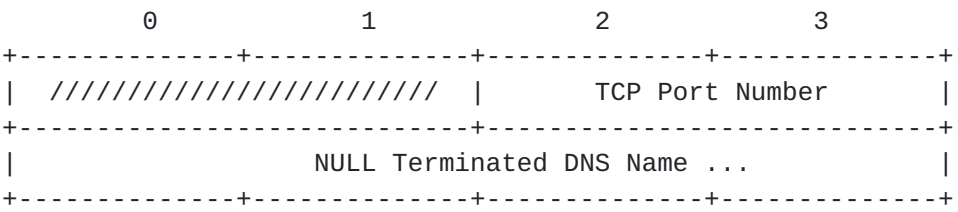
3.6 PDP Redirect Address (PDPRedirAddr)

A PDP when closing a PEP session for a particular client-type may optionally use this object to redirect the PEP to the specified PDP server address and TCP port number:

C-Num = 13,

C-Type = 1-2, Defined in [COPS]

C-Type = 3, DNS Address + TCP Port



3.7 Last PDP Address (LastPDPAddr)

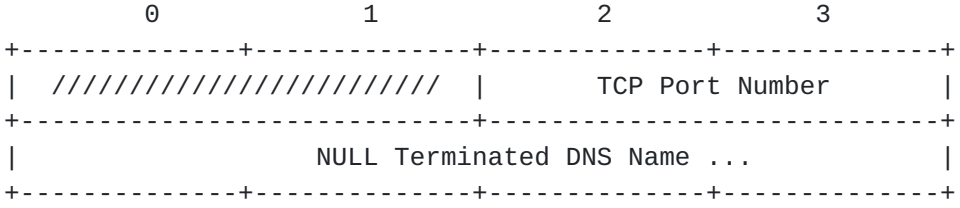
When a PEP sends a Client-Open message for a particular client-type the PEP SHOULD specify the last PDP it has successfully opened (meaning it received a Client-Accept) since the PEP last rebooted.

If no PDP was used since the last reboot, the PEP will simply not include this object in the Client-Open message.

C-Num = 14,

C-Type = 1-2, Defined in [COPS]

C-Type = 3, DNS Address + TCP Port (Same as Redirect Address)



4. COPS-PR Defined Objects

The COPS Policy Provisioning [COPSPR] clients encapsulate several objects within the existing COPS Named Client-specific information object and Named Decision Data object. All the object descriptions and examples in this document require the Basic Encoding Rules as the encoding type (S-Type = 1).

The COPS-PR objects are used to communicate data defined within a Policy Information Base which constitutes the COPS-PR information model.

CMS over COPS [COPSCMS] defines how these BER-encoded COPS-PR objects are secured in the CMS Named Client-Specific Information object and the CMS Named Decision Data object. CMS provides end-to-end security at the object level for sensitive data.

In the AAA use of COPS-PR, the COPS-PR client-handle is to be treated as a session identifier. Each client-handle represents a unique context or non-overlapping and independent namespace initiated by the client to request authentication of a user and/or authorization for the use of network resources. Each Decision corresponds to its respective client-handle and provides a BER encoded response to the client's request.

5. Common Operation for AAA

COPS for AAA uses COPS-PR defined data in its request for authentication or authorization of resources. Similarly, Decision messages use COPS-PR defined data to issue directives to the client based on its request. Likewise, accounting information is to use the COPS-PR defined data and is communicated using Report messages to describe session activity.

1. The Client will open a COPS connection to its server using either the default COPS security mechanisms or COPS over TLS [COPSTLS].

2. The Client will then send a Client-Open message for the AAA Client-Type.

3. At any time, the Client can issue a Request identified by its client-handle carrying COPS-PR defined data relating to user authentication and/or authorization.
4. The Server analyzes the client's request and determines if the request can be handled locally or not.
   a. If the request can be handled locally, the server responds with a COPS decision accepting/rejecting the request or carrying other COPS-PR defined directives for the client.
   b. If the request cannot be handled locally, the server will either redirect the client for the given client-handle to the appropriate server via a redirect decision or will proxy the request to the home domain on behalf of the client.
5. The client will abide by the server's decision:
   a. If the decision is negative, the client must delete the request-state or attempt its request again.
   b. If the decision was positive, the client must abide by the directives and limitations specified by the server.
6. Once a request-state has been accepted by the server, the client MAY send periodic accounting reports specifying the current state of this AAA session according to the server's directive.
7. At any time, the client can re-authenticate or reauthorize its request state by simply sending an updated request message for the same client-handle.
8. At any time, the server may change its decision for the request state by simply sending an updated decision message.
9. At any time, the server may direct the client to re-authenticate its session.
10. At any time, the server may direct the client to send an accounting report for its session.
11. At any time, the client can delete its request state.
12. The client can close its COPS connection by sending a Client-Close message and closing the underlying TCP connection.

6. Security Considerations

6.1 Hop-by-Hop Security :

The COPS protocol provides an Integrity object that can achieve authentication, message integrity, and replay prevention. All COPS implementations MUST support the COPS Integrity object and its mechanisms as described [COPS]. To ensure the client (PEP) is communicating with the correct policy server (PDP) requires authentication of the PEP and PDP using a shared secret, and consistent proof that the connection remains valid. The shared secret minimally requires manual configuration of keys (identified by a Key ID) shared between the PEP and its PDP. The key is used in conjunction with the contents of a COPS message to calculate a message digest that is part of the Integrity object. The Integrity

object is then used to validate all COPS messages sent over the TCP connection between a PEP and PDP.

The COPS Integrity object also provides sequence numbers to avoid replay attacks. The PDP chooses the initial sequence number for the

PEP and the PEP chooses the initial sequence number for the PDP. These initial numbers are then incremented with each successive message sent over the connection in the corresponding direction. The initial sequence numbers SHOULD be chosen such that they are monotonically increasing and never repeat for a particular key.

Additionally, in support of AAA, Transport Layer Security [COPSTLS] SHOULD be used for both connection-level validation and privacy. All AAA implementations using COPS MUST be capable of supporting TLS.

6.2 End-to-End Security:

COPS for AAA requires the use of CMS for object-level end-to-end security of named data [COPSCMS]. CMS can be used to sign and/or encrypt the appropriate PIB data such that it cannot be altered, undetected, by AAA proxies or other man-in-the-middle components. What data is to be protected, for which recipients, and how is defined within the information model or PIB for AAA.

7. IANA Considerations

8. References

[COPS] Boyle, J., Cohen, R., Durham, D., Herzog, S., Rajan, R., Sastry, A., "The COPS (Common Open Policy Service) Protocol", RFC 2748, August 1999.

[COPSPR] Reichmeyer, F., Herzog, S., Chan, K.H., Seligson, J., Durham, D., Yavatkar, R., Gai, S., McCloghrie, K., Smith, A., "COPS Usage for Policy Provisioning", IETF , October 1999.

[COPSCMS] Walker J., "CMS Over COPS", IETF , June 2000.

[COPSTLS] Walker J., "COPS Over TLS", IETF , June 2000.

[RSVP] Braden, R. ed. et al., "Resource ReSerVation Protocol (RSVP) Version 1 - Functional Specification", RFC 2205, September 1997.

[WRK] Yavatkar, R. et al., "A Framework for Policy-Based Admission Control", Internet-Draft, draft-ietf-rap-framework-01.txt, November 1998.

[SRVLOC] Guttman, E. et al., "Service Location Protocol , Version 2", RFC 2608, June 1999.

[IPSEC] Atkinson, R., "Security Architecture for the Internet Protocol", RFC 2401, August 1995.

[HMAC] Krawczyk, H., Bellare, M., Canetti, R., "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, February 1997.

[MD5] Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, April 1992.

[TLS] Dierks T., Allen C., "The TLS Protocol Version 1.0", RFC 2246, January 1999.

[IANA] http://www.isi.edu/in-notes/iana/assignments/port-numbers

[IANA-CONSIDERATIONS] Alvestrand, H. and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 2434, October 1998.

9. Author Information and Acknowledgments

Special thanks to Keith McCloghrie for his valuable contributions.

Avri Doria
Nokia

5 Wayside Road
Burlington MA 01803
+1 781 993 4645
avri.doria@nokia.com

David Durham
Intel
2111 N.E. 25th Avenue JF3-206
Hillsboro OR 97124-5961
1 503 264 6232
david.durham@intel.com

Hormuzd M Khosravi
Intel
2111 N.E. 25th Avenue JF3-206
Hillsboro OR 97124-5961
1 503 264 0334
hormuzd.m.khosravi@intel.com

Walter Weiss
Lucent Technologies
300 Baker Ave.
Concord, MA. 01742
(978) 287-9130
wweiss@lucentctc.com