Network Working Group
Request for Comments: 4712
Category: Standards Track

A. Siddiqui
D. Romascanu
Avaya
E. Golovinsky
Alert Logic
M. Rahman
Samsung Information Systems America
Y. Kim
Broadcom

October 2006

Transport Mappings for Real-time Application Quality-of-Service Monitoring (RAQMON) Protocol Data Unit (PDU)

Status of This Memo

Copyright Notice

Abstract

This memo specifies two transport mappings of the Real-Time Application Quality-of-Service Monitoring (RAQMON) information model defined in RFC 4710 using TCP as a native transport and the Simple Network Management Protocol (SNMP) to carry the RAQMON information from a RAQMON Data Source (RDS) to a RAQMON Report Collector (RRC).

Table of Contents

1. Introduction

The Real-Time Application QoS Monitoring (RAQMON) Framework, as outlined by [RFC4710], extends the Remote Monitoring family of protocols (RMON) by defining entities such as RAQMON Data Sources RDS) and RAQMON Report Collectors (RRC) to perform various application monitoring in real time. [RFC4710] defines the relevant metrics for RAQMON monitoring carried by the common protocol data unit (PDU) used between a RDS and RRC to report QoS statistics. This memo contains a syntactical description of the RAQMON PDU structure.

The following sections of this memo contain detailed specifications for the usage of TCP and SNMP to carry RAQMON information.

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 [RFC2119].

2. Transporting RAQMON Protocol Data Units

The RAQMON Protocol Data Unit (PDU) utilizes a common data format understood by the RDS and the RRC. A RAQMON PDU does not transport application data but rather occupies the place of a payload specification at the application layer of the protocol stack. As part of the specification, this memo also specifies the usage of TCP and SNMP as underlying transport protocols to carry RAQMON PDUs between RDSs and RRCs. While two transport protocol choices have been provided as options to chose from for RDS implementers, RRCs MUST implement the TCP transport and MAY implement the SNMP transport.

2.1. TCP as an RDS/RRC Network Transport Protocol

A transport binding using TCP is included within the RAQMON specification to facilitate reporting from various types of embedded devices that run applications such as Voice over IP, Voice over Wi-Fi, Fax over IP, Video over IP, Instant Messaging (IM), E-mail, software download applications, e-business style transactions, web access from wired or wireless computing devices etc. For many of these devices, PDUs and a TCP-based transport fit the deployment needs.

The RAQMON transport requirements for end-to-end congestion control and reliability are inherently built into TCP as a transport protocol [RFC793].

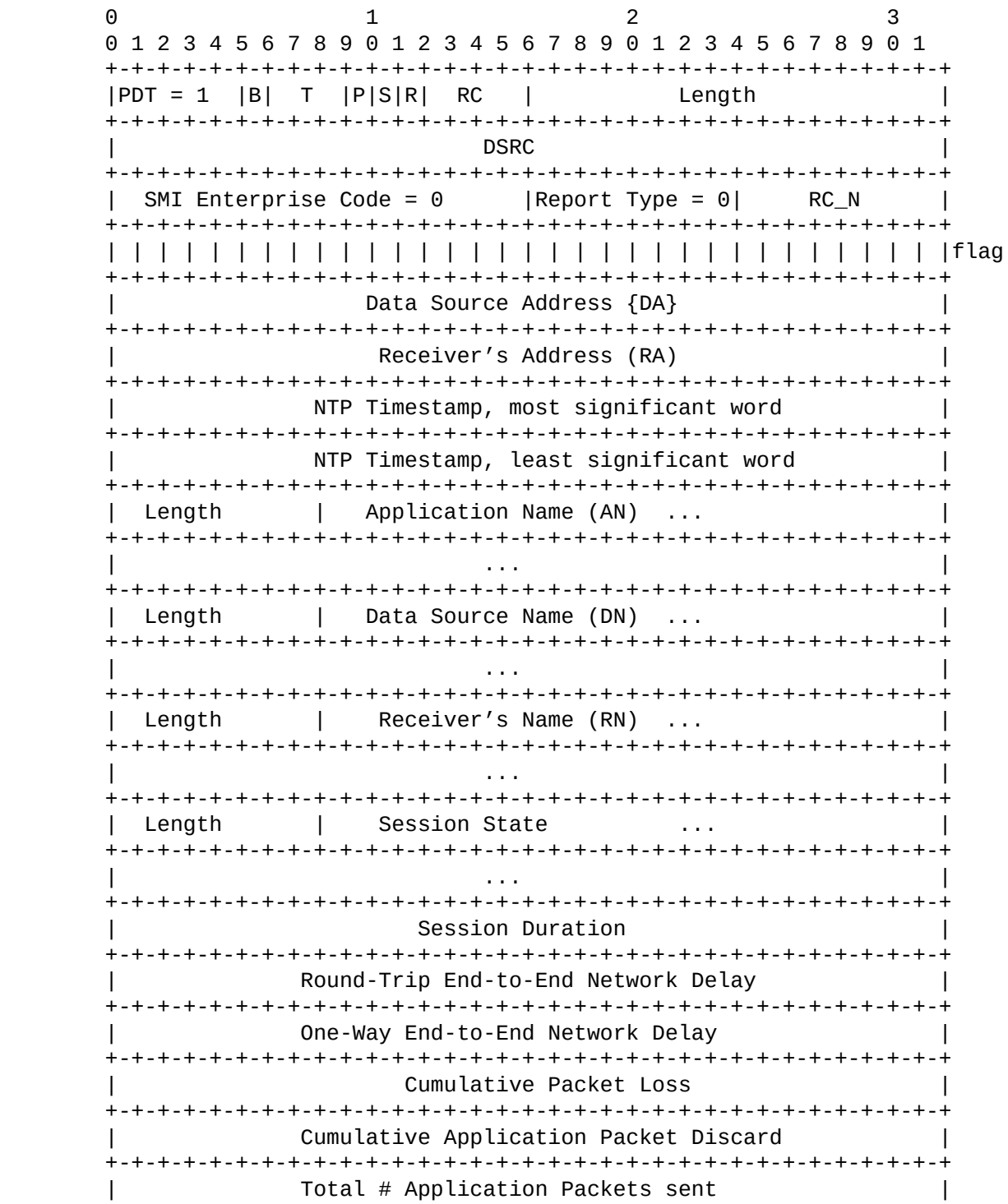
To use TCP to transport RAQMON PDUs, it is sufficient to send the PDUs as TCP data. As each PDU carries its length, the receiver can determine the PDU boundaries.

The following section details the RAQMON PDU specifications. Though transmitted as one Protocol Data Unit, a RAQMON PDU is functionally divided into two different parts: the BASIC part and application extensions required for vendor-specific extension [RFC4710]. Both functional parts follow a field carrying a SMI Network Management Private Enterprise code currently maintained by IANA http://www.iana.org/assignments/enterprise-numbers, which is used to identify the organization that defined the information carried in the PDU.

A RAQMON PDU in the current version is marked as PDU Type (PDT) = 1. The parameters carried by RAQMON PDUs are shown in Figure 1 and are defined in section 5 of [RFC4710].

Vendors MUST use the BASIC part of the PDU to report parameters pre-listed here in the specification for interoperability, as opposed to using the application-specific portion. Vendors MAY also use application-specific extensions to convey application-, vendor-, or device-specific parameters not included in the BASIC part of the specification and explicitly publish such data externally to attain extended interoperability.

2.1.1. The RAQMON PDU



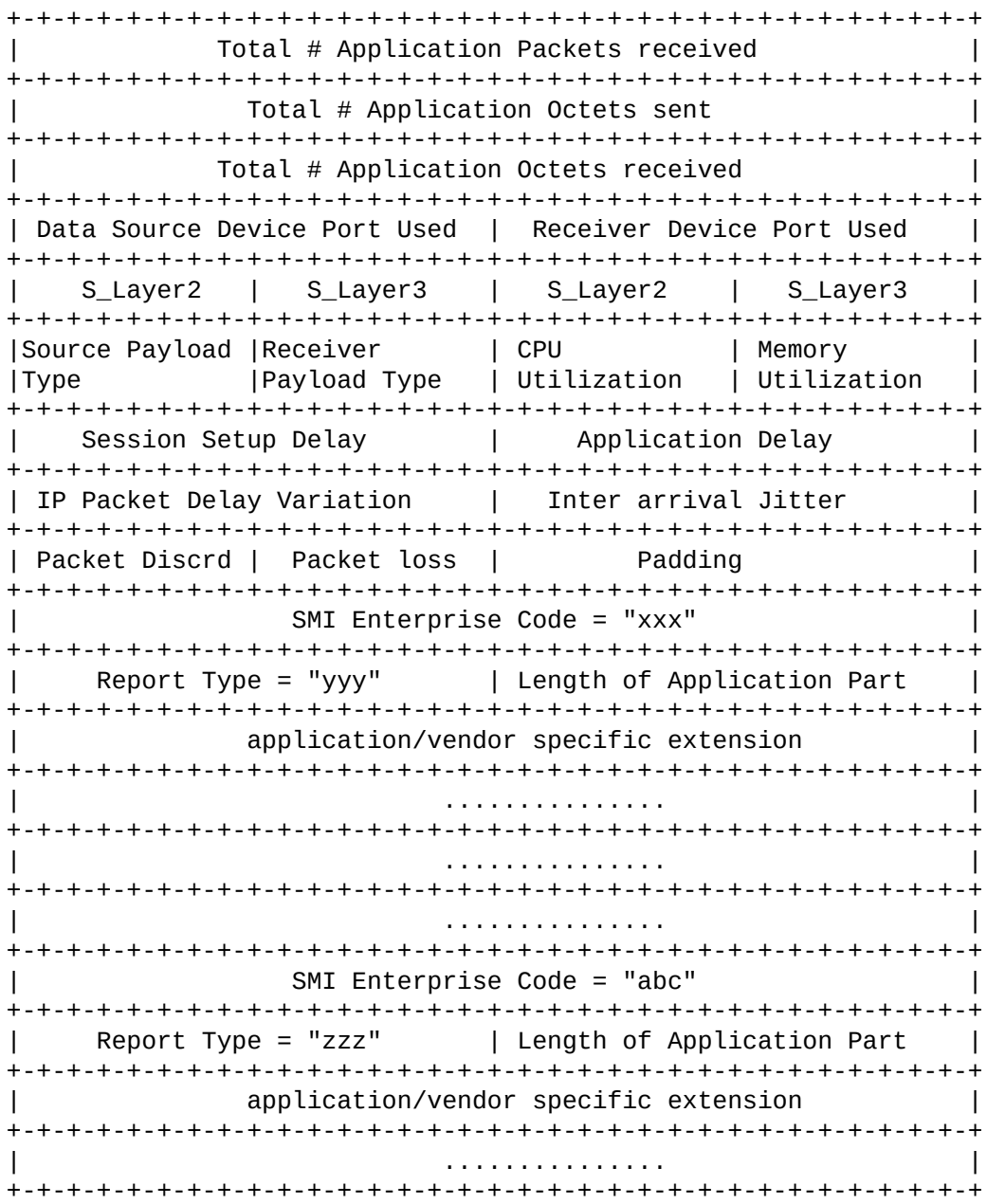


Figure 1: RAQMON Protocol Data Unit

2.1.2. The BASIC Part of the RAQMON Protocol Data Unit

A RAQMON PDU must contain the following BASIC part fields at all times:

PDU type (PDT): 5 bits - This indicates the type of RAQMON PDU being sent. PDT = 1 is used for the current RAQMON PDU version defined in this document.

basic (B): 1 bit - While set to 1, the basic flag indicates that the PDU has BASIC part of the RAQMON PDU. A value of zero is considered valid and indicates a RAQMON NULL PDU.

trailer (T): 3 bits - Total number of Application-Specific Extensions that follow the BASIC part of RAQMON PDU. A value of zero is considered valid as many times as there is no application-specific information to add to the basic information.

padding (P): 1 bit - If the padding bit is set, the BASIC part of the RAQMON PDU contains some additional padding octets at the end of the BASIC part of the PDU that are not part of the monitoring information. Padding may be needed in some cases, as reporting is based on the intent of a RDS to report certain parameters. Also, some parameters may be reported only once at the beginning of the reporting session, e.g., Data Source Name, Receiver Name, payload type, etc. Actual padding at the end of the BASIC part of the PDU is 0, 8, 16, or 24 bits to make the length of the BASIC part of the PDU a multiple of 32 bits

Source IP version Flag (S): 1 bit - While set to 1, the source IP version flag indicates that the Source IP address contained in the PDU is an IPv6 address.

Receiver IP version Flag (R): 1 bit - While set to 1, the receiver IP version flag indicates that the receiver IP address contained in the PDU is an IPv6 address.

record count (RC): 4 bits - Total number of application records contained in the BASIC part of the PDU. A value of zero is considered valid but useless, with the exception of the case of a NULL PDU indicating the end of a RDS reporting session.

length: 16 bits (unsigned integer) - The length of the BASIC part of the RAQMON PDU in units of 32-bit words minus one; this count includes the header and any padding.

DSRC: 32 bits - Data Source identifier represents a unique RAQMON reporting session descriptor that points to a specific reporting session between RDS and RRC. Uniqueness of DSRC is valid only within a reporting session. DSRC values should be randomly generated using vendor-chosen algorithms for each communication session. It is not sufficient to obtain a DSRC simply by calling random() without carefully initializing the state. One could use an algorithm like the one defined in Appendix A.6 in [RFC3550] to create a DSRC. Depending on the choice of algorithm, there is a finite probability that two DSRCs from two different RDSs may be the same. To further reduce the probability that two RDSs pick the same DSRC for two different reporting sessions, it is recommended that an RRC use parameters like Data Source Address (DA), Data Source Name (DN), and layer 2 Media Access Control (MAC) Address in the PDU in conjunction with a DSRC value. It is not mandatory for RDSs to send parameters like Data Source Address (DA), Data Source Name (DN), and MAC Address in every PDU sent to RRC, but occasionally sending these parameters will reduce the probability of DSRC collision drastically. However, this will cause an additional overhead per PDU.

A value of zero for basic (B) bit and trailer (T) bits constitutes a RAQMON NULL PDU (i.e., nothing to report). RDSs MUST send a RAQMON NULL PDU to RRC to indicate the end of the RDS reporting session. A NULL PDU ends with the DSRC field.

SMI Enterprise Code: 16 bits. A value of SMI Enterprise Code = 0 is used to indicate the RMON-WG-compliant BASIC part of the RAQMON PDU format.

Report Type: 8 bits - These bits are reserved by the IETF RMON Working Group. A value of 0 within SMI Enterprise Code = 0 is used for the version of the PDU defined by this document.

The BASIC part of each RAQMON PDU consists of Record Count Number (RC_N) and RAQMON Parameter Presence Flags (RPPF) to indicate the presence of appropriate RAQMON parameters within a record, as defined in Table 1.

RC_N: 8 bits - The Record Count number indicates a sub-session within a communication session. A value of zero is a valid record number. The maximum number of records that can be described in one RAQMON Packet is 256.

RAQMON Parameter Presence Flags (RPPF): 32 bits

Each of these flags, while set, represents that this RAQMON PDU contains corresponding parameters as specified in Table 1.

Bit Sequence Number	Presence/Absence of corresponding Parameter within this RAQMON PDU
0	Data Source Address (DA)
1	Receiver Address (RA)
2	NTP Timestamp
3	Application Name
4	Data Source Name (DN)
5	Receiver Name (RN)
6	Session Setup Status
7	Session Duration
8	Round-Trip End-to-End Net Delay (RTT)
9	One-Way End-to-End Network Delay (OWD)
10	Cumulative Packets Loss
11	Cumulative Packets Discards
12	Total number of App Packets sent
13	Total number of App Packets received
14	Total number of App Octets sent
15	Total number of App Octets received
16	Data Source Device Port Used
17	Receiver Device Port Used
18	Source Layer 2 Priority
19	Source Layer 3 Priority
20	Destination Layer 2 Priority
21	Destination Layer 3 Priority

	22	Source Payload Type
	23	Receiver Payload Type
	24	CPU Utilization
	25	Memory Utilization
	26	Session Setup Delay
	27	Application Delay
	28	IP Packet Delay Variation
	29	Inter arrival Jitter
	30	Packet Discard (in fraction)
	31	Packet Loss (in fraction)

Table 1: RAQMON Parameters and Corresponding RPPF

Data Source Address (DA): 32 bits or 160 bits in binary representation - This parameter is defined in section 5.1 of [RFC4710]. IPv6 addresses are incorporated in Data Source Address by setting the source IP version flag (S bit) of the RAQMON PDU header to 1.

Receiver Address (RA): 32 bits or 160 bits - This parameter is defined in section 5.2 of [RFC4710]. It follows the exact same syntax as Data Source Address but is used to indicate a Receiver Address. IPv6 addresses are incorporated in Receiver Address by setting the receiver IP version flag (R bit) of the RAQMON PDU header to 1.

Session Setup Date/Time (NTP timestamp): 64 bits - This parameter is defined in section 5.7 of [RFC4710] and represented using the timestamp format of the Network Time Protocol (NTP), which is in seconds [RFC1305]. The full resolution NTP timestamp is a 64-bit unsigned fixed-point number with the integer part in the first 32 bits and the fractional part in the last 32 bits.

Application Name: This parameter is defined in section 5.32 of [RFC4710]. The Application Name field starts with an 8-bit octet count describing the length of the text followed by the text itself using UTF-8 encoding. Application Name field is a multiple of 32 bits, and padding will be used if necessary.

A Data Source that does not support NTP SHOULD set the appropriate RAQMON flag to 0 to avoid wasting 64 bits in the PDU. Since the NTP time stamp is intended to provide the setup Date/Time of a session, it is RECOMMENDED that the NTP Timestamp be used only in the first RAQMON PDU after sub-session RC_N setup is completed, in order to use network resources efficiently.

Data Source Name (DN): Defined in section 5.3 of [RFC4710]. The Data Source Name field starts with an 8-bit octet count describing the length of the text followed by the text itself. Padding is used to ensure that the length and text encoding occupy a multiple of 32 bits in the DN field of the PDU. The text MUST NOT be longer than 255 octets. The text is encoded according to the UTF-8 encoding specified in [RFC3629]. Applications SHOULD instruct RDSs to send out the Data Source Name infrequently to ensure efficient usage of network resources as this parameter is expected to remain constant for the duration of the reporting session.

Receiver Name (RN): This metric is defined in section 5.4 of [RFC4710]. Like Data Source Name, the Receiver Name field starts with an 8-bit octet count describing the length of the text, followed by the text itself. The Receiver Name, including the length field encoding, is a multiple of 32 bits and follows the same padding rules as applied to the Data Source Name. Since the Receiver Name is expected to remain constant during the entire reporting session, this information SHOULD be sent out occasionally over random time intervals to maximize success of reaching a RRC and also conserve network bandwidth.

Session Setup Status: The Session (sub-session) Setup Status is defined in section 5.10 of [RFC4710]. This field starts with an 8-bit length field followed by the text itself. Session Setup Status is a multiple of 32 bits.

Session Duration: 32 bits - The Session (sub-session) Duration metric is defined in section 5.9 of [RFC4710]. Session Duration is an unsigned integer expressed in seconds.

Round-Trip End-to-End Network Delay: 32 bits - The Round-Trip End-to-End Network Delay is defined in section 5.11 of [RFC4710]. This field represents the Round-Trip End-to-End Delay of sub-session RC_N, which is an unsigned integer expressed in milliseconds.

One-Way End-to-End Network Delay: 32 bits - The One-Way End-to-End Network Delay is defined in section 5.12 of [RFC4710]. This field represents the One-Way End-to-End Delay of sub-session RC_N, which is an unsigned integer expressed in milliseconds.

Cumulative Application Packet Loss: 32 bits - This parameter is defined in section 5.20 of [RFC4710] as an unsigned integer, representing the total number of packets from sub-session RC_N that have been lost while this RAQMON PDU was generated.

Cumulative Application Packet Discards: 32 bits - This parameter is defined in section 5.22 of [RFC4710] as an unsigned integer representing the total number of packets from sub-session RC_N that have been discarded while this RAQMON PDU was generated.

Total number of Application Packets sent: 32 bits - This parameter is defined in section 5.17 of [RFC4710] as an unsigned integer, representing the total number of packets transmitted within sub-session RC_N by the sender.

Total number of Application Packets received: 32 bits - This parameter is defined in section 5.16 of [RFC4710] and is represented as an unsigned integer representing the total number of packets transmitted within sub-session RC_N by the receiver.

Total number of Application Octets sent: 32 bits - This parameter is defined in section 5.19 of [RFC4710] as an unsigned integer, representing the total number of payload octets (i.e., not including header or padding) transmitted in packets by the sender within sub-session RC_N.

Total number of Application Octets received: 32 bits - This parameter is defined in section 5.18 of [RFC4710] as an unsigned integer representing the total number of payload octets (i.e., not including header or padding) transmitted in packets by the receiver within sub-session RC_N.

Data Source Device Port Used: 16 bits - This parameter is defined in section 5.5 of [RFC4710] and describes the port number used by the Data Source as used by the application in RC_N session while this RAQMON PDU was generated.

Receiver Device Port Used: 16 bits - This parameter is defined in section 5.6 of [RFC4710] and describes the receiver port used by the application to communicate to the receiver. It follows same syntax as Source Device Port Used.

S_Layer2: 8 bits - This parameter, defined in section 5.26 of [RFC4710], is associated to the source's IEEE 802.1D [IEEE802.1D] priority tagging of traffic in the communication sub-session RC_N. Since IEEE 802.1 priority tags are 3 bits long, the first 3 bits of this parameter represent the IEEE 802.1 tag value, and the last 5 bits are padded to 0.

S_Layer3: 8 bits - This parameter, defined in section 5.27 of [RFC4710], represents the layer 3 QoS marking used to send packets to the receiver by this data source during sub-session RC_N.

D_Layer2: 8 bits - This parameter, defined in section 5.28 of [RFC4710], represents layer 2 IEEE 802.1D priority tags used by the receiver to send packets to the data source during sub-session RC_N session if the Data Source can learn such information. Since IEEE 802.1 priority tags are 3 bits long, the first 3 bits of this parameter represent the IEEE 802.1 priority tag value, and the last 5 bits are padded to 0.

D_Layer3: 8 bits - This parameter is defined in section 5.29 of [RFC4710] and represents the layer 3 QoS marking used by the receiver to send packets to the data source during sub-session RC_N, if the Data Source can learn such information.

Source Payload Type: 8 bits - This parameter is defined in section 5.24 of [RFC4710] and specifies the payload type of the data source of the communication sub-session RC_N as defined in [RFC3551].

Receiver Payload Type: 8 bits - This parameter is defined in section 5.25 of [RFC4710] and specifies the receiver payload type of the communication sub-session RC_N as defined in [RFC3551].

CPU Utilization: 8 bits - This parameter, defined in section 5.30 of [RFC4710], represents the percentage of CPU used during session RC_N from the last report until the time this RAQMON PDU was generated. The CPU Utilization is expressed in percents in the range 0 to 100. The value should indicate not only CPU utilization associated to a session RC_N but also actual CPU Utilization, to indicate a snapshot of the CPU utilization of the host running the RDS while session RC_N in progress.

Memory Utilization: 8 bits - This parameter, defined in section 5.31 of [RFC4710], represents the percentage of total memory used during session RC_N up until the time this RAQMON PDU was generated. The memory utilization is expressed in percents 0 to 100. The Memory Utilization value should indicate not only the memory utilization associated to a session RC_N but the total memory utilization, to indicate a snapshot of end-device memory utilization while session RC_N is in progress.

Session Setup Delay: 16 bits - The Session (sub-session) Setup Delay metric is defined in section 5.8 of [RFC4710] and expressed in milliseconds.

Application Delay: 16 bits - The Application Delay is defined in section 5.13 of [RFC4710] and is represented as an unsigned integer expressed in milliseconds.

IP Packet Delay Variation: 16 bits - The IP Packet Delay Variation is defined in section 5.15 of [RFC4710] and is represented as an unsigned integer expressed in milliseconds.

Inter-Arrival Jitter: 16 bits - The Inter-Arrival Jitter is defined in section 5.14 of [RFC4710] and is represented as an unsigned integer expressed in milliseconds.

Packet Discard in Fraction: 8 bits - This parameter is defined in section 5.23 of [RFC4710] and is expressed as a fixed-point number with the binary point at the left edge of the field. (That is equivalent to taking the integer part after multiplying the discard fraction by 256.) This metric is defined to be the number of packets discarded, divided by the total number of packets.

Packet Loss in Fraction: 8 bits - This parameter is defined in section 5.21 of [RFC4710] and is expressed as a fixed-point number, with the binary point at the left edge of the field. The metric is defined to be the number of packets lost divided by the number of packets expected. The value is calculated by dividing the total number of packets lost (after the effects of applying any error protection, such as Forward Error Correction (FEC)) by the total number of packets expected, multiplying the result of the division by 256, limiting the maximum value to 255 (to avoid overflow), and taking the integer part.

padding: 0, 8, 16, or 24 bits - If the padding bit (P) is set, then this field may be present. The actual padding at the end of the BASIC part of the PDU is 0, 8, 16, or 24 bits to make the length of the BASIC part of the PDU a multiple of 32 bits.

2.1.3. APP Part of the RAQMON Protocol Data Unit

The APP part of the RAQMON PDU is intended to accommodate extensions for new applications in a modular manner and without requiring a PDU type value registration.

Vendors may design and publish application-specific extensions. Any RAQMON-compliant RRC MUST be able to recognize vendors' SMI Enterprise Codes and MUST recognize the presence of application-specific extensions identified by using Report Type fields. As represented in Figure 1, the Report Type and Application Length

fields are always located at a fixed offset relative to the start of the extension fields. There is no need for the RRC to understand the semantics of the enterprise-specific parts of the PDU.

SMI Enterprise Code: 32 bits - Vendors and application developers should fill in appropriate SMI Enterprise IDs available at http://www.iana.org/assignments/enterprise-numbers. A non-zero SMI Enterprise Code indicates a vendor- or application-specific extension.

RAQMON PDUs are capable of carrying multiple Application Parts within a PDU.

Report Type: 16 bits - Vendors and application developers should fill in the appropriate report type within a specified SMI Enterprise Code. It is RECOMMENDED that vendors publish application-specific extensions and maintain such report types for better interoperability.

Length of the Application Part: 16 bits (unsigned integer) - The length of the Application Part of the RAQMON PDU in 32-bit words minus one, which includes the header of the Application Part.

Application-dependent data: variable length - Application/ vendor-dependent data is defined by the application developers. It is interpreted by the vendor-specific application and not by the RRC itself. Its length must be a multiple of 32 bits and will be padded if necessary.

2.1.4. Byte Order, Alignment, and Time Format of RAQMON PDUs

All integer fields are carried in network byte order, that is, most significant byte (octet) first. This byte order is commonly known as big-endian. The transmission order is described in detail in [RFC791]. Unless otherwise noted, numeric constants are in decimal (base 10).

All header data is aligned to its natural length, i.e., 16-bit fields are aligned on even offsets, 32-bit fields are aligned at offsets divisible by four, etc. Octets designated as padding have the value zero.

2.2. Securing RAQMON Session

The RAQMON session, initiated over TCP transport, between an RDS and an RRC carries monitoring information from an RDS client to the RRC, the collector. The RRC distinguishes between clients based on various identifiers used by the RDS to identify itself to the RRC

(Data Source Address and Data Source Name) and the RRC (Receiver's Address and Receiver's Name).

In order to ensure integrity of the claimed identities of RDS and RRC to each other, authentication services are required.

Subsequently, where protection from unauthorized modification and unauthorized disclosure of RAQMON data in transit from RDS to RRC is needed, data confidentiality and message integrity services will be required. In order to prevent monitoring-misinformation due to session-recording and replay by unauthorized sources, replay protection services may be required.

TLS provides, at the transport layer, the required authentication services through the handshake protocol and subsequent data confidentiality, message integrity, and replay protection of the application protocol using a ciphersuite negotiated during authentication.

The RDS client authenticates the RRC in session. The RRC optionally authenticates the RDS.

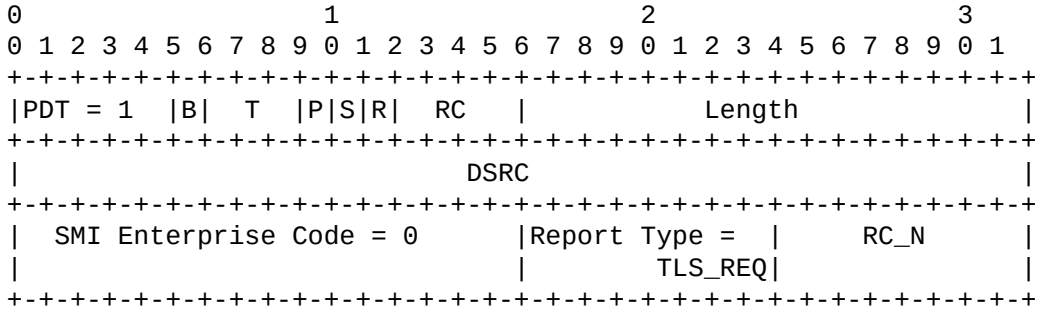


Figure 2: RAQMON StartTLS Request - TLS_REQ

The protection of a RAQMON session starts with the RDS client's StartTLS request upon successful establishment of the TCP session. The RDS sends the StartTLS request by transmitting the TLS_REQ PDU as in Figure 2. This PDU is distinguished by TLS_REQ Report Type.

Following this request, the client MUST NOT send any PDUs on this connection until it receives a StartTLS response.

Other fields of the PDU are as specified in Figure 1.

The flags field do not carry any significance and exist for compatibility with the generic RAQMON PDU. The flags field in this version MUST be ignored.

When a StartTLS request is made, the target server, RRC, MUST return a RAQMON PDU containing a StartTLS response, TLS_RESP. A RAQMON TLS_RESP is defined as follows:

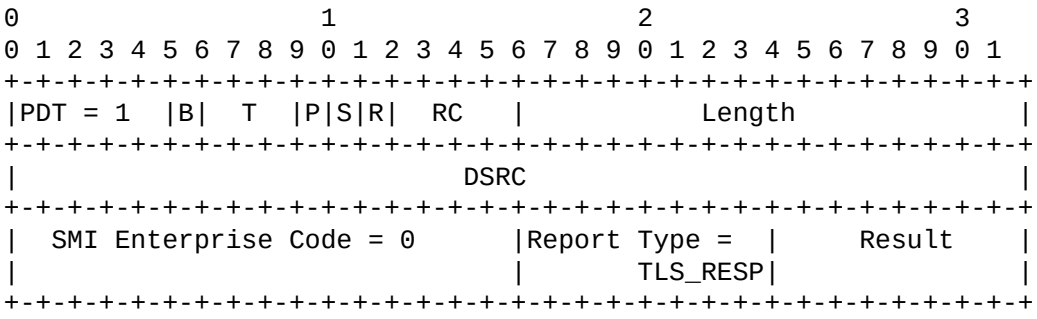


Figure 3: RAQMON StartTLS Response - TLS_RESP

The RRC responds to the StartTLS request by transmitting the TLS_RESP PDU as in Figure 3. This PDU is distinguished by TLS_RESP Report Type.

The Result field is an octet containing the result of the request. This field can carry one of the following values:

Value	Mnemonic	Result
0	OK	Success. The server is willing and able to negotiate TLS.
1	OP_ERR	Sequencing Error (e.g., TLS already established).
2	PROTO_ERR	TLS not supported or incorrect PDU format.
3	UNAVAIL	TLS service problem or RRC server going down.
4	CONF_REQD	Confidentiality Service Required.
5	STRONG_AUTH_REQD	Strong Authentication Service Required.
6	REFERRAL	Referral to a RRC Server supporting TLS.

Table 2

Other fields of the PDU are as specified in Figure 1.

The server MUST return OP_ERR if the client violates any of the StartTLS operation sequencing requirements described in the section below.

If the server does not support TLS (whether by design or by current configuration), it MUST set the resultCode to PROTO_ERR or to REFERRAL. The server MUST include an actual referral value in the RAQMON REFER field if it returns a resultCode of referral. The client's current session is unaffected if the server does not support TLS. The client MAY proceed with RAQMON session, or it MAY close the connection.

The server MUST return UNAVAIL if it supports TLS but cannot establish a TLS connection for some reason, e.g., if the certificate server not responding, if it cannot contact its TLS implementation, or if the server is in process of shutting down. The client MAY retry the StartTLS operation, MAY proceed with RAQMON session, or MAY close the connection.

2.2.1. Sequencing of the Start TLS Operation

This section describes the overall procedures clients and servers MUST follow for TLS establishment. These procedures take into consideration various aspects of the overall security of the RAQMON connection including discovery of resulting security level.

2.2.1.1. Requesting to Start TLS on a RAQMON Association

The client MAY send the StartTLS request at any time after establishing an RAQMON (TCP) connection, except that in the following cases the client MUST NOT send a StartTLS request:

- if TLS is currently established on the connection, or
- if RAQMON traffic is in progress on the connection.

The result of violating any of these requirements is a Result of OP_ERR, as described above in Table 2.

If the client did not establish a TLS connection before sending any other requests, and the server requires the client to establish a TLS connection before performing a particular request, the server MUST reject that request with a CONF_REQD or STRONG_AUTH_REQD result. The client MAY send a Start TLS extended request, or it MAY choose to close the connection.

2.2.1.2. Starting TLS

The server will return an extended response with the resultCode of success if it is willing and able to negotiate TLS. It will return other resultCodes, documented above, if it is unable.

In the successful case, the client, which has ceased to transfer RAQMON PDUs on the connection, MUST either begin a TLS negotiation or close the connection. The client will send PDUs in the TLS Record Protocol directly over the underlying transport connection to the server to initiate TLS negotiation [TLS].

2.2.1.3. TLS Version Negotiation

Negotiating the version of TLS or SSL to be used is a part of the TLS Handshake Protocol, as documented in [TLS]. The reader is referred to that document for details.

2.2.1.4. Discovery of Resultant Security Level

After a TLS connection is established on a RAQMON connection, both parties MUST individually decide whether or not to continue based on the security assurance level achieved. Ascertaining the TLS connection's assurance level is implementation dependent and is accomplished by communicating with one's respective local TLS implementation.

If the client or server decides that the level of authentication or confidentiality is not high enough for it to continue, it SHOULD gracefully close the TLS connection immediately after the TLS negotiation has completed Section 2.2.2.1.

The client MAY attempt to Start TLS again, MAY disconnect, or MAY proceed to send RAQMON session data, if RRC policy permits.

2.2.1.5. Server Identity Check

The client MUST check its understanding of the server's hostname against the server's identity as presented in the server's Certificate message, in order to prevent man-in-the-middle attacks.

Matching is performed according to these rules:

- The client MUST use the server dnsNAME in the subjectAltName field to validate the server certificate presented. The server dnsName MUST be part of subjectAltName of the server.

- Matching is case-insensitive.

- The "*" wildcard character is allowed. If present, it applies only to the left-most name component.

  For example, *.example.com would match a.example.com, b.example.com, etc., but not example.com. If more than one identity of a given type is present in the certificate (e.g., more than one dNSName name), a match in any one of the set is considered acceptable.

If the hostname does not match the dNSName-based identity in the certificate per the above check, automated clients SHOULD close the connection, returning and/or logging an error indicating that the server's identity is suspect.

Beyond the server identity checks described in this section, clients SHOULD be prepared to do further checking to ensure that the server is authorized to provide the service it is observed to provide. The client MAY need to make use of local policy information.

We also refer readers to similar guidelines as applied for LDAP over TLS [RFC4513].

2.2.1.6. Client Identity Check

Anonymous TLS authentication helps establish a TLS RAQMON session that offers

- server-authentication in course of TLS establishment and
- confidentiality and replay protection of RAQMON traffic, but
- no protection against man-in-the-middle attacks during session establishment and
- no protection from spoofing attacks by unauthorized clients.

The server MUST authenticate the RDS client when deployment is susceptible to the above threats. This is done by requiring client authentication during TLS session establishment.

In the TLS negotiation, the server MUST request a certificate. The client will provide its certificate to the server and MUST perform a private-key-based encryption, proving it has the private key associated with the certificate.

As deployments will require protection of sensitive data in transit, the client and server MUST negotiate a ciphersuite that contains a bulk encryption algorithm of appropriate strength.

The server MUST verify that the client's certificate is valid. The server will normally check that the certificate is issued by a known CA, and that none of the certificates on the client's certificate chain are invalid or revoked. There are several procedures by which the server can perform these checks.

The server validates the certificate by the Distinguished Name of the RDS client entity in the Subject field of the certificate.

A corresponding set of guidelines will apply to use of TLS-PSK modes [TLS-PSK] using pre-shared keys instead of client certificates.

2.2.1.7. Refresh of Server Capabilities Information

The client MUST refresh any cached server capabilities information upon TLS session establishment, such as prior RRC state related to a previous RAQMON session based on another DSRC. This is necessary to protect against active-intermediary attacks, which may have altered any server capabilities information retrieved prior to TLS establishment. The server MAY advertise different capabilities after TLS establishment.

2.2.2. Closing a TLS Connection

2.2.2.1. Graceful Closure

Either the client or server MAY terminate the TLS connection on an RAQMON session by sending a TLS closure alert. This will leave the RAQMON connection intact.

Before closing a TLS connection, the client MUST wait for any outstanding RAQMON transmissions to complete. This happens naturally when the RAQMON client is single-threaded and synchronous.

After the initiator of a close has sent a closure alert, it MUST discard any TLS messages until it has received an alert from the other party. It will cease to send TLS Record Protocol PDUs and, following the receipt of the alert, MAY send and receive RAQMON PDUs.

The other party, if it receives a closure alert, MUST immediately transmit a TLS closure alert. It will subsequently cease to send TLS Record Protocol PDUs and MAY send and receive RAQMON PDUs.

2.2.2.2. Abrupt Closure

Either the client or server MAY abruptly close the entire RAQMON session and any TLS connection established on it by dropping the underlying TCP connection. It MAY be possible for RRC to send RDS a disconnection notification, which allows the client to know that the disconnection is not due to network failure. However, this message is not defined in this version.

2.3. SNMP Notifications as an RDS/RRC Network Transport Protocol

It was an inherent objective of the RAQMON Framework to re-use existing application-level transport protocols to maximize the usage of existing installations as well as to avoid transport-protocol-level complexities in the design process. Choice of SNMP as a means to transport RAQMON PDU was motivated by the intent of using existing installed devices implementing SNMP agents as RAQMON Data Sources (RDSs).

There are some potential problems with the usage of SNMP as a transport mapping protocol:

- The potential of congestion is higher than with the TCP transport, because of the usage of UDP at the transport layer.

- The encoding of the information is less efficient, and this results in bigger message size, which again may negatively impact congestion conditions and memory size requirements in the devices.

In order to avoid these potential problems, the following recommendations are made:

- Usage of the TCP transport is RECOMMENDED in deployment over the SNMP transport wherever available for a pair of RDS/RRC.

- The usage of Inform PDUs is RECOMMENDED.

- The usage of Traps PDU is NOT RECOMMENDED.

- It is RECOMMENDED that information carried by notifications be maintained within the limits of the MTU size in order to avoid fragmentation.

If SNMP is chosen as a mechanism to transport RAQMON PDUs, the following specification applies to RAQMON-related usage of SNMP:

- RDSs implement the capability of embedding RAQMON parameters in SNMP Notifications, re-using well-known SNMP mechanisms to report RAQMON Statistics. The RAQMON RDS MIB module, as specified in 2.1.1, MUST be used in order to map the RAQMON PDUs onto the SNMP Notifications transport.

- Since RDSs are not computationally rich, and in order to keep the RDS realization as lightweight as possible, RDSs MAY fail to respond to SNMP requests like GET, SET, etc., with the exception of the GET and SET commands required to implement the User-Based Security Model (USM) defined by [RFC3414].

- In order to meet congestion safety requirements, SNMP INFORM PDUs SHOULD be used. In case INFORM PDUs are used, RDSs MUST process the SNMP INFORM responses from RRCs and MUST serialize the PDU transmission rate, i.e., limit the number of PDUS sent in a specific time interval.

- Standard UDP port 162 SHOULD be used for SNMP Notifications.

2.3.1. Encoding RAQMON Using the RAQMON RDS MIB Module

The RAQMON RDS MIB module is used to map RAQMON PDUs onto SNMP Notifications for transport purposes. The MIB module defines the objects needed for mapping the BASIC part of RAQMON PDU, defined in [RFC4710], as well as the Notifications themselves. In order to incorporate any application-specific extensions in the Application (APP) part of RAQMON PDU, as defined in [RFC4710], additional variable bindings MAY be included in RAQMON notifications as described in the MIB module.

For a detailed overview of the documents that describe the current Internet-Standard Management Framework, please refer to section 7 of [RFC3410].

Managed objects are accessed via a virtual information store, termed the Management Information Base or MIB. MIB objects are generally accessed through the Simple Network Management Protocol (SNMP). Objects in the MIB are defined using the mechanisms defined in the Structure of Management Information (SMI). This memo specifies a MIB module that is compliant to the SMIv2, which is described in STD 58, [RFC2578], STD 58, [RFC2579] and STD 58, [RFC2580].

The following MIB module IMPORTS definitions from the following:

```
        SNMPv2-SMI [RFC2578]
        SNMPv2-TC [RFC2579]
        SNMPv2-CONF [RFC2580]
        RMON-MIB [RFC2819]
        DIFFSERV-DSCP-TC [RFC3289]
        SNMP-FRAMEWORK-MIB [RFC3411]
        INET-ADDRESS-MIB [RFC4001]
```

It also uses REFERENCE clauses to refer to [RFC4710].

```
RAQMON-RDS-MIB DEFINITIONS ::= BEGIN

   IMPORTS
       MODULE-IDENTITY, OBJECT-TYPE, NOTIFICATION-TYPE,
       Counter32, Unsigned32
           FROM SNMPv2-SMI

       DateAndTime
           FROM SNMPv2-TC

       rmon
           FROM RMON-MIB

       SnmpAdminString
           FROM SNMP-FRAMEWORK-MIB

       InetAddressType, InetAddress, InetPortNumber
           FROM INET-ADDRESS-MIB

       Dscp
           FROM DIFFSERV-DSCP-TC

       MODULE-COMPLIANCE, OBJECT-GROUP, NOTIFICATION-GROUP
           FROM SNMPv2-CONF;

   raqmonDsMIB MODULE-IDENTITY
       LAST-UPDATED "200610100000Z"      -- October 10, 2006
       ORGANIZATION "RMON Working Group"
       CONTACT-INFO
           "WG EMail: rmonmib@ietf.org
            Subscribe: rmonmib-request@ietf.org

            MIB Editor:
            Eugene Golovinsky
            Postal: BMC Software, Inc.
                    2101 CityWest Boulevard,
```

```
                    Houston, TX, 77094
                    USA
             Tel:   +713-918-1816
             Email: egolovin@bmc.com
            "
        DESCRIPTION
            "This is the RAQMON Data Source notification MIB Module.
             It provides a mapping of RAQMON PDUs to SNMP
             notifications.

             Ds stands for data source.

             Note that all of the object types defined in this module
             are accessible-for-notify and would consequently not be
             available to a browser using simple Get, GetNext, or
             GetBulk requests.

             Copyright (c) The Internet Society (2006).

             This version of this MIB module is part of RFC 4712;
             See the RFC itself for full legal notices."

        REVISION      "200610100000Z"     -- October 10, 2006
        DESCRIPTION
            "Initial version, published as RFC 4712."

              ::= { rmon 32 }

   -- This OID allocation conforms to [RFC3737]


      raqmonDsNotifications OBJECT IDENTIFIER ::= { raqmonDsMIB 0 }
      raqmonDsMIBObjects OBJECT IDENTIFIER ::= { raqmonDsMIB 1 }
      raqmonDsConformance OBJECT IDENTIFIER ::= { raqmonDsMIB 2 }

      raqmonDsNotificationTable OBJECT-TYPE
          SYNTAX SEQUENCE OF RaqmonDsNotificationEntry
          MAX-ACCESS not-accessible
          STATUS     current
          DESCRIPTION
              "This conceptual table provides the SNMP mapping of
               the RAQMON BASIC PDU.  It is indexed by the RAQMON
               Data Source, sub-session, and address of the peer
               entity.

               Note that there is no concern about the indexation of
               this table exceeding the limits defined by RFC 2578
               Section 3.5.  According to [RFC4710], Section 5.1,
```

```
            only IPv4 and IPv6 addresses can be reported as
            participant addresses."
        ::= { raqmonDsMIBObjects 1 }

    raqmonDsNotificationEntry OBJECT-TYPE
        SYNTAX     RaqmonDsNotificationEntry
        MAX-ACCESS not-accessible
        STATUS     current
        DESCRIPTION
            "The entry (row) is not retrievable and is not kept by
            RDSs.  It serves data organization purposes only."
        INDEX { raqmonDsDSRC, raqmonDsRCN, raqmonDsPeerAddrType,
                raqmonDsPeerAddr }
        ::= { raqmonDsNotificationTable 1 }

    RaqmonDsNotificationEntry ::= SEQUENCE {
            raqmonDsDSRC                       Unsigned32,
            raqmonDsRCN                        Unsigned32,
            raqmonDsPeerAddrType               InetAddressType,
            raqmonDsPeerAddr                   InetAddress,
            raqmonDsAppName                    SnmpAdminString,
            raqmonDsDataSourceDevicePort       InetPortNumber,
            raqmonDsReceiverDevicePort         InetPortNumber,
            raqmonDsSessionSetupDateTime       DateAndTime,
            raqmonDsSessionSetupDelay          Unsigned32,
            raqmonDsSessionDuration            Unsigned32,
            raqmonDsSessionSetupStatus         SnmpAdminString,
            raqmonDsRoundTripEndToEndNetDelay Unsigned32,
            raqmonDsOneWayEndToEndNetDelay     Unsigned32,
            raqmonDsApplicationDelay           Unsigned32,
            raqmonDsInterArrivalJitter         Unsigned32,
            raqmonDsIPPacketDelayVariation     Unsigned32,
            raqmonDsTotalPacketsReceived       Counter32,
            raqmonDsTotalPacketsSent           Counter32,
            raqmonDsTotalOctetsReceived        Counter32,
            raqmonDsTotalOctetsSent            Counter32,
            raqmonDsCumulativePacketLoss       Counter32,
            raqmonDsPacketLossFraction         Unsigned32,
            raqmonDsCumulativeDiscards         Counter32,
            raqmonDsDiscardsFraction           Unsigned32,
            raqmonDsSourcePayloadType          Unsigned32,
            raqmonDsReceiverPayloadType        Unsigned32,
            raqmonDsSourceLayer2Priority       Unsigned32,
            raqmonDsSourceDscp                 Dscp,
            raqmonDsDestinationLayer2Priority Unsigned32,
            raqmonDsDestinationDscp            Dscp,
            raqmonDsCpuUtilization             Unsigned32,
            raqmonDsMemoryUtilization          Unsigned32 }
```

```
raqmonDsDSRC OBJECT-TYPE
    SYNTAX     Unsigned32
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION
        "Data Source identifier represents a unique session
         descriptor that points to a specific session
         between communicating entities.  Identifiers unique for
         sessions conducted between two entities are
         generated by the communicating entities.  Zero is a
         valid value, with no special semantics."
    ::= { raqmonDsNotificationEntry 1 }

raqmonDsRCN OBJECT-TYPE
     SYNTAX      Unsigned32 (0..15)
     MAX-ACCESS  not-accessible
     STATUS      current
     DESCRIPTION
         "The Record Count Number indicates a sub-session
          within a communication session.  A maximum number of 16
          sub-sessions are supported; this limitation is
          dictated by reasons of compatibility with other
          transport protocols."
     ::= { raqmonDsNotificationEntry 2 }

raqmonDsPeerAddrType OBJECT-TYPE
    SYNTAX InetAddressType
    MAX-ACCESS not-accessible
    STATUS current
    DESCRIPTION
        "The type of the Internet address of the peer participant
         for this session."
    REFERENCE
        "Section 5.2 of [RFC4710]"
    ::= { raqmonDsNotificationEntry 3 }

raqmonDsPeerAddr OBJECT-TYPE
    SYNTAX InetAddress
    MAX-ACCESS not-accessible
    STATUS current
    DESCRIPTION
        "The Internet Address of the peer participant for this
         session."
    REFERENCE
        "Section 5.2 of [RFC4710]"
    ::= { raqmonDsNotificationEntry 4 }

raqmonDsAppName  OBJECT-TYPE
```

```
        SYNTAX     SnmpAdminString
        MAX-ACCESS accessible-for-notify
        STATUS     current
        DESCRIPTION
            "This is a text string giving the name and possibly the
             version of the application associated with that session,
             e.g., 'XYZ VoIP Agent 1.2'."
        REFERENCE
            "Section 5.28 of [RFC4710]"
        ::= { raqmonDsNotificationEntry 5 }

    raqmonDsDataSourceDevicePort OBJECT-TYPE
        SYNTAX     InetPortNumber
        MAX-ACCESS accessible-for-notify
        STATUS     current
        DESCRIPTION
            "The port number from which data for this session was sent
             by the Data Source device."
        REFERENCE
            "Section 5.5 of [RFC4710]"
        ::= { raqmonDsNotificationEntry 6 }

    raqmonDsReceiverDevicePort OBJECT-TYPE
        SYNTAX     InetPortNumber
        MAX-ACCESS accessible-for-notify
        STATUS     current
        DESCRIPTION
            "The port number where the data for this session was
             received."
        REFERENCE
            "Section 5.6 of [RFC4710]"
        ::= { raqmonDsNotificationEntry 7 }

    raqmonDsSessionSetupDateTime OBJECT-TYPE
        SYNTAX     DateAndTime
        MAX-ACCESS accessible-for-notify
        STATUS     current
        DESCRIPTION
            "The time when session was initiated."
        REFERENCE
            "Section 5.7 of [RFC4710]"
    ::= { raqmonDsNotificationEntry 8 }

    raqmonDsSessionSetupDelay OBJECT-TYPE
        SYNTAX     Unsigned32 (0..65535)
        UNITS      "milliseconds"
        MAX-ACCESS accessible-for-notify
        STATUS     current
```

```
        DESCRIPTION
            "Session setup time."
        REFERENCE
            "Section 5.8 of [RFC4710]"
        ::= { raqmonDsNotificationEntry 9 }

    raqmonDsSessionDuration OBJECT-TYPE
        SYNTAX     Unsigned32
        UNITS      "seconds"
        MAX-ACCESS accessible-for-notify
        STATUS     current
        DESCRIPTION
            "Session duration, including setup time.  The SYNTAX of
             this object allows expression of the duration of sessions
             that do not exceed 4660 hours and 20 minutes."
        REFERENCE
            "Section 5.9 of [RFC4710]"
        ::= { raqmonDsNotificationEntry 10 }

    raqmonDsSessionSetupStatus OBJECT-TYPE
        SYNTAX     SnmpAdminString
        MAX-ACCESS accessible-for-notify
        STATUS     current
        DESCRIPTION
            "Describes appropriate communication session states, e.g.,
             Call Established successfully, RSVP reservation
             failed, etc."
        REFERENCE
            "Section 5.10 of [RFC4710]"
        ::= { raqmonDsNotificationEntry 11 }

    raqmonDsRoundTripEndToEndNetDelay OBJECT-TYPE
        SYNTAX     Unsigned32
        UNITS      "milliseconds"
        MAX-ACCESS accessible-for-notify
        STATUS     current
        DESCRIPTION
            "Most recent available information about the
             round-trip end-to-end network delay."
        REFERENCE
            "Section 5.11 of [RFC4710]"
        ::= { raqmonDsNotificationEntry  12}

    raqmonDsOneWayEndToEndNetDelay OBJECT-TYPE
        SYNTAX     Unsigned32
        UNITS      "milliseconds"
        MAX-ACCESS accessible-for-notify
        STATUS     current
```

```
            DESCRIPTION
                "Most recent available information about the
                 one-way end-to-end network delay."
            REFERENCE
                "Section 5.12 of [RFC4710]"
            ::= { raqmonDsNotificationEntry  13}

        raqmonDsApplicationDelay OBJECT-TYPE
            SYNTAX     Unsigned32  (0..65535)
            UNITS      "milliseconds"
            MAX-ACCESS accessible-for-notify
            STATUS     current
            DESCRIPTION
                "Most recent available information about the
                 application delay."
            REFERENCE
                "Section 5.13 of [RFC4710"
            ::= { raqmonDsNotificationEntry  14}

        raqmonDsInterArrivalJitter OBJECT-TYPE
            SYNTAX     Unsigned32  (0..65535)
            UNITS      "milliseconds"
            MAX-ACCESS accessible-for-notify
            STATUS     current
            DESCRIPTION
                "An estimate of the inter-arrival jitter."
            REFERENCE
                "Section 5.14 of [RFC4710]"
            ::= { raqmonDsNotificationEntry  15}

        raqmonDsIPPacketDelayVariation OBJECT-TYPE
            SYNTAX     Unsigned32  (0..65535)
            UNITS      "milliseconds"
            MAX-ACCESS accessible-for-notify
            STATUS     current
            DESCRIPTION
                "An estimate of the inter-arrival delay variation."
            REFERENCE
                "Section 5.15 of [RFC4710]"
            ::= { raqmonDsNotificationEntry  16}

        raqmonDsTotalPacketsReceived OBJECT-TYPE
            SYNTAX     Counter32
            UNITS     "packets"
            MAX-ACCESS accessible-for-notify
            STATUS     current
            DESCRIPTION
                "The number of packets transmitted within a communication
```

```
             session by the receiver since the start of the session."
        REFERENCE
            "Section 5.16 of [RFC4710]"
        ::= { raqmonDsNotificationEntry 17 }

    raqmonDsTotalPacketsSent OBJECT-TYPE
        SYNTAX     Counter32
        UNITS     "packets"
        MAX-ACCESS accessible-for-notify
        STATUS     current
        DESCRIPTION
            "The number of packets transmitted within a communication
             session by the sender since the start of the session."
        REFERENCE
            "Section 5.17 of [RFC4710]"
        ::= { raqmonDsNotificationEntry 18 }

    raqmonDsTotalOctetsReceived OBJECT-TYPE
        SYNTAX     Counter32
        UNITS      "octets"
        MAX-ACCESS accessible-for-notify
        STATUS     current
        DESCRIPTION
            "The total number of payload octets (i.e., not including
             header or padding octets) transmitted in packets by the
             receiver within a communication session since the start
             of the session."
        REFERENCE
            "Section 5.18 of [RFC4710]"
        ::= { raqmonDsNotificationEntry 19 }

    raqmonDsTotalOctetsSent OBJECT-TYPE
        SYNTAX     Counter32
        UNITS      "octets"
        MAX-ACCESS accessible-for-notify
        STATUS     current
        DESCRIPTION
            "The number of payload octets (i.e., not including headers
             or padding) transmitted in packets by the sender within
             a communication sub-session since the start of the
             session."
        REFERENCE
            "Section 5.19 of [RFC4710]"
        ::= { raqmonDsNotificationEntry 20 }

    raqmonDsCumulativePacketLoss OBJECT-TYPE
        SYNTAX     Counter32
        UNITS      "packets"
```

```
        MAX-ACCESS accessible-for-notify
        STATUS     current
        DESCRIPTION
            "The number of packets from this session whose loss
             had been detected since the start of the session."
        REFERENCE
            "Section 5.20 of [RFC4710]"
        ::= { raqmonDsNotificationEntry 21 }

    raqmonDsPacketLossFraction OBJECT-TYPE
        SYNTAX     Unsigned32 (0..100)
        UNITS      "percentage of packets sent"
        MAX-ACCESS accessible-for-notify
        STATUS     current
        DESCRIPTION
            "The percentage of lost packets with respect to the
             overall packets sent.  This is defined to be 100 times
             the number of packets lost divided by the number of
             packets expected."
        REFERENCE
            "Section 5.21 of [RFC4710]"
        ::= { raqmonDsNotificationEntry 22 }

    raqmonDsCumulativeDiscards OBJECT-TYPE
        SYNTAX     Counter32
        UNITS      "packets"
        MAX-ACCESS accessible-for-notify
        STATUS     current
        DESCRIPTION
            "The number of packet discards detected since the
             start of the session."
        REFERENCE
            "Section 5.22 of [RFC4710]"
        ::= { raqmonDsNotificationEntry 23 }

    raqmonDsDiscardsFraction OBJECT-TYPE
        SYNTAX     Unsigned32 (0..100)
        UNITS      "percentage of packets sent"
        MAX-ACCESS accessible-for-notify
        STATUS     current
        DESCRIPTION
            "The percentage of discards with respect to the overall
             packets sent.  This is defined to be 100 times the number
             of discards divided by the number of packets expected."
        REFERENCE
            "Section 5.23 of [RFC4710]"
        ::= { raqmonDsNotificationEntry 24 }
```

```
raqmonDsSourcePayloadType OBJECT-TYPE
    SYNTAX     Unsigned32 (0..127)
    MAX-ACCESS accessible-for-notify
    STATUS     current
    DESCRIPTION
        "The payload type of the packet sent by this RDS."
    REFERENCE
        "RFC 1890, Section 5.24 of [RFC4710] "
    ::= { raqmonDsNotificationEntry 25 }

raqmonDsReceiverPayloadType OBJECT-TYPE
    SYNTAX     Unsigned32 (0..127)
    MAX-ACCESS accessible-for-notify
    STATUS     current
    DESCRIPTION
        "The payload type of the packet received by this RDS."
    REFERENCE
        "RFC 1890, Section 5.25 of [RFC4710] "
::= { raqmonDsNotificationEntry 26 }

raqmonDsSourceLayer2Priority OBJECT-TYPE
    SYNTAX     Unsigned32 (0..7)
    MAX-ACCESS accessible-for-notify
    STATUS     current
    DESCRIPTION
        "Source Layer 2 priority used by the data source to send
         packets to the receiver by this data source during this
         communication session."
    REFERENCE
        "Section 5.26 of [RFC4710]"
    ::= { raqmonDsNotificationEntry 27 }

raqmonDsSourceDscp OBJECT-TYPE
    SYNTAX     Dscp
    MAX-ACCESS accessible-for-notify
    STATUS     current
    DESCRIPTION
        "Layer 3 TOS/DSCP values used by the Data Source to
         prioritize traffic sent."
    REFERENCE
        "Section 5.27 of [RFC4710]"
    ::= { raqmonDsNotificationEntry 28 }

raqmonDsDestinationLayer2Priority OBJECT-TYPE
    SYNTAX     Unsigned32 (0..7)
    MAX-ACCESS accessible-for-notify
    STATUS     current
    DESCRIPTION
```

```
            "Destination Layer 2 priority.  This is the priority used
             by the peer communicating entity to send packets to the
             data source."
        REFERENCE
            "Section 5.28 of [RFC4710]"
        ::= { raqmonDsNotificationEntry 29 }

    raqmonDsDestinationDscp OBJECT-TYPE
        SYNTAX     Dscp
        MAX-ACCESS accessible-for-notify
        STATUS     current
        DESCRIPTION
            "Layer 3 TOS/DSCP values used by the
             peer communicating entity to prioritize traffic
             sent to the source."
        REFERENCE
            "Section 5.29 of [RFC4710]"
        ::= { raqmonDsNotificationEntry 30 }

    raqmonDsCpuUtilization OBJECT-TYPE
        SYNTAX     Unsigned32 (0..100)
        UNITS      "percent"
        MAX-ACCESS accessible-for-notify
        STATUS     current
        DESCRIPTION
            "Latest available information about the total CPU
             utilization."
        REFERENCE
            "Section 5.30 of [RFC4710]"
        ::= { raqmonDsNotificationEntry 31 }

    raqmonDsMemoryUtilization OBJECT-TYPE
        SYNTAX     Unsigned32 (0..100)
        UNITS      "percent"
        MAX-ACCESS accessible-for-notify
        STATUS     current
        DESCRIPTION
            "Latest available information about the total memory
             utilization."
        REFERENCE
            "Section 5.31 of [RFC4710]"
        ::= { raqmonDsNotificationEntry 32 }

    -- definitions of the notifications
    --
    -- raqmonDsAppName is the only object that MUST be sent by an
    -- RDS every time the static notification is generated.
```

```
      -- raqmonDsTotalPacketsReceived is the only object that MUST be
      -- sent by an RD every time the dynamic notification is generated.

      -- Other objects from the raqmonDsNotificationTable may be
      -- included in the variable binding list.  Specifically, a raqmon
      -- notification will include MIB objects that provide information
      -- about metrics that characterize the application session

         raqmonDsStaticNotification NOTIFICATION-TYPE
          OBJECTS { raqmonDsAppName }
          STATUS current
          DESCRIPTION
              "This notification maps the static parameters in the
               BASIC RAQMON PDU onto an SNMP transport.
               This notification is expected to be sent once per
               session, or when a new sub-session is initiated.
               The following objects MAY be carried by the
               raqmonDsStaticNotification:

               raqmonDsDataSourceDevicePort,
               raqmonDsReceiverDevicePort,
               raqmonDsSessionSetupDateTime,
               raqmonDsSessionSetupDelay,
               raqmonDsSessionDuration,
               raqmonDsSourcePayloadType,
               raqmonDsReceiverPayloadType,
               raqmonDsSourceLayer2Priority,
               raqmonDsSourceDscp,
               raqmonDsDestinationLayer2Priority,
               raqmonDsDestinationDscp

               It is RECOMMENDED to keep the size of a notification
               within the MTU size limits in order to avoid
               fragmentation."
          ::= { raqmonDsNotifications  1 }

      raqmonDsDynamicNotification NOTIFICATION-TYPE
          OBJECTS { raqmonDsTotalPacketsReceived }
          STATUS current
          DESCRIPTION
              "This notification maps the dynamic parameters in the
               BASIC RAQMON PDU onto an SNMP transport.

               The following objects MAY be carried by the
               raqmonDsDynamicNotification:

               raqmonDsRoundTripEndToEndNetDelay,
               raqmonDsOneWayEndToEndNetDelay,
```

```
                raqmonDsApplicationDelay,
                raqmonDsInterArrivalJitter,
                raqmonDsIPPacketDelayVariation,
                raqmonDsTotalPacketsSent,
                raqmonDsTotalOctetsReceived,
                raqmonDsTotalOctetsSent,
                raqmonDsCumulativePacketLoss,
                raqmonDsPacketLossFraction,
                raqmonDsCumulativeDiscards,
                raqmonDsDiscardsFraction,
                raqmonDsCpuUtilization,
                raqmonDsMemoryUtilization

                It is RECOMMENDED to keep the size of a notification
                within the MTU size limits in order to avoid
                fragmentation."

            ::= { raqmonDsNotifications  2 }

        raqmonDsByeNotification NOTIFICATION-TYPE
            OBJECTS { raqmonDsAppName }
            STATUS current
            DESCRIPTION
                "The BYE Notification.  This Notification is the
                 equivalent of the RAQMON NULL PDU, which signals the
                 end of a RAQMON session."
            ::= { raqmonDsNotifications  3 }

        --
        -- conformance information
        raqmonDsCompliance OBJECT IDENTIFIER ::=
                                              { raqmonDsConformance 1 }
        raqmonDsGroups OBJECT IDENTIFIER ::= { raqmonDsConformance 2 }

    raqmonDsBasicCompliance MODULE-COMPLIANCE
            STATUS current
            DESCRIPTION
               "The compliance statement for SNMP entities that
                implement this MIB module.

                There are a number of INDEX objects that cannot be
                represented in the form of OBJECT clauses in SMIv2, but
                for which we have the following compliance requirements,
                expressed in OBJECT clause form in this description
                clause:

                -- OBJECT      raqmonDsPeerAddrType
                -- SYNTAX      InetAddressType { ipv4(1), ipv6(2) }
```

```
                -- DESCRIPTION
                --     This MIB requires support for only global IPv4
                --     and IPv6 address types.
                --
                -- OBJECT      raqmonDsPeerAddr
                -- SYNTAX      InetAddress (SIZE(4|16))
                -- DESCRIPTION
                --     This MIB requires support for only global IPv4
                --     and IPv6 address types.
                --
               "
            MODULE  -- this module
                MANDATORY-GROUPS { raqmonDsNotificationGroup,
                                   raqmonDsPayloadGroup }
            ::= { raqmonDsCompliance 1 }

        raqmonDsNotificationGroup NOTIFICATION-GROUP
            NOTIFICATIONS { raqmonDsStaticNotification,
                            raqmonDsDynamicNotification,
                            raqmonDsByeNotification }
            STATUS current
            DESCRIPTION
                "Standard RAQMON Data Source Notification group."
            ::= { raqmonDsGroups 1 }

        raqmonDsPayloadGroup OBJECT-GROUP
            OBJECTS { raqmonDsAppName,
                      raqmonDsDataSourceDevicePort,
                      raqmonDsReceiverDevicePort,
                      raqmonDsSessionSetupDateTime,
                      raqmonDsSessionSetupDelay,
                      raqmonDsSessionDuration,
                      raqmonDsSessionSetupStatus,
                      raqmonDsRoundTripEndToEndNetDelay,
                      raqmonDsOneWayEndToEndNetDelay,
                      raqmonDsApplicationDelay,
                      raqmonDsInterArrivalJitter,
                      raqmonDsIPPacketDelayVariation,
                      raqmonDsTotalPacketsReceived,
                      raqmonDsTotalPacketsSent,
                      raqmonDsTotalOctetsReceived,
                      raqmonDsTotalOctetsSent,
                      raqmonDsCumulativePacketLoss,
                      raqmonDsPacketLossFraction,
                      raqmonDsCumulativeDiscards,
                      raqmonDsDiscardsFraction,
                      raqmonDsSourcePayloadType,
                      raqmonDsReceiverPayloadType,
```

```
                    raqmonDsSourceLayer2Priority,
                    raqmonDsSourceDscp,
                    raqmonDsDestinationLayer2Priority,
                    raqmonDsDestinationDscp,
                    raqmonDsCpuUtilization,
                    raqmonDsMemoryUtilization }
          STATUS current
          DESCRIPTION
              "Standard RAQMON Data Source payload MIB objects group."
          ::= { raqmonDsGroups 2 }

      END
```

3. IANA Considerations

Applications using the RAQMON Framework require a single fixed port. Port number 7744 is registered with IANA for use as the default port for RAQMON PDUs over TCP. Hosts that run multiple applications may use this port as an indication to have used RAQMON or provision a separate TCP port as part of provisioning RAQMON RDS and RAQMON Collector.

The particular port number was chosen to lie in the range above 5000 to accommodate port number allocation practice within the Unix operating system, where privileged processes can only use port numbers below 1024 and port numbers between 1024 and 5000 are automatically assigned by the operating systems.

The OID assignment for the raqmonDsMIB MODULE-IDENTITY is made according to [RFC3737], and there is no need for any IANA action on this respect.

4. Congestion-Safe RAQMON Operation

As outlined in earlier sections, the TCP congestion control mechanism provides inherent congestion safety features when TCP is implemented as transport to carry RAQMON PDU.

To ensure congestion safety, clearly the best thing to do is to use a congestion-safe transport protocol such as TCP. If this is not feasible, it may be necessary to fall back to UDP since SNMP over UDP is a widely deployed transport protocol.

When SNMP is chosen as RAQMON PDU Transport, implementers MUST follow section 3 of [RFC4710], which outlines measures that MUST be taken to use RAQMON in a congestion-safe manner. Congestion safety

requirements in section 3 of [RFC4710] would ensure that a RAQMON implementation using SNMP over UDP does not lead to congestion under heavy network load.

5. Acknowledgements

The authors would like to thank Bill Walker and Joseph Mastroguilio from Avaya and Bin Hu from Motorola for their discussions. The authors would also like to extend special thanks to Randy Presuhn, who reviewed this document for spelling and formatting purposes, and who provided a deep review of the technical content. We also would like to thank Bert Wijnen for the permanent coaching during the evolution of this document and the detailed review of its final versions. The Security Considerations section was reviewed by Sam Hartman and Kurt D. Zeilenga and almost completely re-written by Mahalingam Mani.

6. Security Considerations

[RFC4710] outlines a threat model associated with RAQMON and security considerations to be taken into account in the RAQMON specification to mitigate against those threats. It is imperative that RAQMON PDU implementations be able to provide the following protection mechanisms in order to attain end-to-end security:

1. Authentication: The RRC SHOULD be able to verify that a RAQMON report was originated by the RDS claiming to have sent it. At minimum, an RDS/RRC pair MUST use a digest-based authentication procedure to authenticate, like the one defined in [RFC1321].

2. Privacy: RAQMON information includes identification of the parties participating in a communication session. RAQMON deployments SHOULD be able to provide protection from eavesdropping, and to prevent an unauthorized third party from gathering potentially sensitive information. This can be achieved by using secure transport protocols supporting confidentiality based on encryption technologies such as DES (Data Encryption Standard), [3DES], and AES (Advanced Encryption Standard) [AES].

3. Protection from DoS attacks directed at the RRC: RDSs send RAQMON reports as a side effect of external events (for example, receipt of a phone call). An attacker can try to overwhelm the RRC (or the network) by initiating a large number of events in order to swamp the RRC with excessive numbers of RAQMON PDUs.

To prevent DoS attacks against the RRC, the RDS will send the first report for a session only after the session has been established, so that the session set-up process is not affected.

4. NAT and Firewall Friendly Design: The presence of IP addresses and TCP/UDP port information in RAQMON PDUs may be NAT-unfriendly. Where NAT-friendliness is a requirement, the RDS MAY omit IP address information from the RAQMON PDU. Another way to avoid this problem is by using NAT-Aware Application Layer Gateways (ALGs) to ensure that correct IP addresses appear in RAQMON PDUs.

For the usage of TCP, TLS MUST be used to provide transport layer security. Section 6.1 describes the usage of TLS with RAQMON.

This memo also defines the RAQMON-RDS-MIB module with the purpose of mapping the RAQMON PDUs into SNMP Notifications. To attain end-to-end security, the following measures have been taken in the RAQMON-RDS-MIB module design:

There are no management objects defined in this MIB module that have a MAX-ACCESS clause of read-write and/or read-create. Consequently, if this MIB module is implemented correctly, there is no risk that an intruder can alter or create any management objects of this MIB module via direct SNMP SET operations.

Some of the readable objects in this MIB module (i.e., objects with a MAX-ACCESS other than not-accessible) may be considered sensitive or vulnerable in some network environments. It is thus important to control even GET and/or NOTIFY access to these objects and possibly to even encrypt the values of these objects when sending them over the network via SNMP. These are the tables and objects and their sensitivity/vulnerability:

raqmonDsNotificationTable

The objects in this table contain user session information, and their disclosure may be sensitive in some environments.

SNMP versions prior to SNMPv3 did not include adequate security. Even if the network itself is secure (for example by using IPsec), even then, there is no control as to who on the secure network is allowed to access and GET/SET (read/change/create/delete) the objects in this MIB module.

It is RECOMMENDED that implementers consider the security features as provided by the SNMPv3 framework (see [RFC3410], section 8), including full support for the SNMPv3 cryptographic mechanisms (for authentication and confidentiality).

It is a customer/operator responsibility to ensure that the SNMP entity giving access to an instance of this MIB module is properly configured to give access to the objects only to those principals (users) that have legitimate rights to indeed GET or SET (change/create/delete) them.

6.1. Usage of TLS with RAQMON

6.1.1. Confidentiality & Message Integrity

The subsequently authorized RAQMON data flow itself is protected by the same TLS security association that protects the client-side exchange. This standard TLS channel is now bound to the server through the above client-side authentication. The session itself is identified by the tuple {RDS ip-address:RDS_port / RRC ip-address: RRC port}.

6.1.2. TLS CipherSuites

Several issues should be considered when selecting TLS ciphersuites that are appropriate for use in a given circumstance. These issues include the following:

The ciphersuite's ability to provide adequate confidentiality protection for passwords and other data sent over the transport connection. Client and server implementers should recognize that some TLS ciphersuites provide no confidentiality protection, while other ciphersuites that do provide confidentiality protection may be vulnerable to being cracked using brute force methods, especially in light of ever-increasing CPU speeds that reduce the time needed to successfully mount such attacks.

Client and server implementers should carefully consider the value of the password or data being protected versus the level of confidentiality protection provided by the ciphersuite to ensure that the level of protection afforded by the ciphersuite is appropriate.

The ciphersuite's vulnerability (or lack thereof) to man-in-the-middle attacks. Ciphersuites vulnerable to man-in-the-middle attacks SHOULD NOT be used to protect passwords or sensitive data, unless the network configuration is such that the danger of a man-in-the-middle attack is negligible.

After a TLS negotiation (either initial or subsequent) is completed, both protocol peers should independently verify that the security services provided by the negotiated ciphersuite are adequate for the intended use of the RAQMON session. If not, the TLS layer should be closed.

Spoofing Attacks: When anonymous TLS alone is negotiated without client authentication, the client's identity is never established. This easily allows any end-entity to establish a TLS-secured RAQMON connection to the RRC. This not only offers an opportunity to spoof legitimate RDS clients and hence compromise the integrity of RRC monitoring data, but also opens the RRC up to unauthorized clients posing as genuine RDS entities to launch a DoS by flooding data. RAQMON deployment policy MUST consider requiring RDS client authentication during TLS session establishment, especially when RDS clients communicate across unprotected internet.

Insider attacks: Even client-authenticated TLS connections are open to spoofing attacks by one trusted client on another. Validation of RDS source address against RDS TLS-session source address SHOULD be performed to detect such attempts.

6.1.3. RAQMON Authorization State

Every RAQMON session (between RDS and RRC) has an associated authorization state. This state is comprised of numerous factors such as what (if any) authorization state has been established, how it was established, and what security services are in place. Some factors may be determined and/or affected by protocol events (e.g., StartTLS, or TLS closure), and some factors may be determined by external events (e.g., time of day or server load).

While it is often convenient to view authorization state in simplistic terms (as we often do in this technical specification) such as "an anonymous state", it is noted that authorization systems in RAQMON implementations commonly involve many factors that interrelate.

Authorization in RAQMON is a local matter. One of the key factors in making authorization decisions is authorization identity. The initial session establishment defined in Section 2.2 allows information to be exchanged between the client and server to establish an authorization identity for the RAQMON session. The RRC is not to allow any RDS-transactions-related traffic through for processing until the client authentication is complete, unless anonymous authentication mode is negotiated.

Upon initial establishment of the RAQMON session, the session has an anonymous authorization identity. Among other things, this implies that the client need not send a TLSStartRequired in the first PDU of the RAQMON message. The client may send any operation request prior to binding RDS to any authentication, and the RRC MUST treat it as if it had been performed after an anonymous RAQMON session start.

The RDS automatically is placed in an unauthorized state upon RRC sending a TLSstart request to the RRC.

It is noted that other events both internal and external to RAQMON may result in the authentication and authorization states being moved to an anonymous one. For instance, the establishment, change, or closure of data security services may result in a move to an anonymous state, or the user's credential information (e.g., certificate) may have expired. The former is an example of an event internal to RAQMON, whereas the latter is an example of an event external to RAQMON.

7. References

7.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2578] McCloghrie, K., Perkins, D., Schoenwaelder, J., Case, J., Rose, M., and S. Waldbusser, "Structure of Management Information Version 2 (SMIv2)", STD 58, RFC 2578, April 1999.

[RFC2579] McCloghrie, K., Perkins, D., Schoenwaelder, J., Case, J., Rose, M., and S. Waldbusser, "Textual Conventions for SMIv2", STD 58, RFC 2579, April 1999.

[RFC2580] McCloghrie, K., Perkins, D., Schoenwaelder, J., Case, J., Rose, M., and S. Waldbusser, "Conformance Statements for SMIv2", STD 58, RFC 2580, April 1999.

[RFC2819] Waldbusser, S., "Remote Network Monitoring Management Information Base", STD 59, RFC 2819, May 2000.

[RFC3289] Baker, F., Chan, K., and A. Smith, "Management Information Base for the Differentiated Services Architecture", RFC 3289, May 2002.

[RFC3411] Harrington, D., Preshun, R., and B. Wijnen, "An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks", STD 62, RFC 3411, December 2002.

[RFC4001] Daniele, M., Haberman, B., Routhier, S., and J. Schoenwalder, "Textual Conventions for Internet Network Addresses", RFC 4001, February 2005.

[RFC791] Postel, J., "Internet Protocol", STD 5, RFC 791, September 1981.

[RFC793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.

[RFC4710] Siddiqui, A., Romascanu, D., and E. Golovinsky, "Real-time Application Quality-of-Service Monitoring (RAQMON)", RFC 4710, October 2006.

[TLS] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.1", RFC 4346, April 2006.

7.2. Informative References

[3DES] Americation National Standards Institute, "Triple Data Encryption Algorithm Modes of Operation", ANSI X9.52-1998.

[AES] Federal Information Processing Standard (FIPS), "Specifications for the ADVANCED ENCRYPTION STANDARD(AES)", Publication 197, November 2001.

[IEEE802.1D] "Information technology-Telecommunications and information exchange between systems--Local and metropolitan area networks-Common Specification a--Media access control (MAC) bridges:15802-3: 1998(ISO/IEC)", [ANSI/IEEE Std 802.1D Edition], 1998.

[RFC1305] Mills, D., "Network Time Protocol Version 3", RFC 1305, March 1992.

[RFC1321] Rivest, R., "Message Digest Algorithm MD5", RFC 1321, April 1992.

[RFC3410] Case, J., Mundy, R., Partain, D., and B. Stewart, "Introduction and Applicability Statements for Internet-Standard Management Framework", RFC 3410, December 2002.

[RFC3414] Blumenthal, U. and B. Wijnen, "User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)", RFC 3414, December 2002.

[RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", RFC 3550, July 2003.

[RFC3551] Schulzrinne, H. and S. Casner, "RTP Profile for Audio and Video Conferences with Minimal Control", STD 65, RFC 3551, July 2003.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003.

[RFC3737] Wijnen, B. and A. Bierman, "IANA Guidelines for the Registry of Remote Monitoring (RMON) MIB modules", RFC 3737, April 2004.

[RFC4513] Harrison, R., "Lightweight Directory Access Protocol (LDAP): Authentication Methods and Security Mechanisms", RFC 4513, June 2006.

[TLS-PSK] Eronen, P. and H. Tschofenig, "Pre-Shared Key Ciphersuites for Transport Layer Security (TLS)", RFC 4279, December 2005.

Appendix A. Pseudocode

The implementation notes included in Appendix are for informational purposes only and are meant to clarify the RAQMON specification.

Pseudocode for RDS & RRC

We provide examples of pseudocode for aspects of RDS and RRC. There may be other implementation methods that are faster in particular operating environments or have other advantages.

RDS:

```
when (session starts} {
  report.identifier = session.endpoints, session.starttime;
  report.timestamp = 0;
  while (session in progress) {
    wait interval;
    report.statistics = update statistics;
    report.curtimestamp += interval;
    if encryption required
       report_data = encrypt(report, encrypt parameters);
    else
       report_data = report;
       raqmon_pdu = header, report_data;
    send raqmon-pdu;
  }
}
```

RRC:

```
listen on raqmon port
when ( raqmon_pdu received ) {
    decrypt raqmon_pdu.data if needed

    if report.identifier in database
       if report.current_time_stamp > last update
          update session statistics from report.statistics
       else
          discard report
 }
```

Authors' Addresses

Anwar Siddiqui
Avaya
307 Middletown Lincroft Road
Lincroft, NJ 80302
USA

Phone: +1 732 852-3200
EMail: anwars@avaya.com

Dan Romascanu
Avaya
Atidim Technology Park, Bldg #3
Tel Aviv, 61131
Israel

Phone: +972-3-645-8414
EMail: dromasca@avaya.com

Eugene Golovinsky
Alert Logic

Phone: +1 713 918-1816
EMail: gene@alertlogic.net

Mahfuzur Rahman
Samsung Information Systems America
75 West Plumeria Drive
San Jose, CA 95134
USA

Phone: +1 408 544-5559

Yongbum Yong Kim
Broadcom
3151 Zanker Road
San Jose, CA 95134
USA

Phone: +1 408 501-7800
EMail: ybkim@broadcom.com

Full Copyright Statement

Intellectual Property

Acknowledgement

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).